**Admirer walkthrough**

# **Index**

[Index 1](#_Toc176023706)

[List of pictures 1](#_Toc176023707)

[Disclaimer 2](#_Toc176023708)

[Reconnaissance 2](#_Toc176023709)

[Initial foothold 2](#_Toc176023710)

[User flag 5](#_Toc176023711)

[Privilege escalation 9](#_Toc176023712)

[Personal comments 12](#_Toc176023713)

[Appendix A – CVE-2021-43008 12](#_Toc176023714)

[References 13](#_Toc176023715)

# **List of pictures**

[Figure 1 - nMap scan results 2](#_Toc176023682)

[Figure 2 - Dirbuster results scan 3](#_Toc176023683)

[Figure 3 - FTP login 3](#_Toc176023684)

[Figure 4 - Clue of using an PHP database manager open source 4](#_Toc176023685)

[Figure 5 - New page found 4](#_Toc176023686)

[Figure 6 - Adminer home page 4](#_Toc176023687)

[Figure 7 - Local MySQL configuration 5](#_Toc176023688)

[Figure 8 - Local DB with an user on it 6](#_Toc176023689)

[Figure 9 - Evil connection 6](#_Toc176023690)

[Figure 10 - Query to import the index page 7](#_Toc176023691)

[Figure 11 - Current database credentials 8](#_Toc176023692)

[Figure 12 - SSH login and user flag 9](#_Toc176023693)

[Figure 13 - Interestin information for privilege escalation 9](#_Toc176023694)

[Figure 14 - Script invoked 9](#_Toc176023695)

[Figure 15 - Backup.py script 10](#_Toc176023696)

[Figure 16 - Exploit script uploaded on the target 10](#_Toc176023697)

[Figure 17 - Sudo crontabs 11](#_Toc176023698)

[Figure 18 - Running exploit 11](#_Toc176023699)

[Figure 19 - Root shell and root flag 12](#_Toc176023700)

# **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:

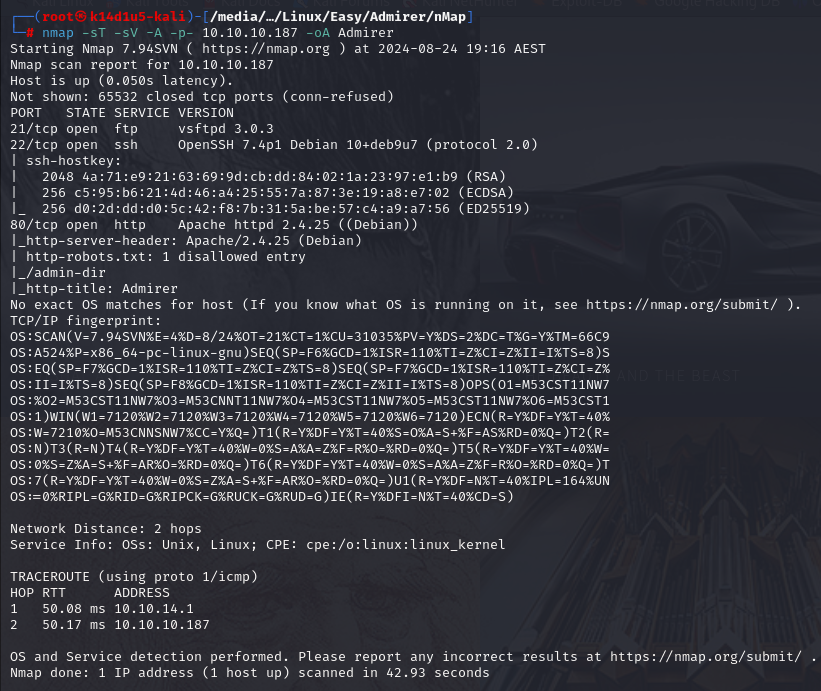


Figure 1 - nMap scan results

Open ports are 21, 22 and 80. So, this box has FTP (port 21) and SSH (port 22) services enabled and a web application running on port 80. Also, nMap has identified Linux as Operative System, but it didn’t identify the OS version.

# **Initial foothold**

The first thing I tried was accessing anonymously to FTP service. But it didn’t work. So, I started to search some “hidden content” on the web application. I was able to access to the file, and I found the path. However, this path require authentication. In the meanwhile, I run the Dirbuster tool (in particular in the path I previously found). In this way I found the following files:

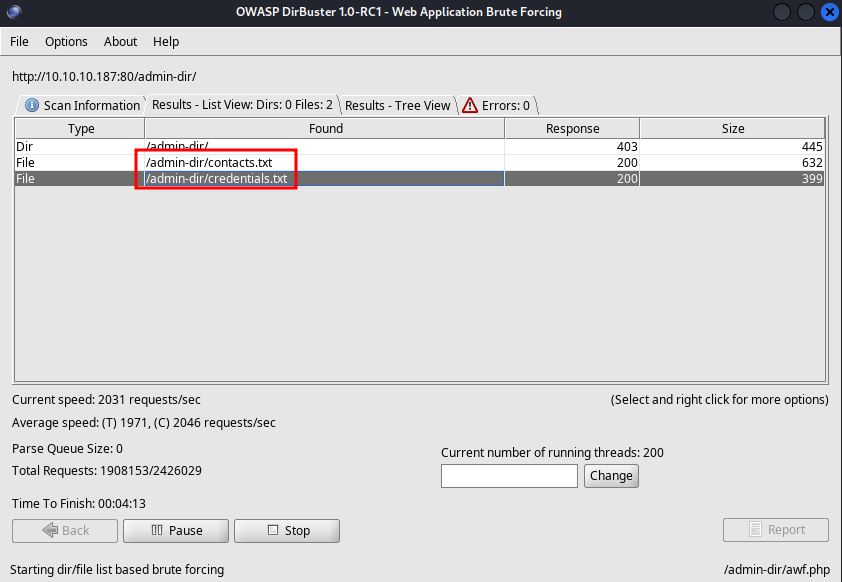


Figure 2 - Dirbuster results scan

In these files I found a list of plausible nicknames and credentials. In particular, in the file I found plausible FTP credentials, so I tried them. Luckly, they worked:

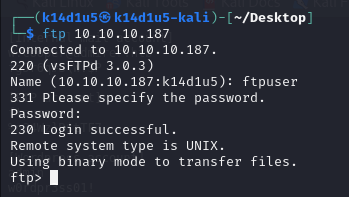


Figure 3 - FTP login

Since I have an FTP access, I download all files I can (an .tar.gz and an .sql file). The .tar.gz file contain a website backup with some other plausible credentials. So, I took note of all I found. However, I just found in the file in the .tar.gz archive that it could be used a different PHP database manager than PHPMyAdmin:

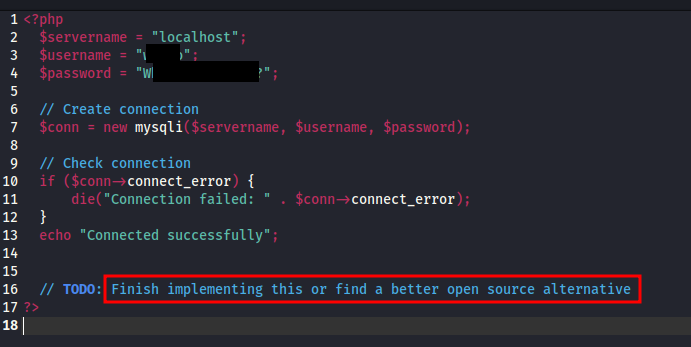


Figure 4 - Clue of using an PHP database manager open source

Also, this file was in a specific directory () in the archive. This path is present on the published website too. I search for a while for other useful information, but nothing. At this point, I tried to find new “hidden content” in the new path I found. In this way, I found a new page:

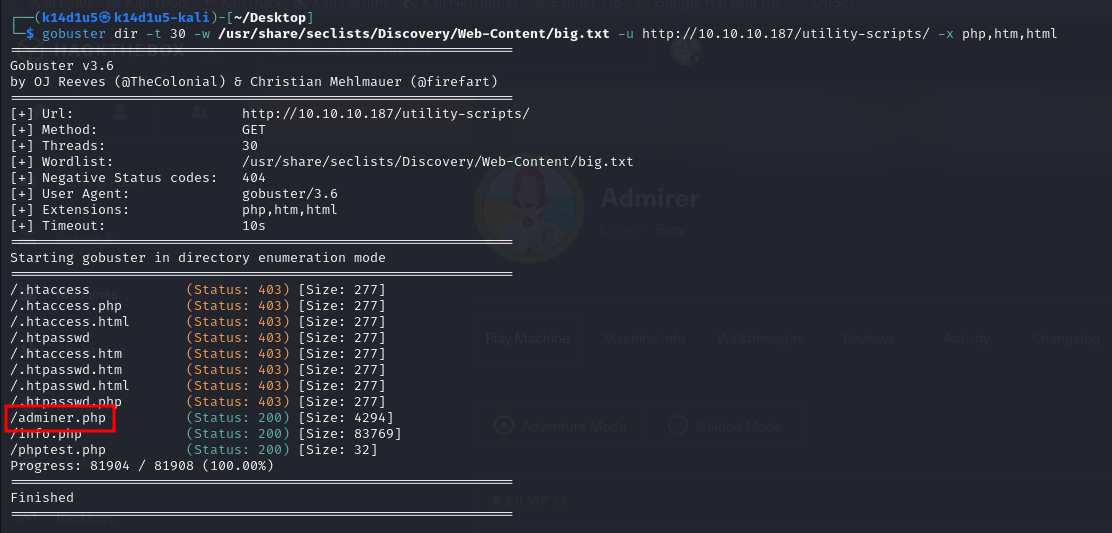


Figure 5 - New page found

This page let me to connect to a specific database of my choice:

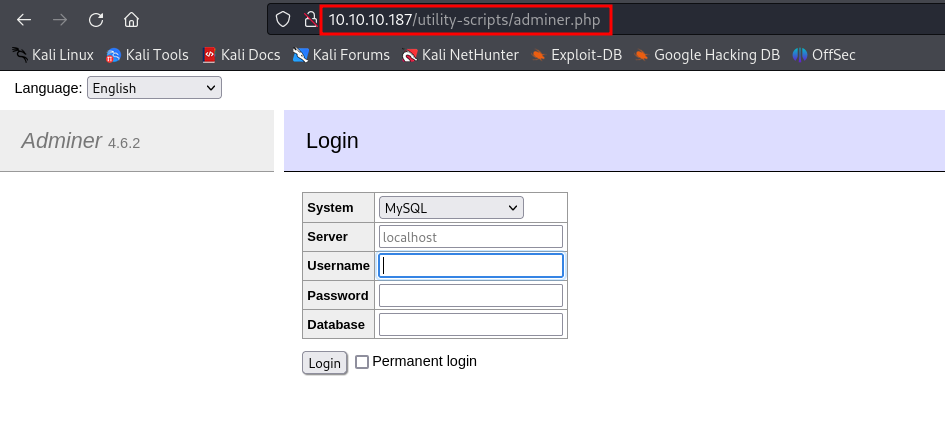


Figure 6 - Adminer home page

# **User flag**

Since I have the service (Adminer) and the version, I looked for some plausible exploit on the Internet. I found the **CVE-2021-43008**. To leverage it against the target, I need to create a target DB to which the target has to connect to. To do it, I need to configure the MySQL configuration and create the database. I modified the MySQL configuration as following:

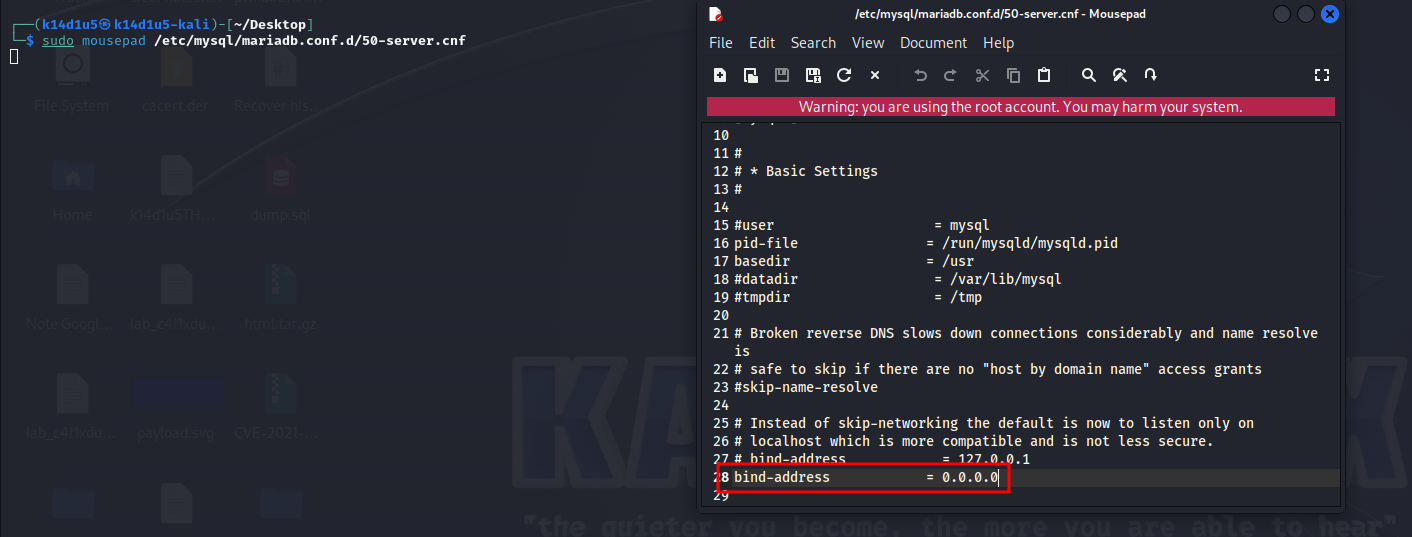


Figure 7 - Local MySQL configuration

To create a new database and create a user on it, instead, I run the following commands:

as shown in the following figure:

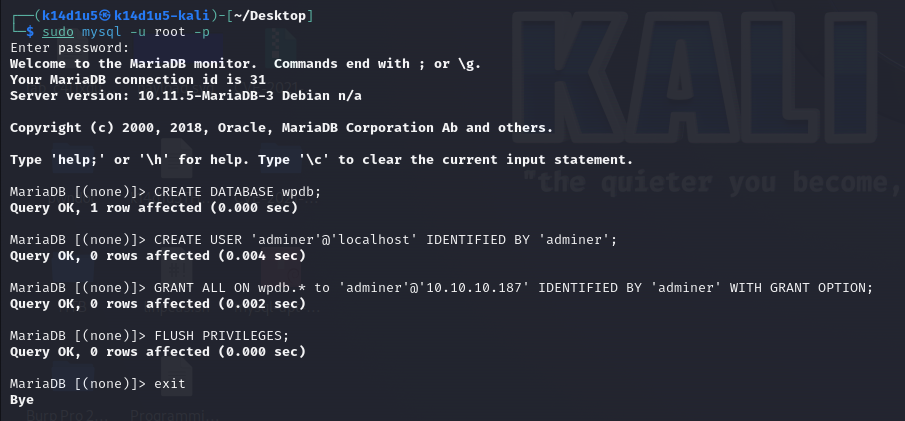


Figure 8 - Local DB with an user on it

At this point, I need to let the target to connect to my database in the following way:

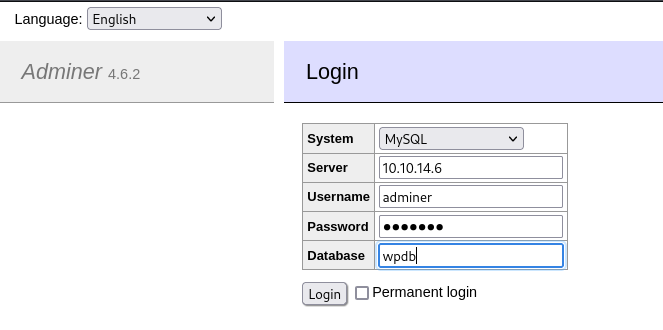


Figure 9 - Evil connection

So, I can import on my DB all files the interface can access to. For example, a very interesting file to import in this case, is the index page. I need to execute the following query to accomplish this task:

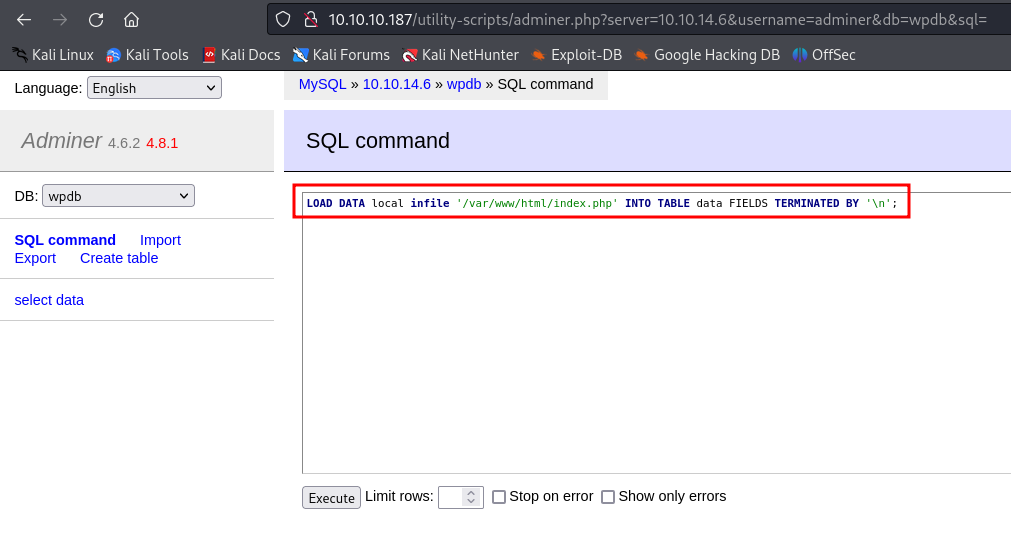


Figure 10 - Query to import the index page

When I analyzed this page, I found some interesting new credentials:

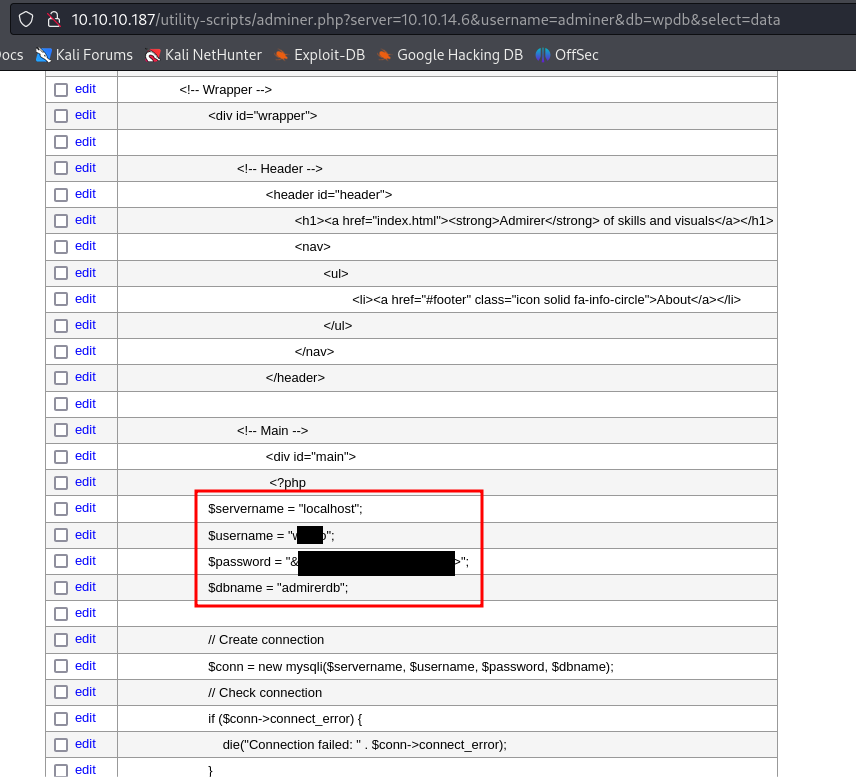


Figure 11 - Current database credentials

At this point, I just tried to use these credentials to login to the target machine via SSH. They worked and finally I retrieved the user flag:

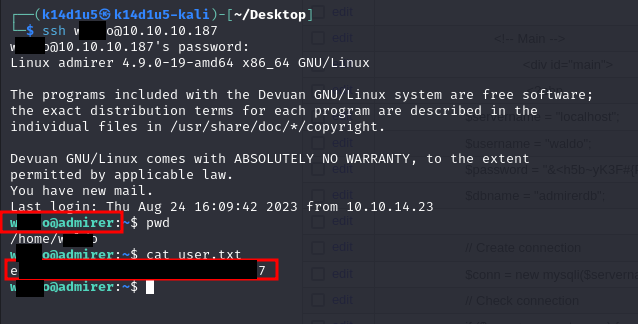


Figure 12 - SSH login and user flag

# **Privilege escalation**

Looking for some useful information to escalate my privileges on the target machine, I found that my current user can run a script as sudo using the option:

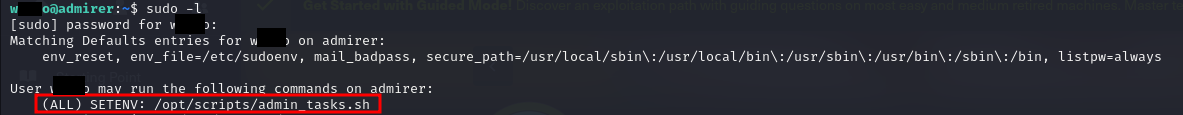


Figure 13 - Interesting information for privilege escalation

Analyzing the script, I noted that it invokes a different script in a specific use case:

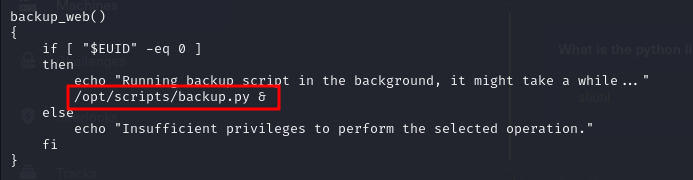


Figure 14 - Script invoked

At this point I need to investigate this new script I found. In this way, I found out that the script import the library:

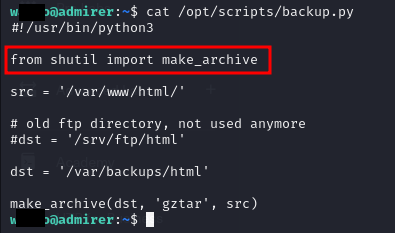


Figure 15 - Backup.py script

So, I can try to inject this library. To do it, I simply need to develop a python script that has a function with the same prototype used by the script. My exploit script needs to be named as the library to inject, . Also, I need an file to use my script as library:

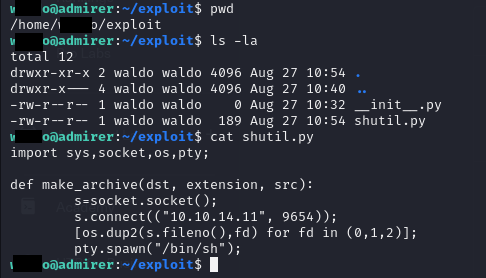


Figure 16 - Exploit script uploaded on the target

Please, note that I uploaded exploit files in a new directory called exploit. In fact, during my analysis I found some crontab that delete python files in the user home directory and all files with extension in the directory, as shown in the following figure:

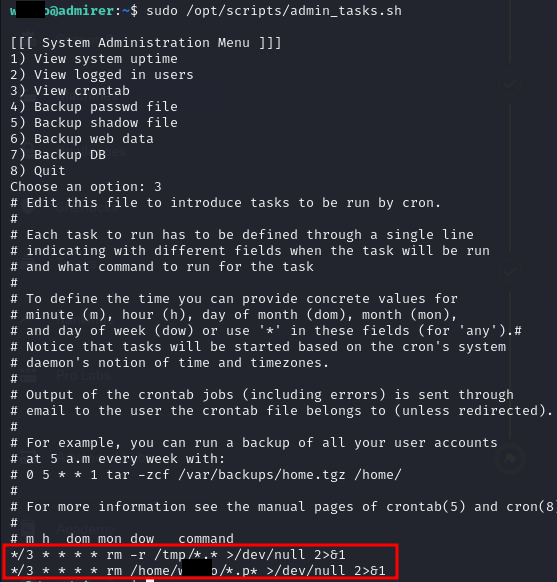


Figure 17 - Sudo crontabs

So, at this point I just needed to run my exploit with a listener ready to accept the incoming connection:

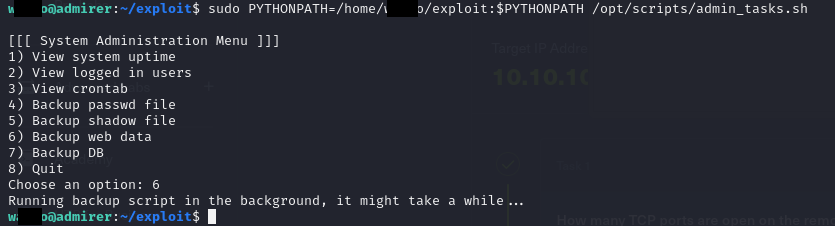


Figure 18 - Running exploit

Everything was successful, so I obtained a shell as root and I retrieved the root flag:

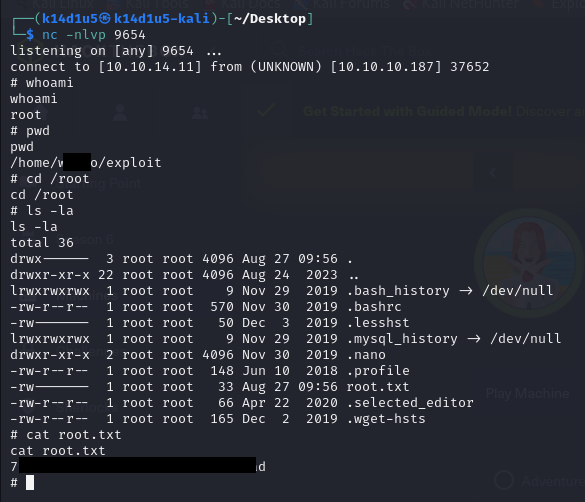


Figure 19 - Root shell and root flag

# **Personal comments**

This box was very interesting and taught me some important concepts. I never heard about the SETENV options, so I needed to study it to understand how I could use it in the penetration testing activities. I consider this box very close to a verisimilar real scenario. So, I enjoyed very much to resolve and pwned it. In this case, I surely improved my penetration tester skills and my knowledge is wider and wider. In conclusion, I rated as **medium** on the HackTheBox website because it requires some specific knowledge to complete it and you pay attention about which wordlist you have to use to find the useful information. In fact, after I pwned this box, I created a personal all-inclusive wordlist to find web hidden content.

# **Appendix A – CVE-2021-43008**

[Adminer](https://www.adminer.org/) is a popular PHP tool to administer MySQL and PostgreSQL databases. However, it can be lured to disclose arbitrary files. Attackers can abuse that to fetch passwords for popular apps such as Magento and WordPress, and gain control of a site's database.

Exploitation happens in three stages:

* First, the attacker needs a modified MySQL server, which is altered to send out data import requests to any client that connects.
* Second, an attacker needs to find an open adminer.php on the victim system. That is not hard, as many people install it in the root of their site. Once found, the attacker can instruct Adminer to connect to his rigged MySQL server (external connections are actually a feature of Adminer). Adminer will then connect to the foreign server, login with the credentials, and immediately receive a data import request from the server for a specific file.
* Third, as the attacker now has the master password for the victim site, he can use the same Adminer to access the database of the victim. And continue to steal private data or inject a skimmer.

# **References**

<https://sansec.io/research/adminer-4.6.2-file-disclosure-vulnerability>