I. General Information:

HW1 – RAW SOCKET

Le Tien Dat - 1712328

II. Documents are included

1. Folder np-hw: 1712328.c + Makefile
   - Makefile

```
src = $(wildcard *.c)
obj = $(src:.c=.o)

CC=/usr/bin/gcc
CFLAGS=-g -Wall -pthread -Wall -Wextra -Wshadow -Wpointer-arith -Wwrite-strings

1712328: $(obj)
    $(CC) -o $@ $^ $(CFLAGS)

.PHONY: clean
clean:
    rm -f $(obj) 1712328
```

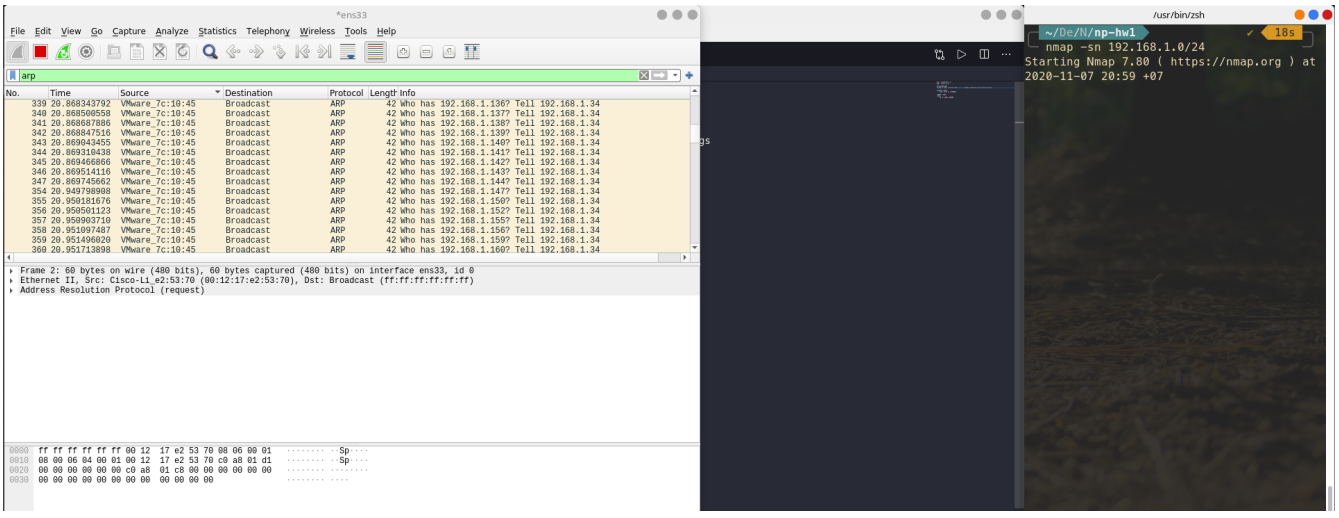
2. Report

III. Doing part:

1. Explain

We will clone to the command: nmap -sn

- After checking the wireshark, I see nmap send the arp packet to broadcast



- First of all, need to **edit** *ens33* to your network card (*like eth1, Etc. ifconfig to know it*)

```
103     char* if_name = (char *) malloc (40 * sizeof (char));
104     memset(if_name,0, 40 * sizeof (char));
105     strcpy (if_name, "ens33");
```

- Processing on the command, we have ./1712328 28.32.32.0/24. Let's run it with Sudo privileges.

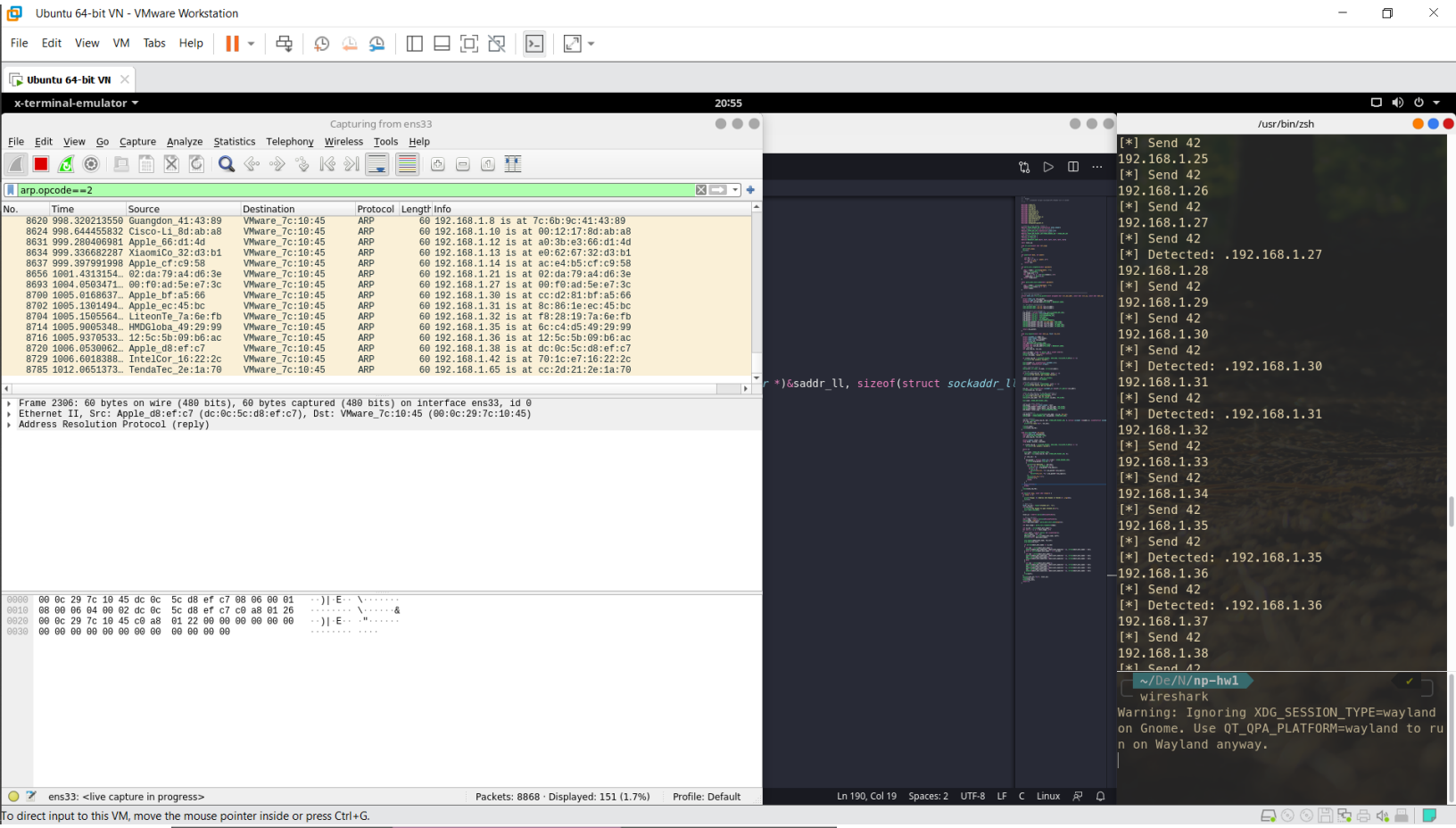
- Then to process it we get the range host and network path (*pharse functions*)
- We use the string processing methods in C (*pharse functions*)
- I am putting in *for* is how to handle it. (*in main funciton*)
- The proccess of my program base on step below:
    - Input with argrument
    - Pharse it to range dest ip and number of range host
    - For 1 -> host range (maximum is 255) (I don't have enough time for solve it)
        - Send the arp request
        - Receive the arp reply

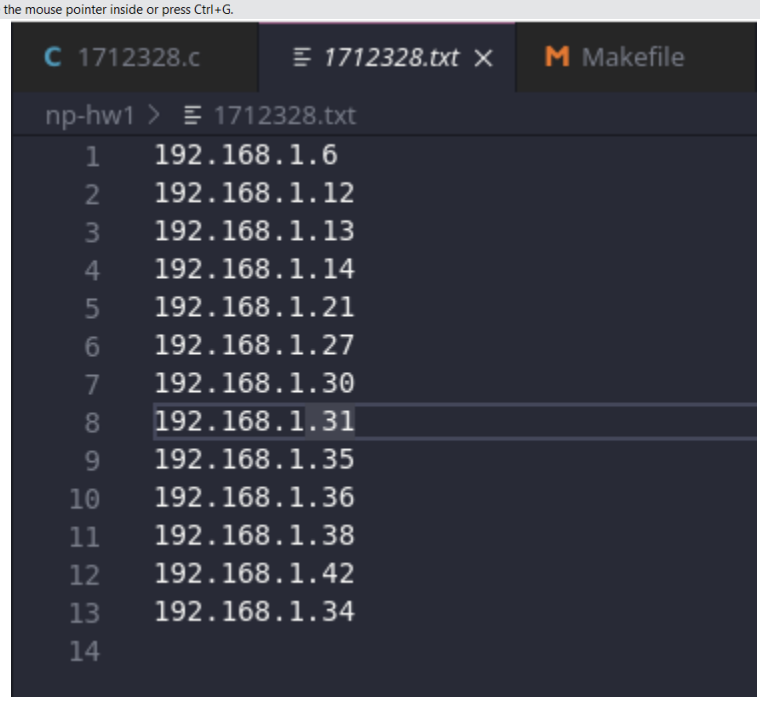
```
for (int i = 1; i <= host_range; i++)
{
  char* buff = (char*) malloc (4* sizeof(char));
  sprintf(buff, "%d", i);
  dest_host_name = strcat(dest_host_name, buff);
  printf("%s\n", dest_host_name);

  arp_request(dest_host_name, res_txt);
  arp_reply(res_txt);
```

        - If arp.opcode == 2 put this IP in my file
        - If the return of recvfrom = 0. Drop this IP and handle next ip

- The result


Ubuntu 64-bit VN - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit VN
x-terminal-emulator
20:55
Capturing from ens33
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp.opcode==2
No. Time Source Destination Protocol Length Info
8620 998.320213550 Guangdon_41:43:89 VMware_7c:10:45 ARP 60 192.168.1.8 is at 7c:6b:9c:41:43:89
8624 998.644455832 Cisco-Li_8d:ab:a8 VMware_7c:10:45 ARP 60 192.168.1.10 is at 00:12:17:8d:ab:a8
8631 999.280406981 Apple_66:d1:4d VMware_7c:10:45 ARP 60 192.168.1.12 is at a0:3b:e3:66:d1:4d
8634 999.336682287 XiaomiCo_32:d3:b1 VMware_7c:10:45 ARP 60 192.168.1.13 is at e0:62:67:32:d3:b1
8637 999.397991998 Apple_cf:c9:58 VMware_7c:10:45 ARP 60 192.168.1.14 is at ac:e4:b5:cf:c9:58
8656 1001.4313154… 02:da:79:a4:d6:3e VMware_7c:10:45 ARP 60 192.168.1.21 is at 02:da:79:a4:d6:3e
8693 1004.0503471… 00:f0:ad:5e:e7:3c VMware_7c:10:45 ARP 60 192.168.1.27 is at 00:f0:ad:5e:e7:3c
8700 1005.0168637… Apple_bf:a5:66 VMware_7c:10:45 ARP 60 192.168.1.30 is at cc:d2:81:bf:a5:66
8702 1005.1301494… Apple_ec:45:bc VMware_7c:10:45 ARP 60 192.168.1.31 is at 8c:86:1e:ec:45:bc
8704 1005.1505564… LiteonTe_7a:6e:fb VMware_7c:10:45 ARP 60 192.168.1.32 is at f8:28:19:7a:6e:fb
8714 1005.9005348… HMDGloba_49:29:99 VMware_7c:10:45 ARP 60 192.168.1.35 is at 6c:c4:d5:49:29:99
8716 1005.9370533… 12:5c:5b:09:b6:ac VMware_7c:10:45 ARP 60 192.168.1.36 is at 12:5c:5b:09:b6:ac
8720 1006.0530062… Apple_d8:ef:c7 VMware_7c:10:45 ARP 60 192.168.1.38 is at dc:0c:5c:d8:ef:c7
8729 1006.6018388… IntelCor_16:22:2c VMware_7c:10:45 ARP 60 192.168.1.42 is at 70:1c:e7:16:22:2c
8785 1012.0651373… TendaTec_2e:1a:70 VMware_7c:10:45 ARP 60 192.168.1.65 is at cc:2d:21:2e:1a:70
Frame 2306: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens33, id 0
Ethernet II, Src: Apple_d8:ef:c7 (dc:0c:5c:d8:ef:c7), Dst: VMware_7c:10:45 (00:0c:29:7c:10:45)
Address Resolution Protocol (reply)
0000 00 0c 29 7c 10 45 dc 0c 5c d8 ef c7 08 06 00 01
0010 08 00 06 04 00 02 dc 0c 5c d8 ef c7 c0 a8 01 26
0020 00 0c 29 7c 10 45 c0 a8 01 22 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00
ens33: <live capture in progress>
Packets: 8868 · Displayed: 151 (1.7%)
Profile: Default
Ln 190, Col 19 Spaces: 2 UTF-8 LF C Linux
/usr/bin/zsh
[*] Send 42
192.168.1.25
[*] Send 42
192.168.1.26
[*] Send 42
192.168.1.27
[*] Send 42
[*] Detected: .192.168.1.27
192.168.1.28
[*] Send 42
192.168.1.29
[*] Send 42
192.168.1.30
[*] Send 42
[*] Detected: .192.168.1.30
192.168.1.31
[*] Send 42
[*] Detected: .192.168.1.31
192.168.1.32
[*] Send 42
192.168.1.33
[*] Send 42
192.168.1.34
[*] Send 42
192.168.1.35
[*] Send 42
[*] Detected: .192.168.1.35
192.168.1.36
[*] Send 42
[*] Detected: .192.168.1.36
192.168.1.37
[*] Send 42
192.168.1.38
~/De/N/np-hw1
wireshark
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.


1712328.c
1712328.txt
Makefile
np-hw1 > 1712328.txt
1 192.168.1.6
2 192.168.1.12
3 192.168.1.13
4 192.168.1.14
5 192.168.1.21
6 192.168.1.27
7 192.168.1.30
8 192.168.1.31
9 192.168.1.35
10 192.168.1.36
11 192.168.1.38
12 192.168.1.42
13 192.168.1.34
14

IV. Conclusion

- My program just handle for more or equal 24 bit of NetID, it mean if you give the input with < /23, it would excute wrong so maximum of for is 255.
- I can solve it but I need more time.
- The different thing between of my program and nmap is nmap send the arp request recently, my program just send the arp req and recv at the time.
- I understand the nmap method with arp packet.

V. Reference

https://www.programmersought.com/article/40053885963/