


Exploiting The Entity: XXE (XML External Entity Injection)

2.2k
SHARES

 Share

 Tweet

History

In the recent year, major tech giants, like Google, Facebook, Magento, Shopify, Uber, Twitter, and Microsoft, have undergone XML External Entity attacks on their major applications. One such vulnerability that has been around for many years is XML external entity injection or XXE. For example, this vulnerability can be used to read arbitrary files from the server, including sensitive files, such as the application configuration files. An XXE attack helped the hackers to gain read-only access on Google's production servers itself. So far, major vulnerabilities like SQL injection and Command injection have been playing a major role on the web application attacks. But XXE is also a major critical bug that helps the attacker gain access to the server itself. OWASP Top Ten standards also added the XXL as one of the critical vulnerabilities lists. This vulnerability is an important one to understand because it exists by default for many popular XML parsers. To best explain and demonstrate the exploitation of XXE, we must first start with the basics of XML. So let's dig in deeper.

Introduction

What is XML?

XML STANDS FOR eXtensible Markup Language

- XML is a markup language similar to HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML Tree Structure

Extensible Markup Language (XML) is a feature rich and widely used information exchange format and standard. The standard allows for defining the structure of the XML using a Document Type Declaration, or DTD. The DTD provides a mechanism for defining entities whose values can be substituted into the XML document contents. This is helpful when the entity value is used multiple times.

Here is a sample XML structure

It's all about entities:

XML specification [1] describes several types of so-called entities (we know many of them: entities are usually used for conducting attacks on XML, named XML eXternal Entity, XXE):

- Predefined entities
- Internal entities
- External entities

- External entities
- Internal parameter entities
- External parameter entities

So far, the third type of entities has been most frequently attacked (except for DoS): using various files of a file system as a source of an external entity, it was possible (not always) to read files of the file system via data output in XML or error output. Besides, it was possible to conduct DoS attacks, brute force the content of a parsed entity, read files via a Document Type Declaration (DTD), which, if error output was enabled, allowed displaying the content of the read file.

XML 1.0 standard defines the structure of an XML document. The standard defines a concept referred to as an entity, which is a storage unit of some type.

There are different types of entities, but the one we're focusing on is externally referenced. External entities are valuable to attackers because they can access local or remote content via declared system identifiers, which are a more critical attack on the web application.

So, let's see how it works

The way it works is simple, a SYSTEM identifier is declared. The identifier references the local file "/etc/passwd" which discloses all users of the machine. The result of the entity 'xxe' (which includes the results of /etc/passwd) is included within the application's failed login response.

For IIS servers:

XXE Attacks:

There are two primary types of XML injection:

- XXE attacks that include output within the server's response.
- Blind XXE - Attacks that process an entity, but do not include the results within the output. We must instead entice the application server to 'send us' the response.

Attacking XML Parsers

Upon receiving user-supplied requests, application servers parse the provided data and process it to perform some action. Examples include:

- Authentication
- Transferring money
- Updating a profile

Unfortunately, however, XML parsers are often times misconfigured and enable the processing of external XML entities when they did not intend to. In addition, no sort of input validation occurs, resulting in the ability to reference any content referenced by an entity. This misconfiguration can result in the ability to access local system resources.

Let's see a demonstration of how the XXE attack happens on the real-world application.

Proof of concept:

Let's see how XXE injection vulnerability affects a real-time web application. This vulnerability will allow us to access the password of the root user and help to privilege escalate on the main system, so, let's begin!

- **Step 1: Port scanning the application IP address**

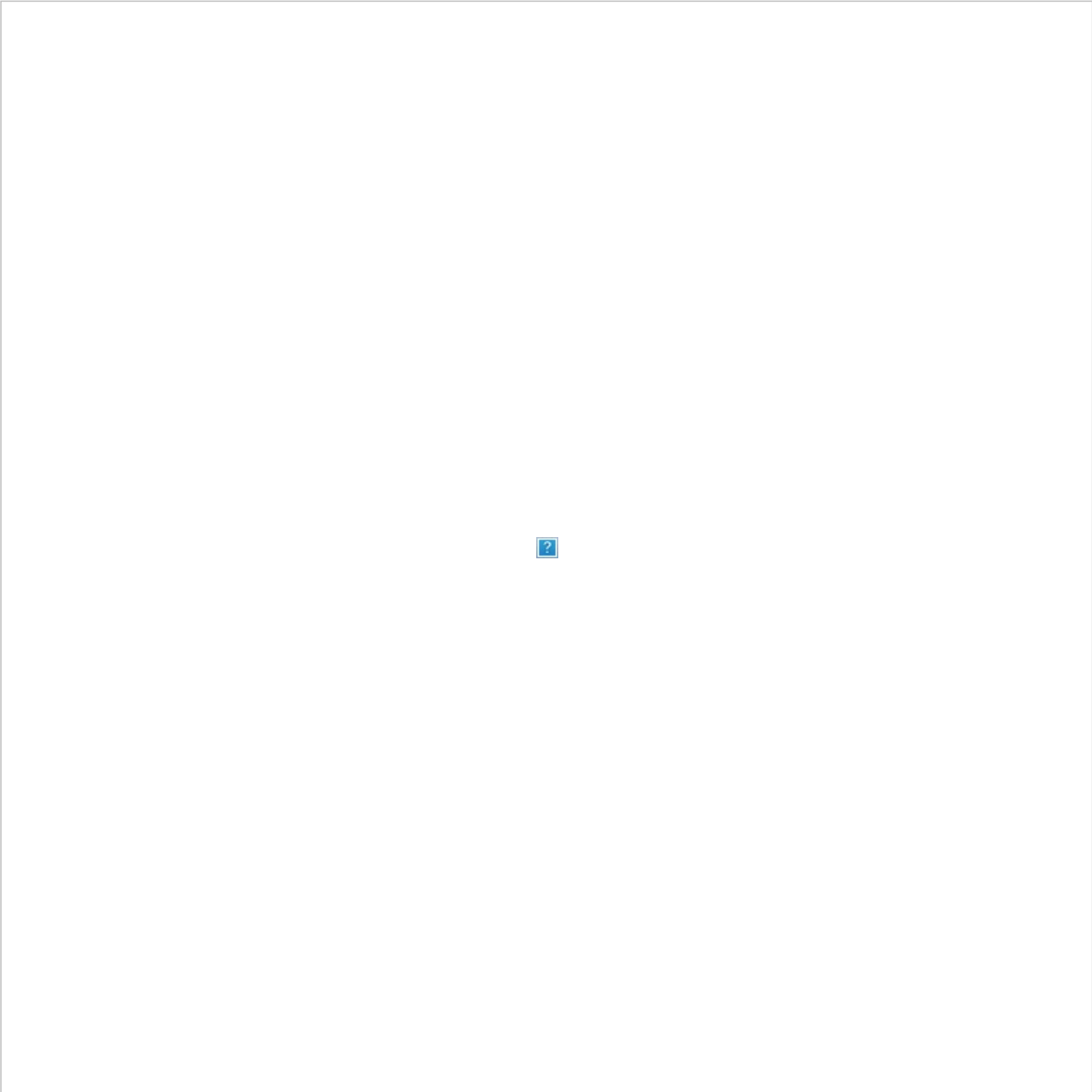
Port Scanning process is done the network IP address using nmap for enumeration process.

Command: Use Nmap -A <IP address> - in my case, I will run the command - **nmap -A 10.10.10.78**

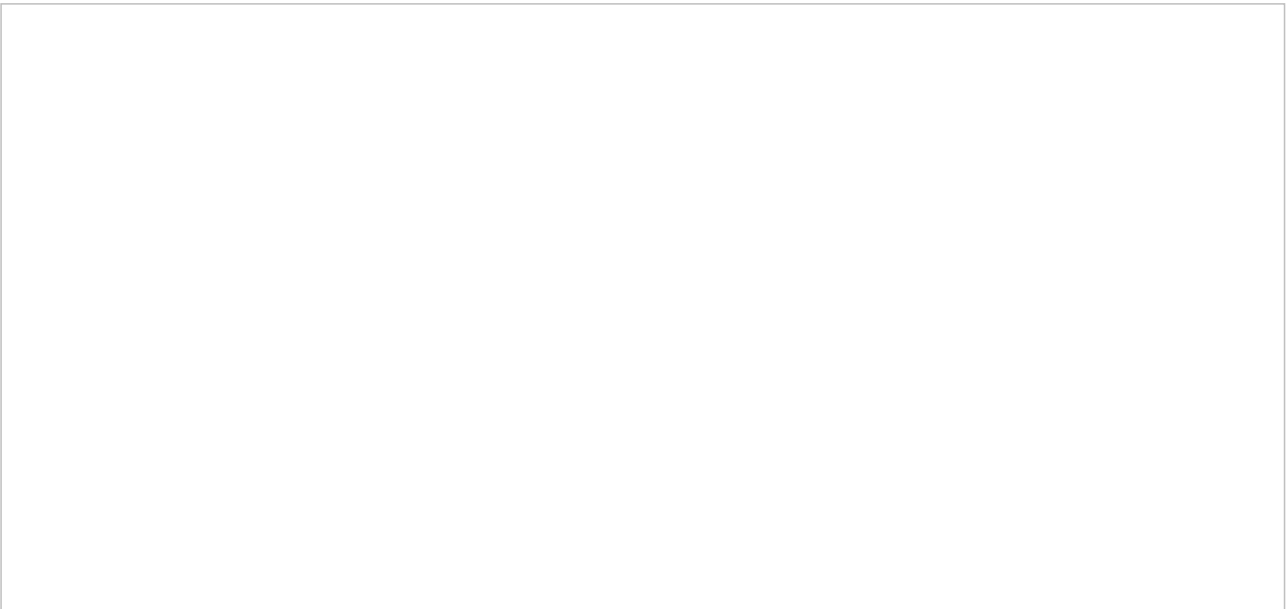
Ports Enumerated

In the above image, we can see that the **Ports 21,22,80** has been enumerated with useful information. Port 21 FTP has an anonymous FTP Login, which is a useful piece of information were we can log in without using the password and grab the test.txt file for any useful information.

• **Step 2: Logging to FTP with anonymous**



After logging into the ftp using anonymous login, we can clearly see a test.txt file, which will have valid information, using get command to download the file.





We clearly see that the test.txt file has XML data inside it, XML data contain as subnet information. Then open the target IP over web browser. By opening the IP address, we could clearly see the Apache2 Ubuntu Default port 80 web Page.


For a moment, we couldn't see any valid information on the page because it's an Apache default specification page and it doesn't have any pages apart from spec pages or links, so we need to try another method get any access to critical information of the system.

- **Step 3: Directory brute force using DirBuster**

When I found nothing on port 80, then I thought of using DirBuster so I was able to enumerate certain pages on the web directory brute force attack on the application. I found a page called **/hosts.php**.

- **Step 4: Intercepting using Burp suite**





Accessing the hosts.php file in the web browser found valid information - "There are 4294967294 possible hosts for" - about the host's system connected with the server as shown in the above image. So, searched in Google for 4294967294 hosts which were related to 255.255.255.192 as found in the above test.txt file.

It means that test.txt file can be used in the request to proceed with a possibility of XXE attack on the server of the web application.

So, let's capture the request and send the intercepted data into the repeater.

So, let's try the XML from the test.txt file we got from the FTP login. Add the XML content to the repeater and wait for a response to show the result.

We clearly see that XXE payload added to the XML from the test.txt, where it gets executed and luckily, we found that the application is vulnerable to XEE injection.

- **Step 5: Exploiting XXE payload on the application**

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>

<details>

<subnet_mask>&xxe;</subnet_mask>

<test></test>

</details>
```

Now we can simply exploit it to fetch the /etc/passwd file with the help of the following XXE script, then check its response.

The above image shows that we were able to access the etc/passwd files from the Ubuntu server. This clearly shows a successful attack and also enumerated two local usernames.

- **Step 6: Digging deeper to get more information to gain access**

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///home/florian/.ssh/id_rsa" >]>


<details>

    <subnet_mask>&xxe;</subnet_mask>

    <test></test>

</details>
```

By running the above XXE payload, we can get the id_rsa through XEE script mentioned above.





Finally, we got the **ssh** private key successfully, copy the key and save it as a text file. Then give permission 600 to the saved key (id-RSA) and then try to connect with SSH as we knew the port 22 is open in the Victim's network which was enumerated during port scanning.



Wow! Finally got into the system using SSH private key from the XXE payload, we gained complete access to the server.

The impact of this vulnerability shows that it is very dangerous, as it allows the attacker to gain complete access and take privilege over the system and perform denial of service attack on the server, etc.

Recommendation:

- XML parsers are vulnerable to XML external entity injection attack (XXE) by default. The best solution would be to configure the XML processor to use a local static DTD.
- Disallow any declared DTD included in the XML document.
- If external Entities aren't required then disable them completely.
- Sanitization process should be done for all users' input.
- Encode the user input in such a way that entities cannot be defined through user input.
- Use less complex data formats, such as JSON, and avoiding serialization of sensitive data.
- Patch or upgrade all XML processors and libraries in use by the application or on the operating system.
- Use a dependency checker. Update the SOAP to SOAP 1.2 or higher.
- Implement the positive whitelisting server-side input validation, filtering or sanitization to prevent hostile data within XML documents, header or nodes.
- Verify the XML or XSL file upload function for validation process.

Conclusion:

XXE is not a new vulnerability but an existing one that has gained more popularity in recent years on a web application. A successful XXE injection attack could result in massive damages on both security and business functionalities. Few better ways to control XXE attacks include. Depending upon the misconfiguration, this could lead to this major XXE attack on the application. Developers should have more concern on the security side, so that the application and the server can be protected to maximum extent. If these controls are not possible on the application, consider using virtual patching, API security gateways, Web Application Firewalls (WAF), or Interactive Application Security Testing (IAST) tools to detect, monitor, and block XXE attacks to prevent access by attackers.

References:

- [https://www.owasp.org/index.php/XML_External_Entity_\(XXE\)_Processing](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing)
- http://www.ws-attacks.org/XML_External_Entity_DOS
- [https://www.owasp.org/index.php/XML_External_Entity_\(XXE\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet)

Written by Anand M



Email: anand@infysec.com

Company: www.infysec.com

© SEPTEMBER 6, 2018

✉ Subscribe ▼

➔ Login



This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

2 COMMENTS

⚡ 🔥 Newest ▼



nutthaphon ⌚ 3 years ago

test.txt in captured image it show 255.255.255.192 not 255.255.255.254 from explanation , It must be the same? or I am misunderstanding

+ 0 - ➔ Reply



Andrewjerome ⌚ 3 years ago

🗨 Reply to *nutthaphon*

Its a typo error, it should be 255.255.255.192 only @nutthaphon, Apart from that this article is so good. Keep it up, Man. Personally you showed some real time attack, apart from other ppl showing DVWA and other vulnerable applications.

+ 0 - ➔ Reply

SEARCH

Newsletter

Signup for newsletter to receive free Articles !

☐ Yes, please sign me up for newsletters. This includes offers, latest news, and exclusive promotions

Subscribe

FREE CONTENT

MOST POPULAR



Ettercap and middle-attacks tutorial



TOP 5 Latest Cyber Security Books (2017-2019) | Best & Latest Must-Reads For Any Aspiring or Seasoned Hacker



Metasploit Cheat Sheet



Julia: a Language for the Future of Cybersecurity



How I Hacked Into Your Corporate Network Using Your Own Antivirus Agent

RELATED BRANDS

HAKINS

eForensics
M a g a z i n e

OUR PRODUCTS

COMPANY

SUPPORT

MORE

We use cookies to offer you a better browsing experience, analyze site traffic, personalize content, and serve targeted advertisements. Read about how we use cookies and how you can control them by clicking "Privacy Preferences". If you continue to use this site, you consent to our use of cookies.

© HAKIN9 MEDIA SP. Z O.O. SP. K. 2013

[Privacy Preferences](#)

I Agree

