Forge 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 22 and 80. Also, port 21 is filtered, so I was not able to contact FTP service from my Kali machine, but it is probably running. Services running are SSH (22) and a web application running on port 80. Lastly, nMap identified Linux as operative system.

Initial foothold

Analyzing the nMap scan results, I can only browse the web application on port 80. It is an image gallery. I was able to use an upload image functionality, but the image I uploaded was not visible in the gallery. I was able to access to them using the link the application provided me after the upload. When I tried to upload a file using an Internet URL or an URL to a netcat connection, I found out that this functionality was implemented in Python, as shown in the following picture:

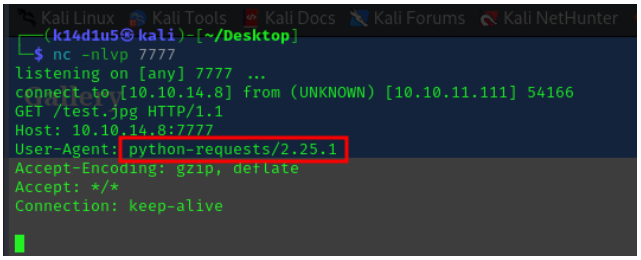

Figure 2 - Python implementation

Also, I run $ffuf$ tool to find some hidden subdomain. Luckily, I found out the $admin.forge.htb$ subdomain:

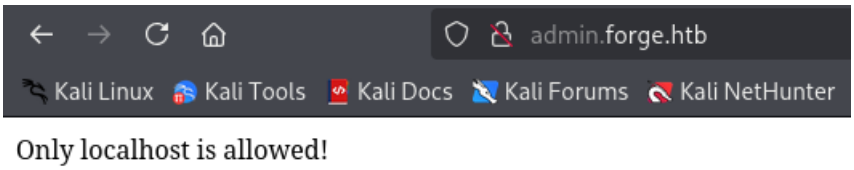

Figure 3 - Admin subdomain

However, I can access to it only by localhost. So, I used the URL subdomain as URL to upload images and it was accepted (implementing an SSRF attack). I was able to read the source code performing a curl request to the link normally used to access to the image uploaded. In its source code, I found a new path, $/announcements$, and I tried to navigate it as I just did. On its source code, I found a couple of credentials:

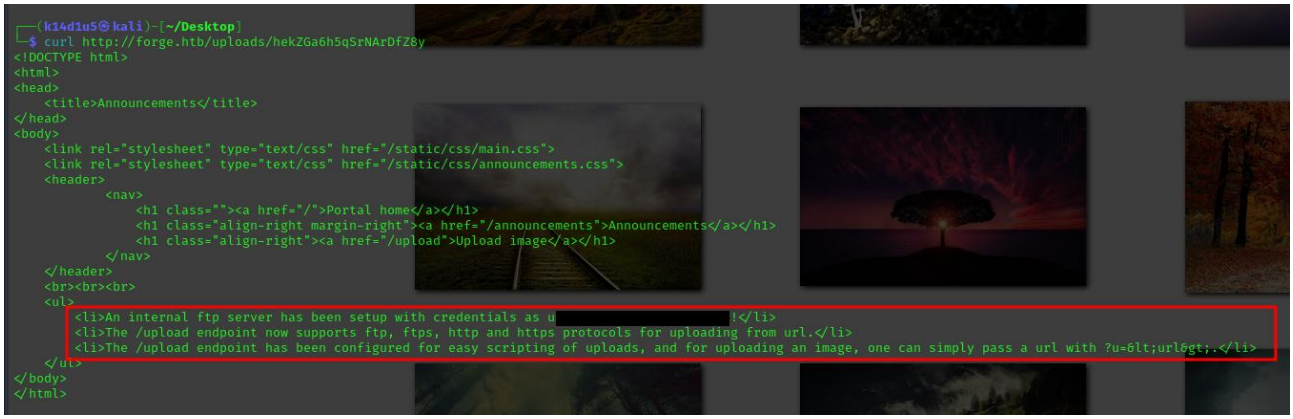

Figure 4 - Credentials fund

User flag

As said in the source code, the credentials found are useful to access to the FTP server. Also, the source code provided me the information about I was able to access to the FTP server using that subdomain, path $/upload$ and u parameter. Therefore, in the same way I did before, I exploited the SSRF vulnerability in the upload functionality on the main domain to access to the FTP server. On its root, I found out the user flag:

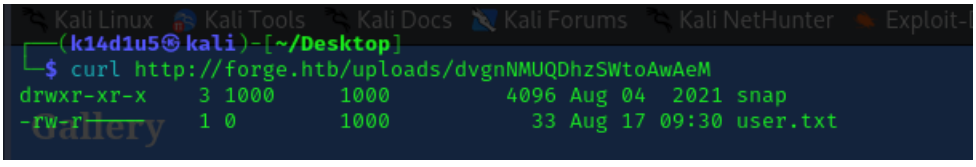

Figure 5 - FTP server root folder

I retrieved the flag using this way, but I forgot to take the screenshot of it.

Privilege escalation

As shown in the previous screenshot, I found a $snap$ folder in the FTP server root. I browsed it, but I didn't find anything useful. Looking for some interesting information on the Internet, I found out that folder is located in the user home directory for default. Therefore, I tried to access to the user's SSH keys. Luckily, I was able to read user private key, locate in $.ssh/id\_rsa$ file in the FTP root server folder, and I used it to log in via SSH on the target:

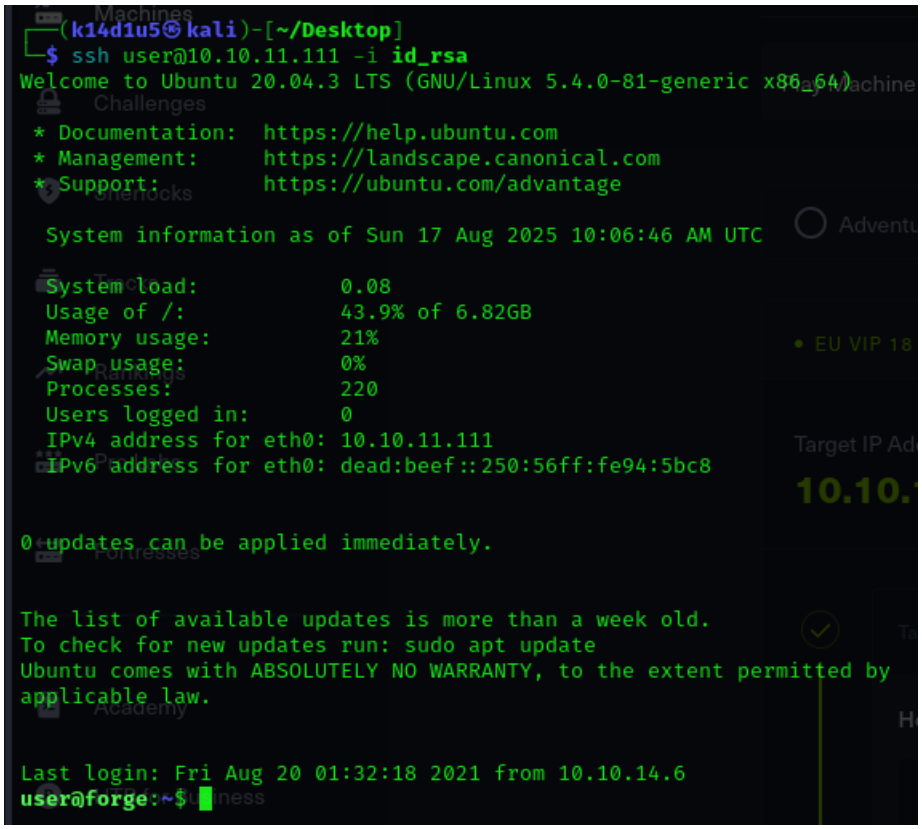

Figure 6 - SSH login as user

Using this connection, I started to navigate the file system and look for a way to escalate my privileges. In particular, I found out that my user was able to run a specific command as sudo without using password:

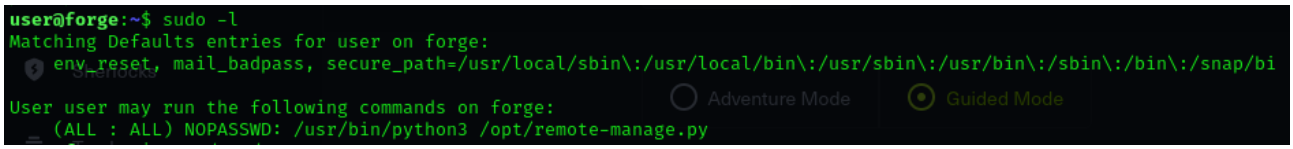

Figure 7 - Sudoers configuration

I read the script and I found a password to use the service it run hardcoded in the source code (I forgot the script screenshot). Also, this script allowed to choice which operation to perform using a menu option and providing the relative number to activate the code I wanted to run. However, the script handle error just using the exception and the pdb module $post\_mortem$ function. So, when I forced an error inserting a character instead of a number as my choice, I was able to get and access to the pdb python debugger:

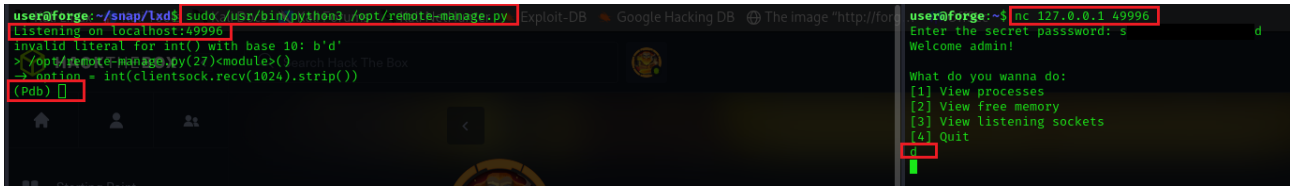

Figure 8 - Python debugger opened

Exploiting the python debugger, I was able to execute Python code and open a new shell as root (in fact, I run the script as root) and I was able to retrieve the root flag:

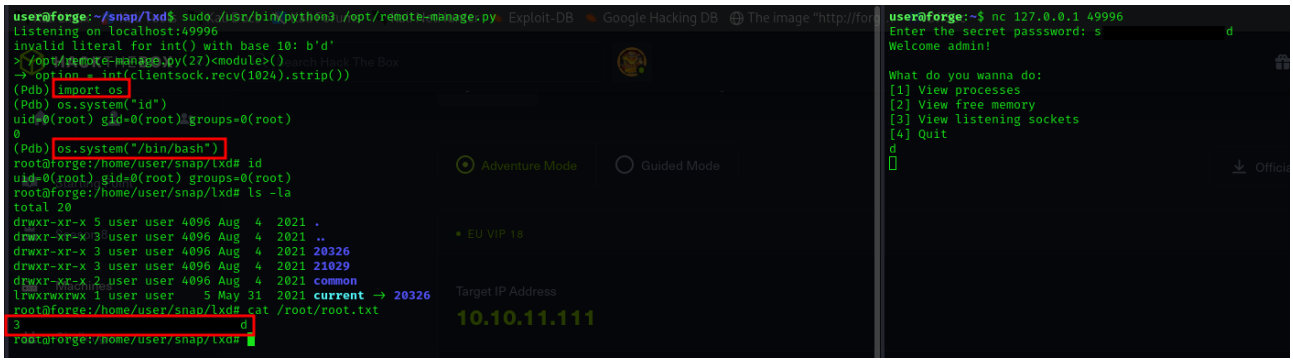

Figure 9 - Privilege escalation and root flag

Personal comments

I found this box very funny and interesting. I learnt that it is important to read documentation of any function I don't know yet and go deeper to understand well how it works. Anything can be useful and could be exploited. I overall evaluate it as a Medium box, a very good one.

References

1. Post mortem function: https://docs.python.org/3/library/pdb.html#pdb.post_mortem;
2. PDB module: https://ironpython-test.readthedocs.io/en/latest/library/pdb.html.