**Headless walkthrough**

# **Index**

[Index 1](#_Toc157089938)

[List of pictures 1](#_Toc157089939)

[Disclaimer 2](#_Toc157089940)

[Reconnaissance 2](#_Toc157089941)

# **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:

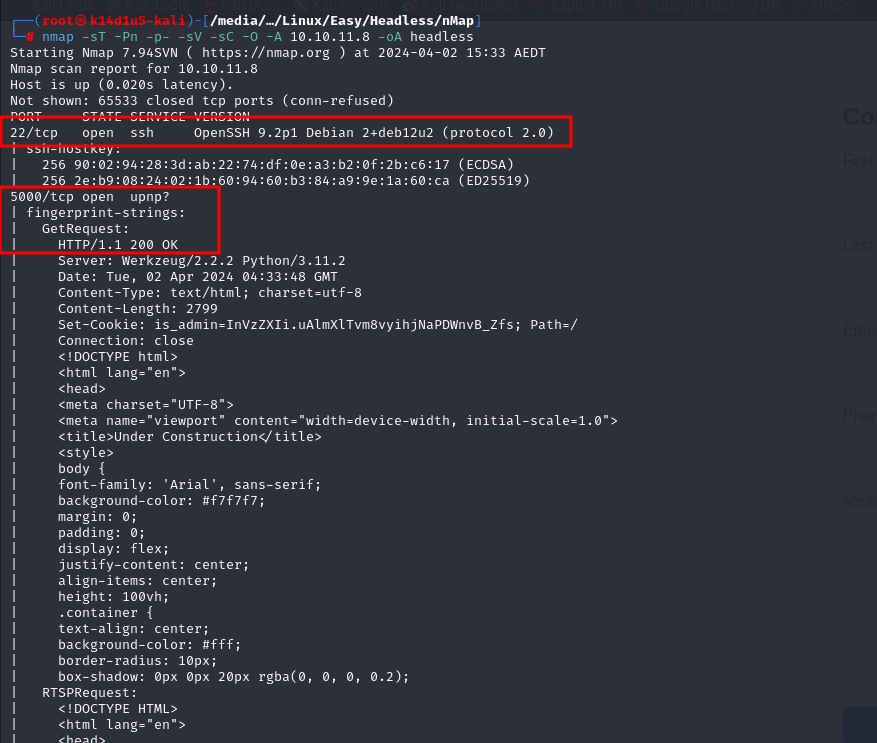


Figure 1 - nMap scan results

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:

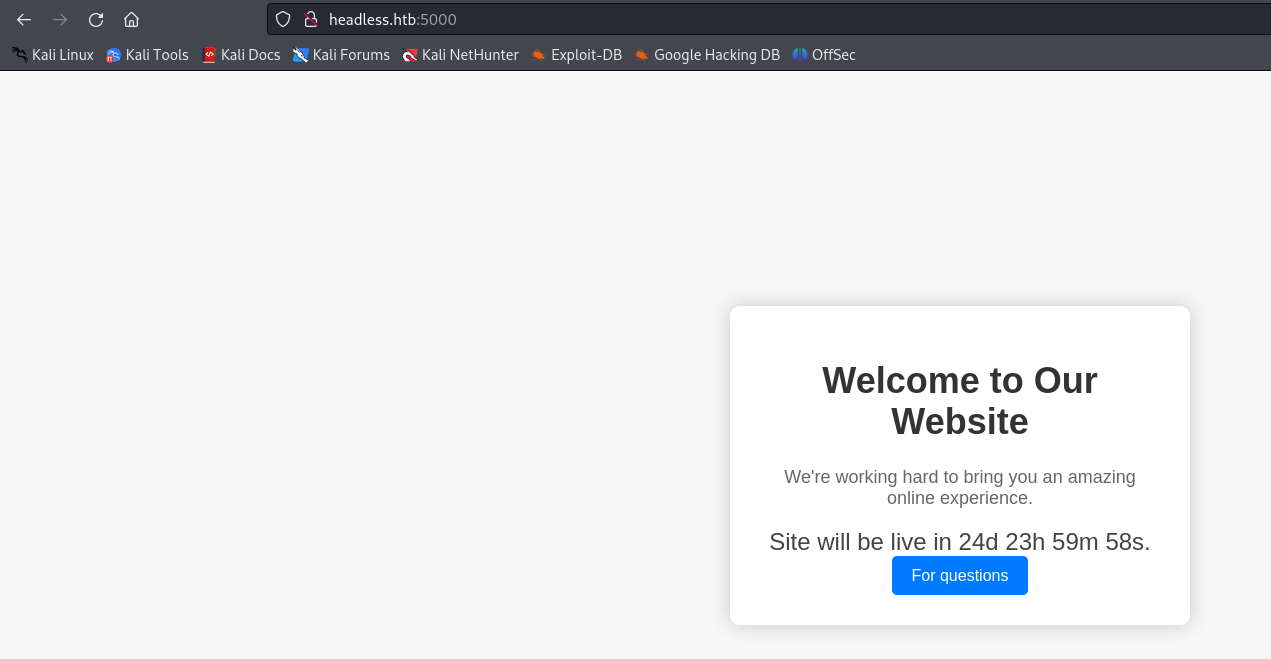


Figure 2 - Web application home page

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

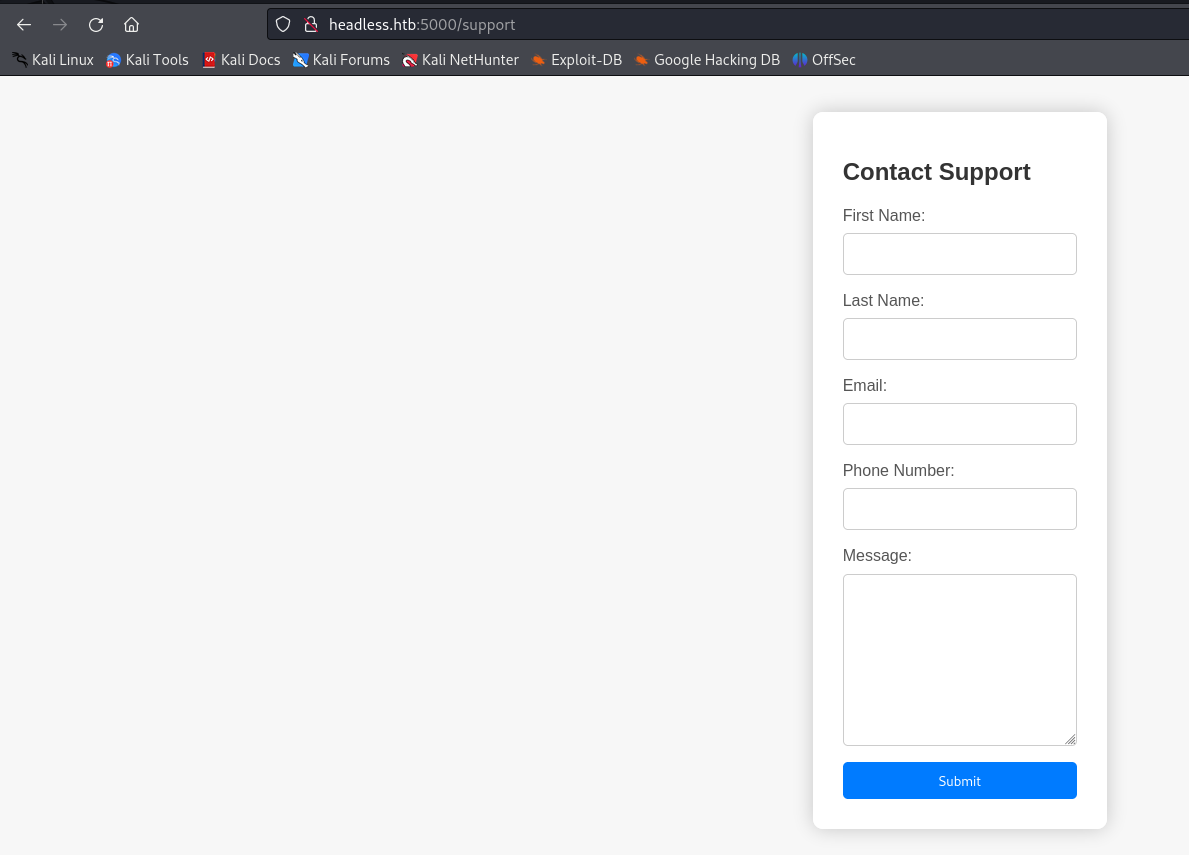


Figure 3 - Web application support form

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):

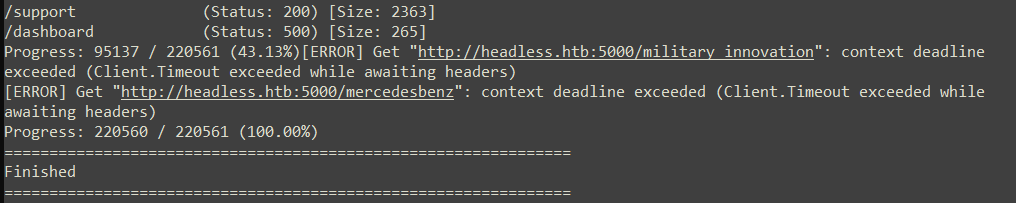


Figure 4 - Gobuster scan results

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:

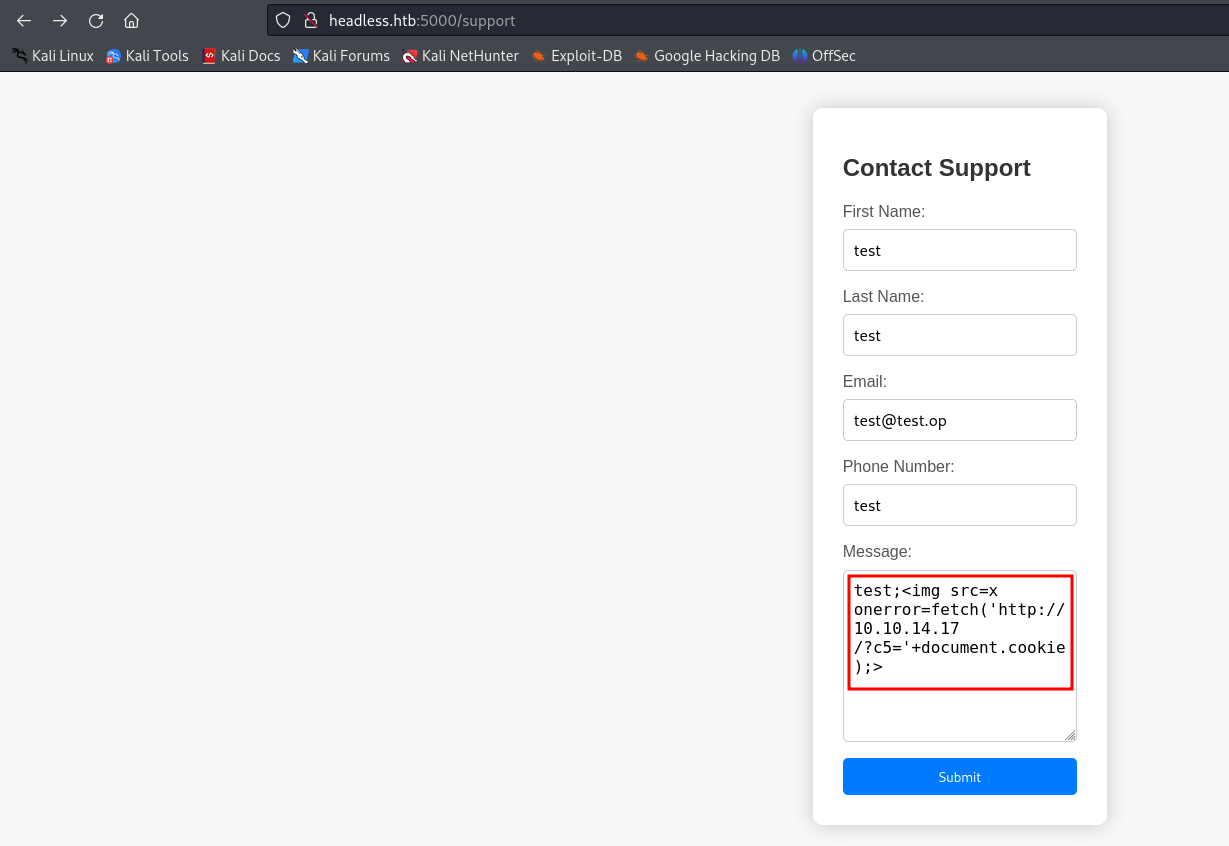


Figure 5 - XSS injection in message body

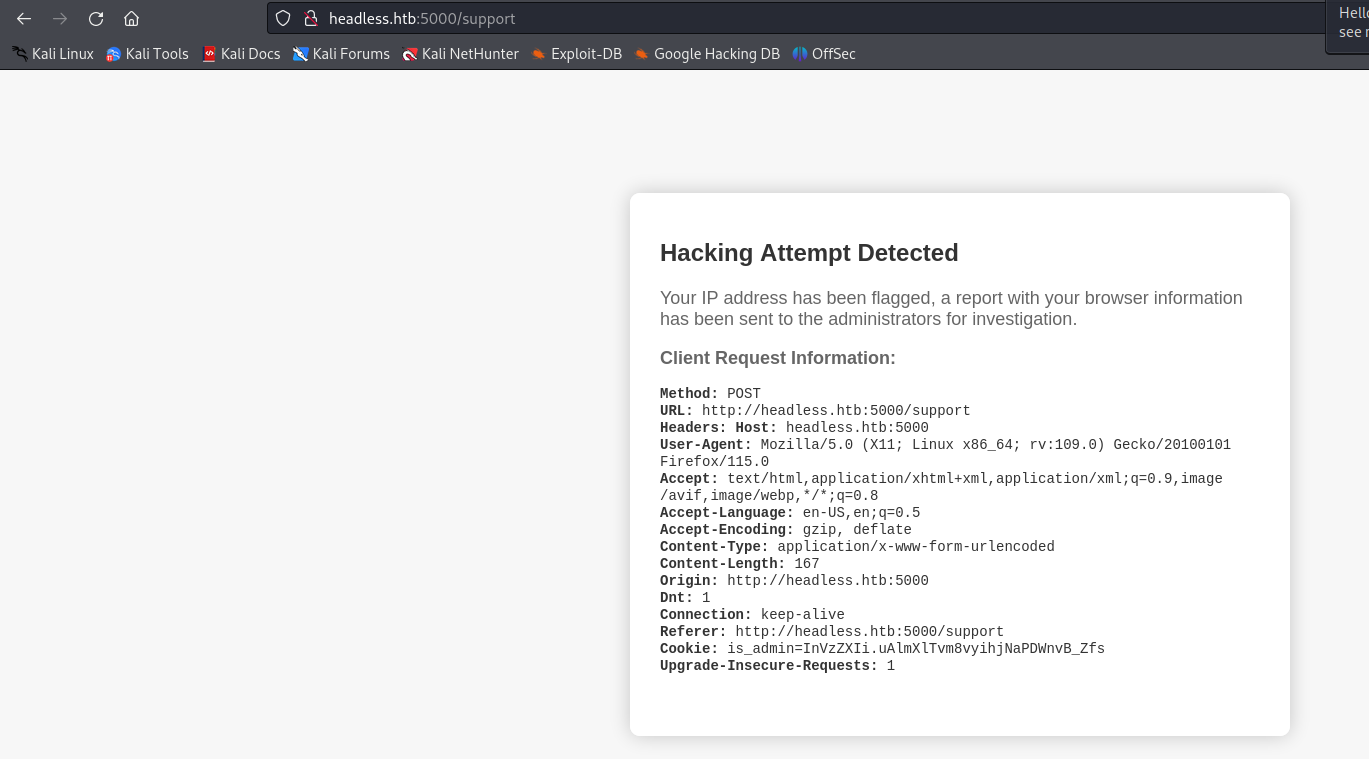


Figure 6 - XSS detection

# **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:

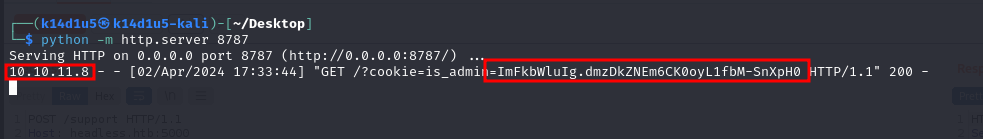


Figure 8 - Stealed cookie

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

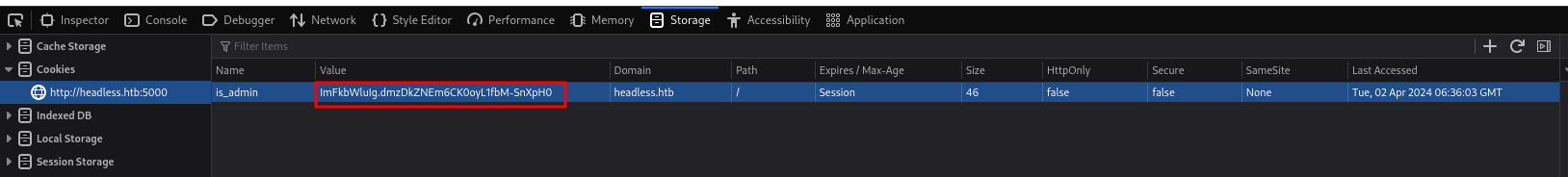


Figure 9 - Cookie set in the web application

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

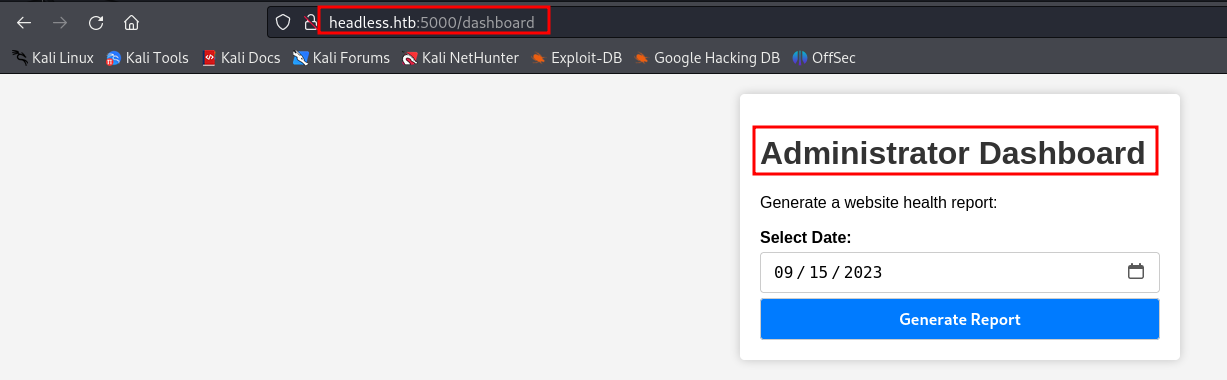


Figure 10 - Administration dashboard

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:

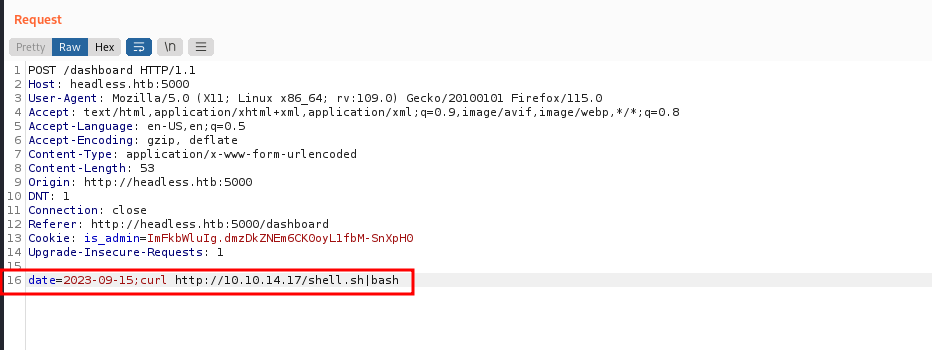


Figure 11 - Command injection

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

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:

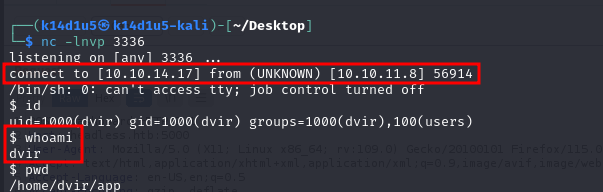


Figure 12 - User shell

At this point, I simply retrieved the user flag:

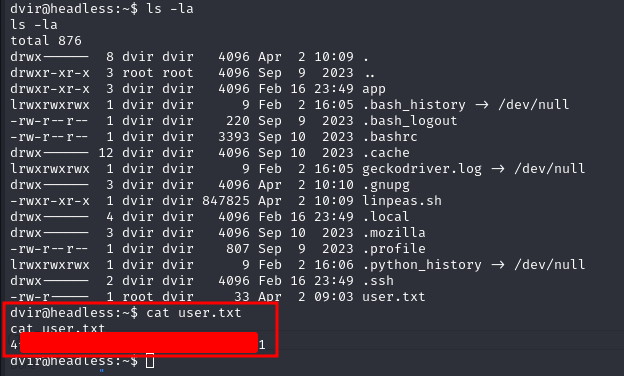


Figure 13 - User flag

# **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 command:

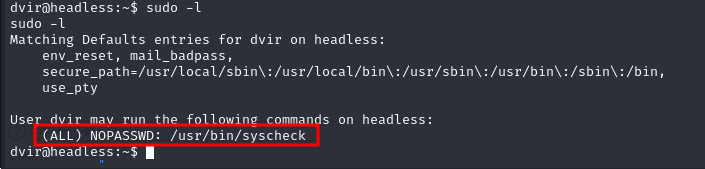


Figure 14 - Script root executable without providing password

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:

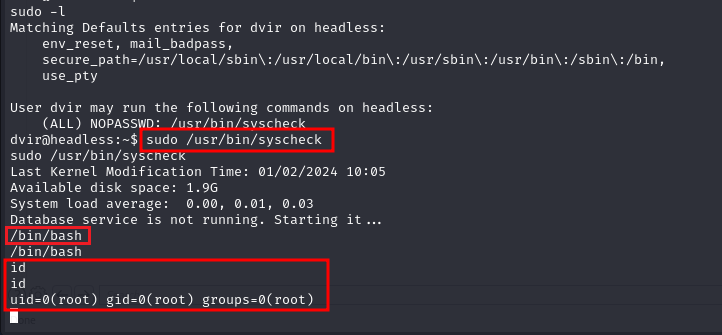


Figure 15 - Privilege escalation

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

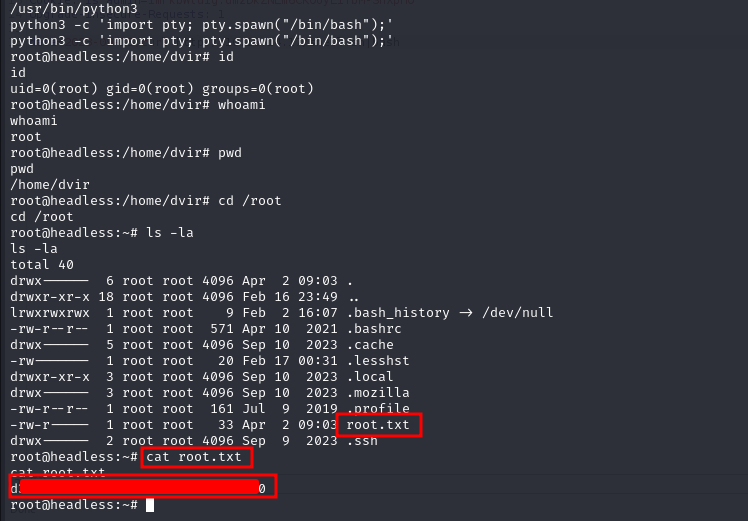


Figure 16 - Root flag