SneakyMailer walkthrough

Index

List of pictures

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 are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

Reconnaissance

The results of an initial nMap scan are the following:

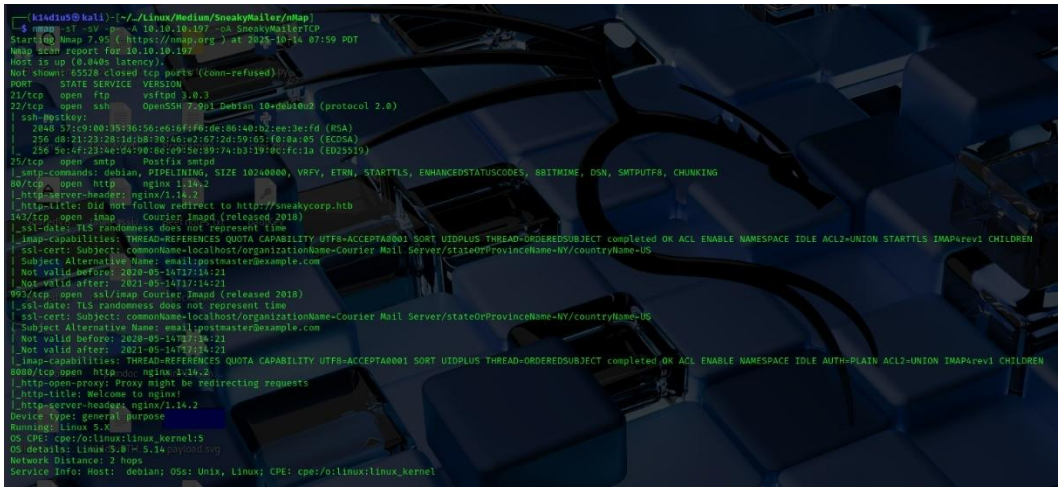

Figure 1 - nMap scan results

Open ports are 21, 22, 25, 80, 143, 993 and 8080. Therefore, enabled services are FTP (21), SSH (22), SMTP (25), IMAP (143 and 993) and there two web servers running on port 80 and 8080. Also, nMap recognized Linux 5.0-5.14 as operative system.

Initial foothold

First of all, I analyzed the web application on port 80. In particular, I found out a new subdomain:

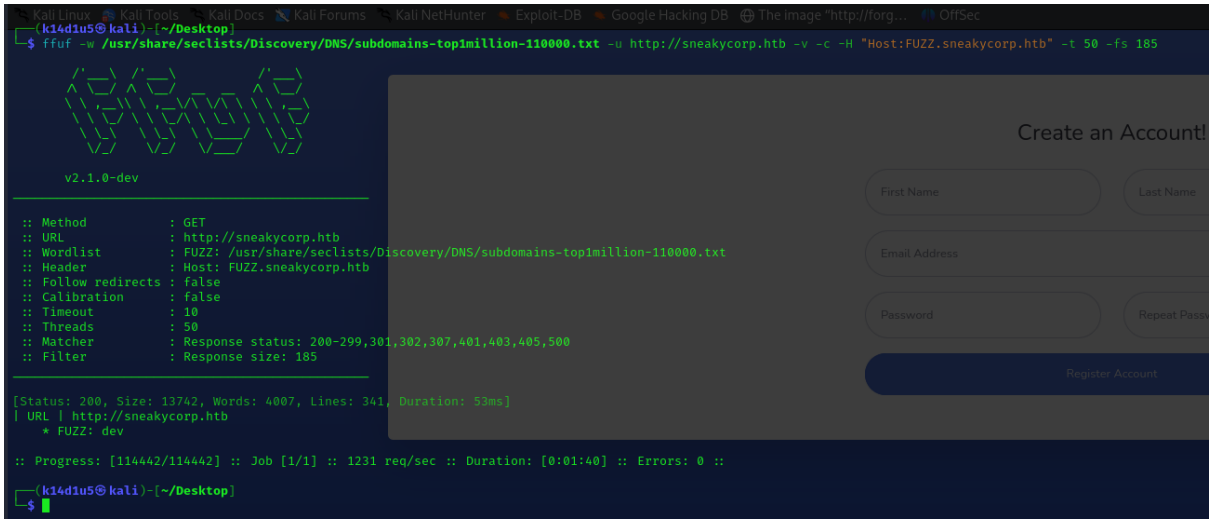

Figure 2 - New subdomain found

On the web application I found out a list of names and e-mails. I kept to look around the web application but I didn't find anything useful.

User flag

After a while, I thought that I could be able to leverage the e-mail addresses I found via phishing. Therefore, I developed a little python script that let me to send e-mails and I named it $testSendMail.py$. I built a list of receivers using all e-mail addresses I found on the web application. Also, I inserted in the body of the e-mail a link to connect to my attacker machine. In particular, I needed to open a $netcat$ listener to properly receive an answer. In fact, I was lucky and surprised when I received back a POST request. I was even more surprised when I saw that it was a POST to change a password and I found some credentials:

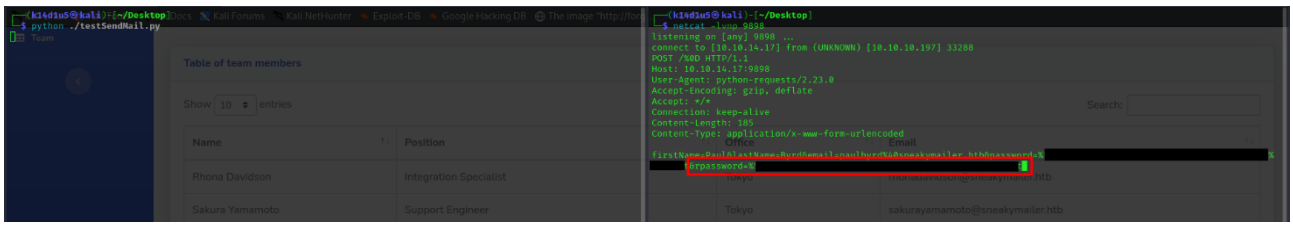

Figure 3 - Password leaked

Of course, I decode the password from the URL-encoding. I tried to use this password on all services and using some plausible usernames. However, no one of these tries worked at first glance. After a little bit of work and research, I finally found out a way to use credentials to log in the IMAP service. First of all, I needed to convert the pair username-password in base64, because the password contained some special characters. At this point I connected to the IMAP service running the command $openssl\ s\_client\ -crlf\ -connect\ 10.10.10.197{:}993$. After that I provided the command $A002\ AUTHENTICATE\ PLAIN$ to perform authentication. Finally, I provided the base64 generated before from the credentials I found. Finally, I got it, I logged in the user e-mail. At this point I checked the inbox running the command $A1\ LIST\ "INBOX"\ "*"$ and I found that some folder existed. I explored any of them, but the more interesting was the sent e-mails. I was able to access to them running the command $G21\ SELECT\ "INBOX.Sent\ Items"$. In this way I found out that he sent two e-mails. I read them and one of them provided a new pair of credentials, as shown in the following:

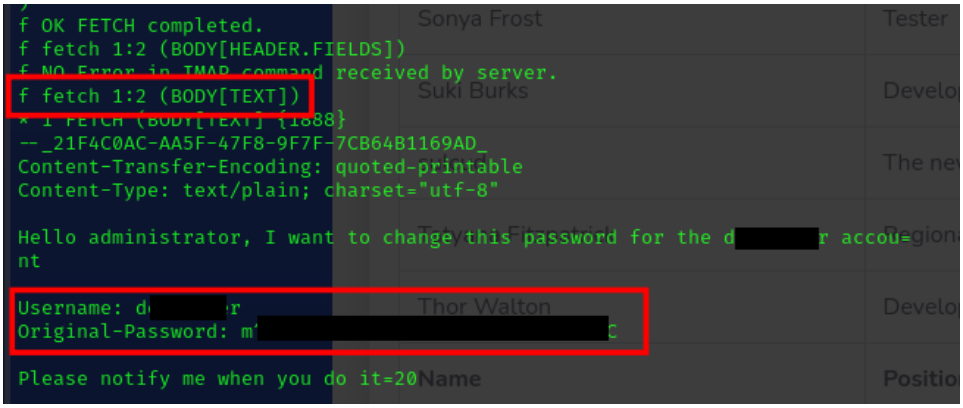

Figure 4 – First new credentials found

Also, the second e-mail let me know that the user's tasks were to install, test and erase the python modules found in the PyPI service. At this point, I tried to use the new credentials to log in a different service. This time I was lucky and I was able to log in FTP service:

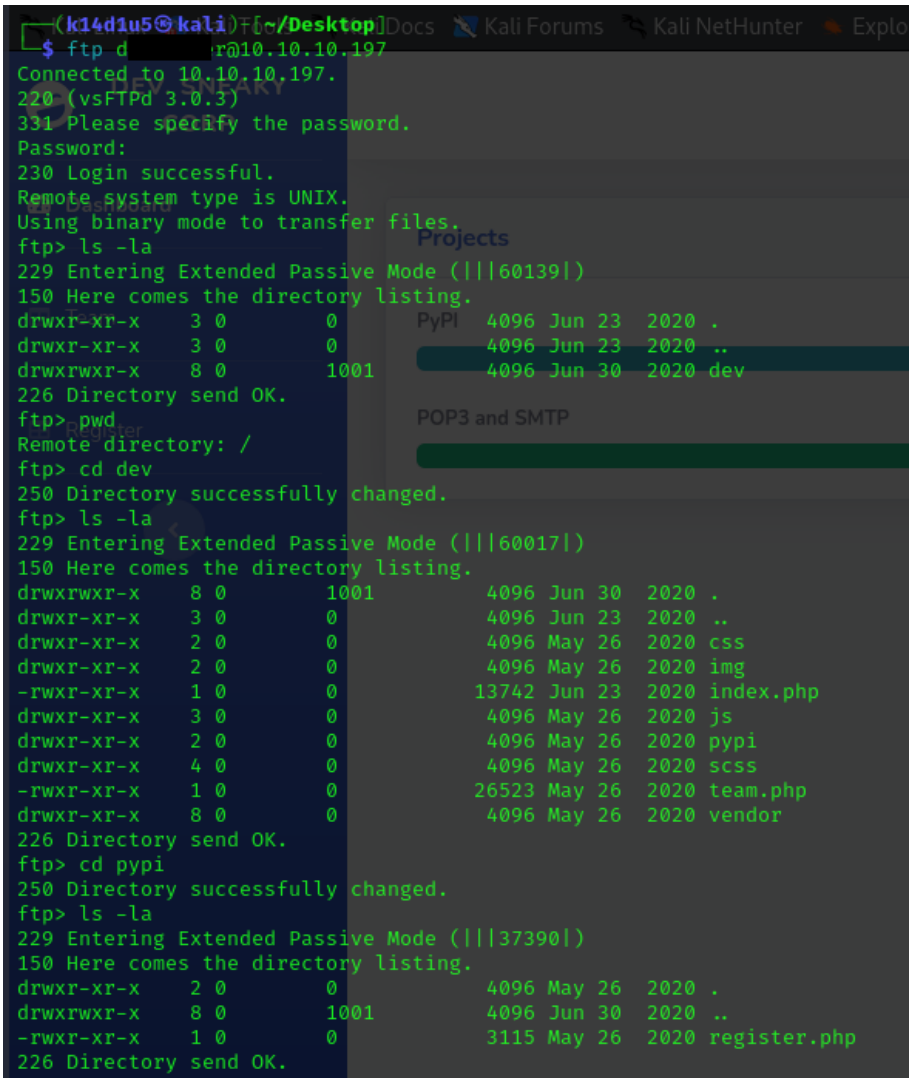

Figure 5 - FTP login successful

The previous figure showed that the FTP root is the dev subdomain I already found. Luckily, I was able to upload a PHP reverse shell in the dev folder, and I invoked it browsing the relating PHP web page:

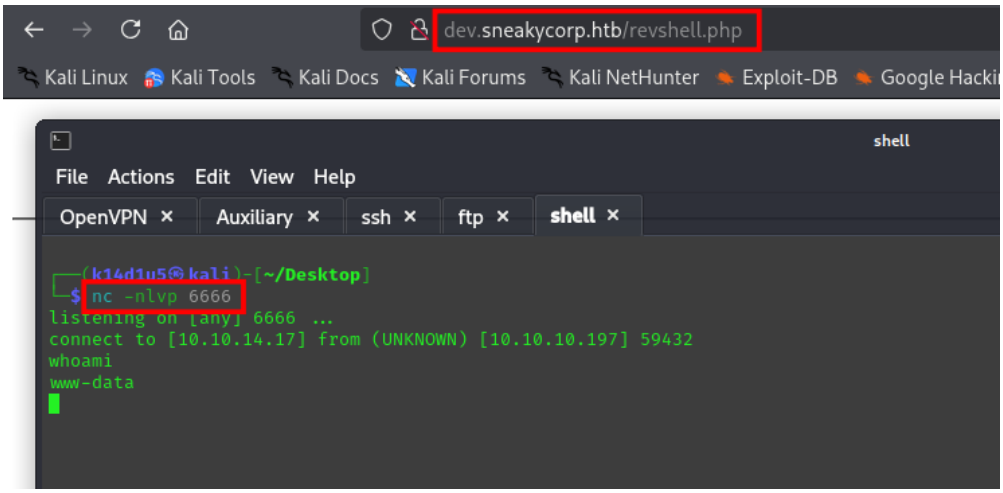

Figure 6 - First shell obtained

Using this shell, I was able to read the $/etc/passwd$ file:

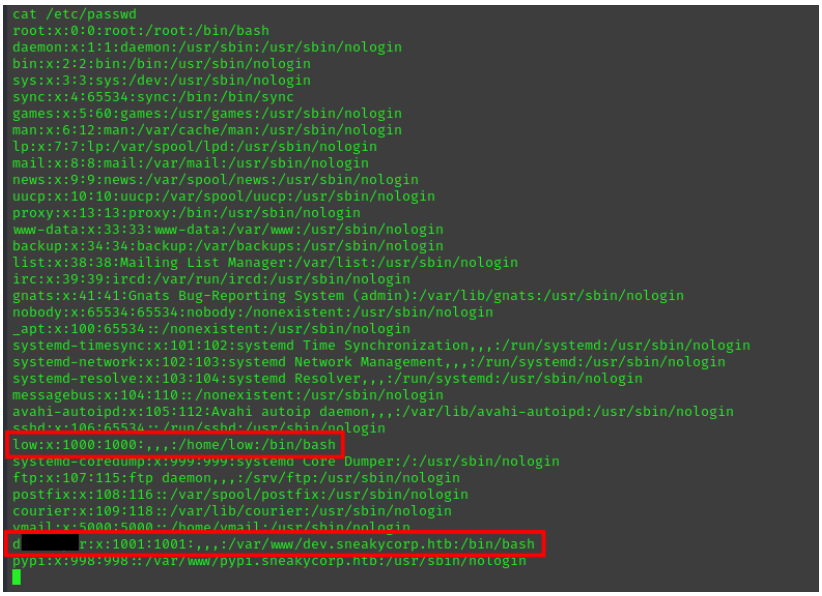

Figure 7 - /etc/passwd file

I found the user which I have found credentials. So, I was able to became him with su command:

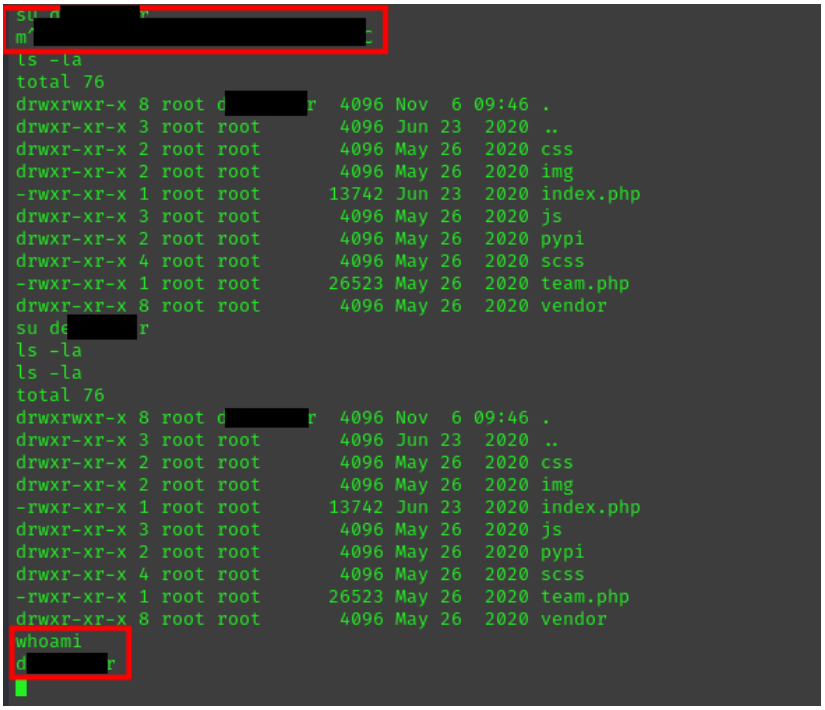

Figure 8 - Switch user

At this point I analyzed the filesystem. In particular, I found a new subdomain, $pypi$. One of the files in its folder contained a new set of credentials:

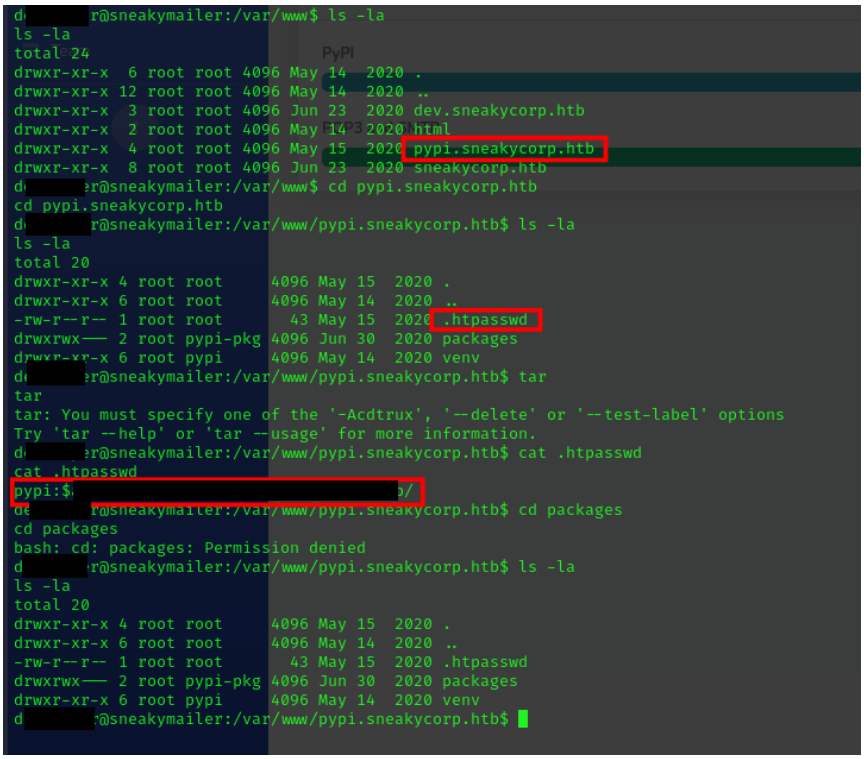

Figure 9 – Second new credentials found

I tried to crack this new password using $hashcat$ on my attacker machine and I was successful:

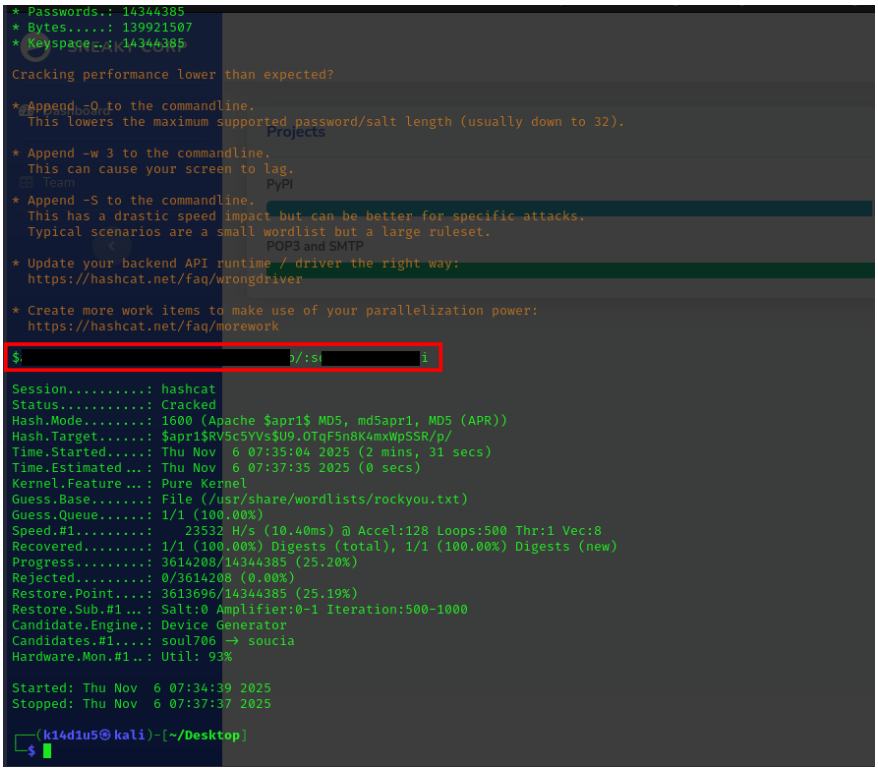

Figure 10 - Password cracked

I looked other information about this subdomain and I found out that it run on port 8080 from the $sites - enabled$ nginx file. I remembered one of e-mails I found and I understood I needed to upload a malicious python module. I looked on the Internet to learn how to upload a python module in PyPI. After a while, I created on my attacker machine a $.pypirc$ file and a python module. In particular, I needed just the

setup.py file for the python module. In this file I inserted the malicious code to open a new reverse shell. At this point, the following is the command I needed to properly run the exploit:

Figure 11 - PyPI exploitation

Of course, I opened even the listener and, in few seconds, I obtained a new shell I used to retrieve the user flag, as shown in the following picture:

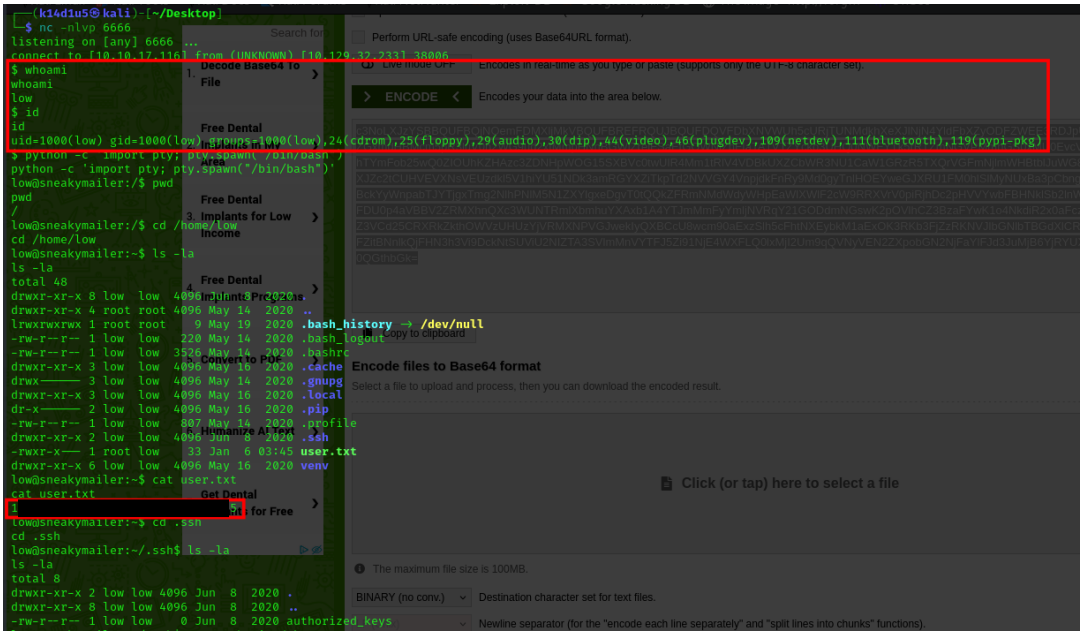

Figure 12 - Shell as low user and user flag

Privilege escalation

After all I already did, I needed to escalate my privileges. To do so, I checked if the *low* user was able to run some application as sudo:

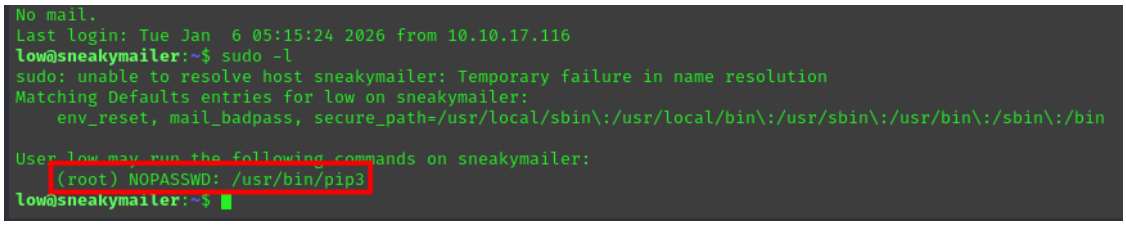

Figure 13 - Sudoers

I was able to exploit $pip3$ command using an exploit found on GTFObins and I retrieved the root flag:

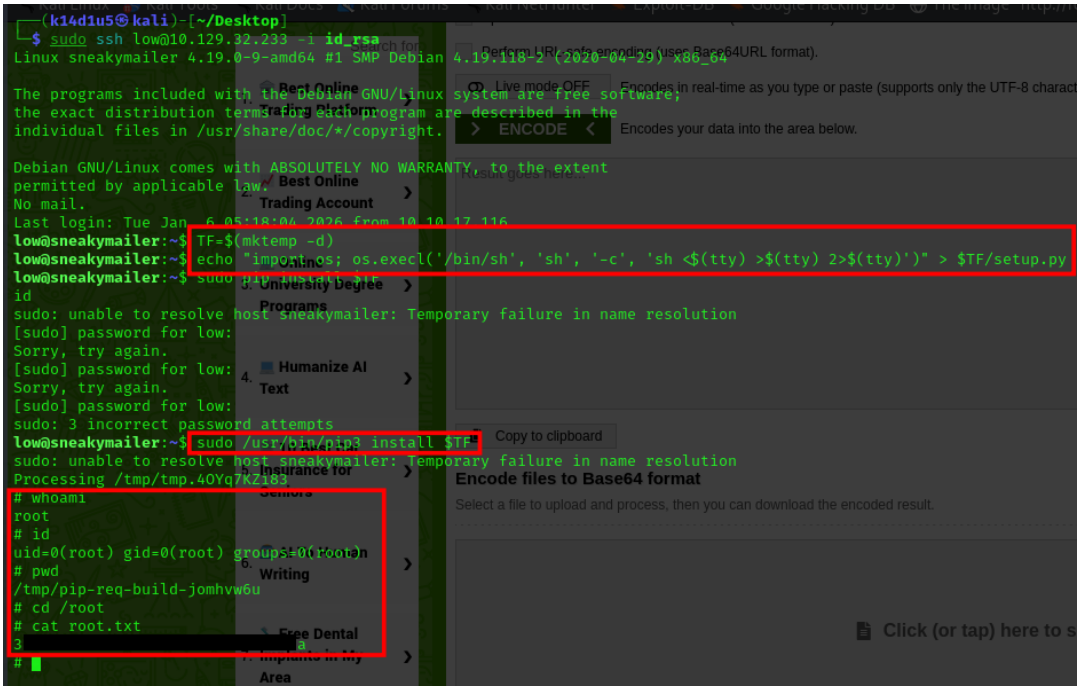

Figure 14 - Privilege escalation and root flag

Personal comments

This box was totally crazy. I need to perform a lot of things to gain the user flag; on the other hand, luckily, the root flag was a piece of cake. In my opinion, insert a phishing task in a box o a platform as HackTheBox or similar is completely no-sense because there is not any real user. Also, the list of e-mail addresses found on the web site is quite long, at least could provide a shorter one. On the other hand, it could be a very realistic scenario. I learnt a lot about python modules and IMAP service during the resolution of this box. I never thought that the $setup.py$ file could contain true code, and honestly it is obvious. It was very important open my mind on this topic. At the end of the day, I consider this box very interesting and let me to learn a lot. I evaluated it as Medium on the platform, but I really enjoyed it at the end of the day.

Appendix A – Uploading python packages

This exploitation was very interesting to me and I learnt a lot about python packages. I would like to go a little bit deeper about this exploit for this box. At first glance, I thought that the $setup.py$ code was executed by client and, if an exception was raised, by server. But when I thought about what it was really happening, I understood I was wrong.

First of all, I said I found two e-mails in the sent items folder, but I took a screenshot of just one. Also, I said what was written in the other one. I don't have a screenshot of it, but I copied the body and it was the following:

```
Hello low
Your current task is to install, test and then erase every python module you
find in our PyPI service, let me know if you have any inconvenience.
```

Also, I found a service run by *low* user:

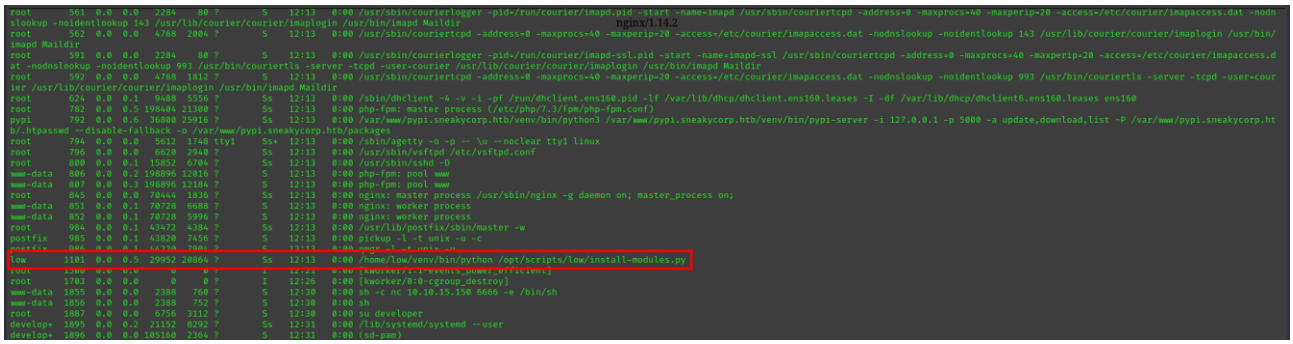

Figure 15 - Appendix A: service run by "low" user

This was a python script named $install - modules.py$. This simply means that when I uploaded a python module, *low* installed it, tested it and erased it. So, if the module contains malicious code, it will be run by *low*. The server itself, or its service user, never execute it. This is the reason I was able to obtain the user shell as *low* user.

References

1. IMAP authentication: https://www.atmail.com/blog/imap-101-manual-imap-sessions/, https://stackoverflow.com/questions/7192130/how-to-connect-imap-using-authenticate-plain-correctly;
2. IMAP commands: https://www.atmail.com/blog/imap-commands/;
3. Uploading python packages: https://www.linode.com/docs/guides/how-to-create-a-private-python-package-repository/, https://blog.jonasneubert.com/2017/09/13/publishing-your-first-pypi-package/, https://github.com/pypiserver/pypiserver?tab=readme-ov-file#upload-with-setuptools, https://github.com/joelbarmettlerUZH/PyPi_Guide, https://docs.python.org/3.11/distutils/setupscript.html.