XXE Injection

Web App Vulnerabilities

What is it?

XXE injection can be used on web applications that parse XML input. An attacker able to submit XML can make use of references to external entities. The attack occurs when the XML processes that input.

What causes it?

XXE injections occur when not properly sanitized input is being processed by an XML parser with default or weakly configured settings.

What could happen?

An attacker can be able to read arbitrary files from the server, perform internal port scanning originating from the server and cause other system impacts like a denial of service.

How to prevent it?

Sanitize user input through filtering or validation. XML parsers should disable support for external entities (DTD).

XXE Injection

Understanding the security vulnerability

Reading arbitrary files

A website uses XML to submit data for it’s reset password functionality.

```
POST /forgot_pass HTTP/1.1
Host: shop.localstore.com
User-Agent: Mozilla/5.0
Accept-Language: en-US
Connection: keep-alive

<forgot_pass>
<user>admin@localstore.com</user>
</forgot_pass>
```

An attacker intercepts the xml request and changes it to in order to access arbitrary files on the server.

```
POST /forgot_pass HTTP/1.1
Host: shop.localstore.com
User-Agent: Mozilla/5.0
Accept-Language: en-US
Connection: keep-alive

<?xml version="1.0" ?>
<!DOCTYPE forgot_pass [
<!ELEMENT field ANY>
<!ENTITY testxxe SYSTEM "file:///etc/passwd">
]>
<forgot_pass>
<user>
<field name="id">&textxxe;</field>
</user>
</forgot_pass>
```

The web server parses the xml and returns the /etc/passwd file.

```
root:!:0:0::/:/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
```

XXE Injection

Realizing the impact

XXE could lead to an attacker scanning and mapping your entire network.

An attacker could be able to retrieve the documents from the web server, resulting in compromised data.

System unavailability could cause revenue and reputation loss.

XXE Injection

Preventing the mistake

Never trust user input!

Apply application-wide **filters** or **sanitization** on **all** user-provided input.

GET and POST parameters, Cookies and other HTTP headers.

Apply white-list **input validation**.

Disable external entities (DTD) completely.

Check framework specific settings.