CSC Project1 Report

1. Snapshots of Task1 and Task2:

First, check the IP of the attacker and the victim. I use VM (cs2021, ubuntu) as attacker and localhost (Windows 10) as victim.

Attacker:

```
cs2021@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.171.132  netmask 255.255.255.0  broadcast 192.168.171.255
        inet6 fe80::6862:2786:d010:5e78  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:90:10:e8  txqueuelen 1000  (Ethernet)
        RX packets 12798  bytes 18525522 (18.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1267  bytes 120388 (120.3 KB)
        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 521  bytes 34887 (34.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 521  bytes 34887 (34.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

cs2021@ubuntu:~$
```

Victim:

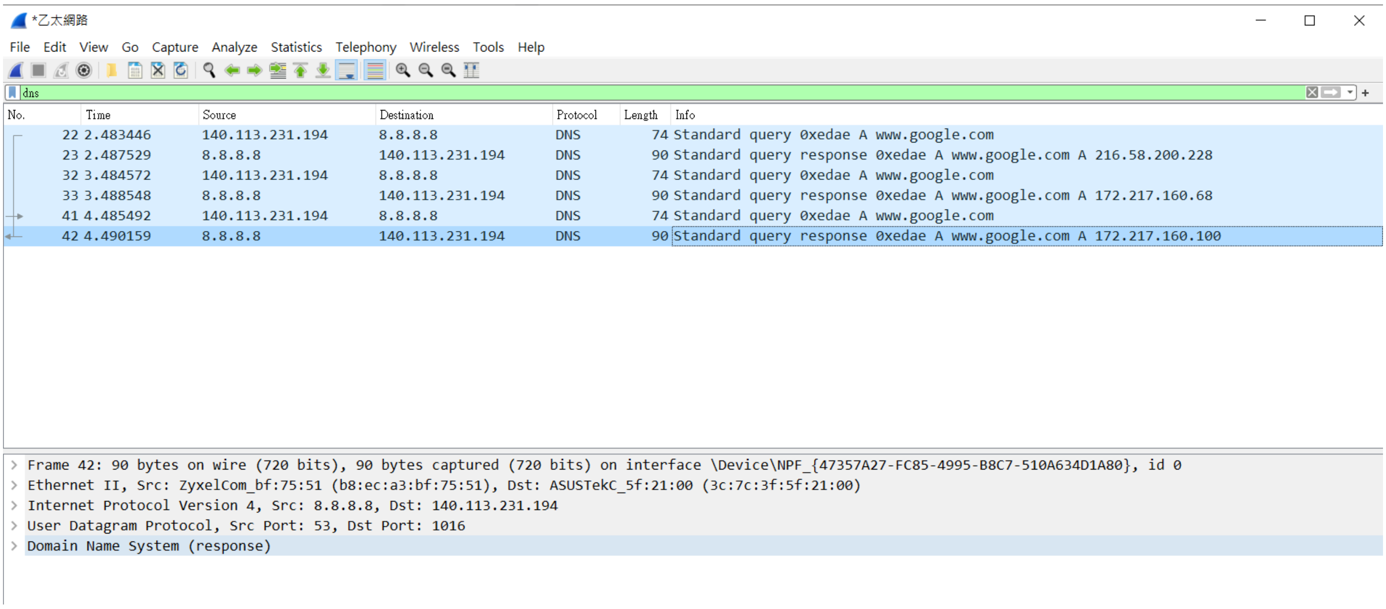
```
C:\Users\user>ipconfig

Windows IP 設定


不明的介面卡 OpenVPN Wintun:

   媒體狀態 . . . . . . . . . . . . . . : 媒體已中斷連線
   連線特定 DNS 尾碼 . . . . . . . . . :

乙太網路卡 乙太網路:

   連線特定 DNS 尾碼 . . . . . . . . . : DOMAIN.local
   連結-本機 IPv6 位址 . . . . . . . . : fe80::681b:c6c0:d6be:7339%9
   IPv4 位址 . . . . . . . . . . . . . : 140.113.231.194
   子網路遮罩 . . . . . . . . . . . . .: 255.255.255.0
   預設閘道 . . . . . . . . . . . . . .: 140.113.231.254
```

Task1:

I run my dns_attack program on cs2021 (need to use sudo) . I choose Google server (8.8.8.8) as DNS server and my local IP (140.113.231.194) as victim. Then it sent 3 DNS queries:

```
cs2021@ubuntu:~/Desktop$ sudo ./dns_attack 140.113.231.194 7 8.8.8.8
set done.
header done.
start sending...
No.0: success.
No.1: success.
No.2: success.
cs2021@ubuntu:~/Desktop$ 
```

Then I captured the packets using Wireshark in victim:



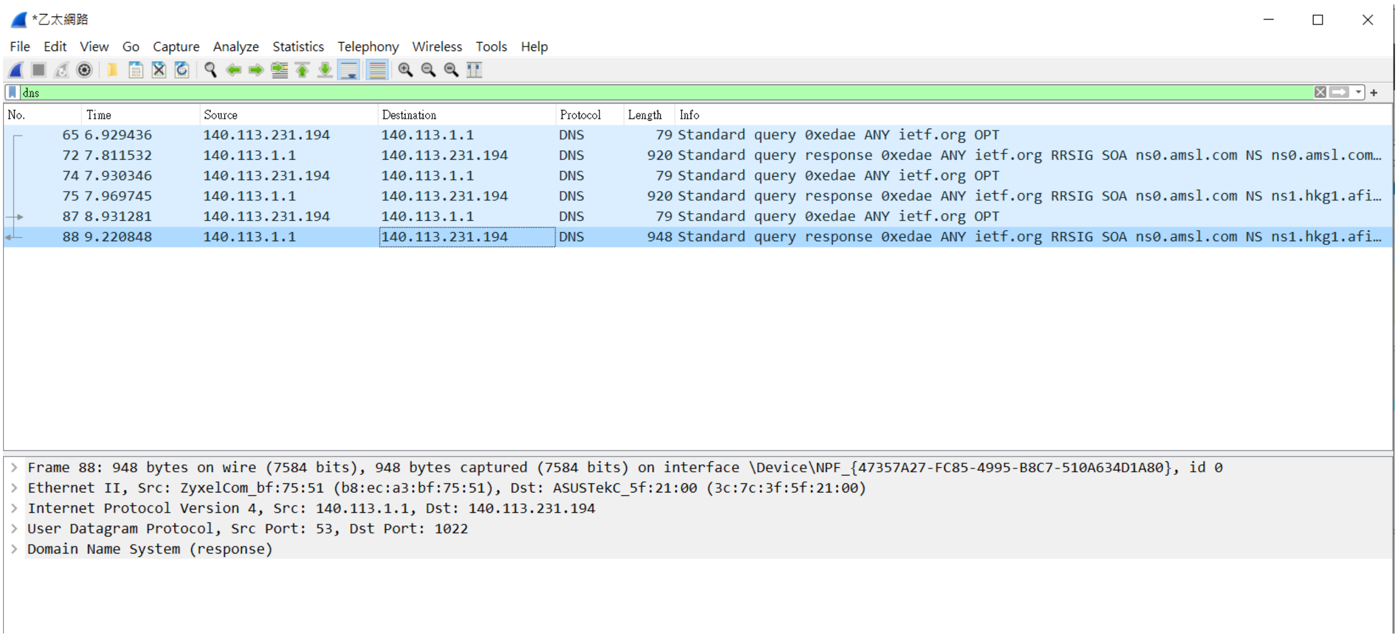
In above picture, we can see that three DNS request packets sent from 140.113.231.194 and each received an DNS response packet from 8.8.8.8, which has met the requirement of Task1.

Task2:

Same as Task1. But this time I choose NCTU DNS server (140.113.1.1) as DNS server and my local IP (140.113.231.194) as victim. It also sent 3 DNS queries:

```
cs2021@ubuntu:~/Music$ sudo ./dns_attack 140.113.231.194 7 140.113.1.1
set done.
header done.
start sending...
No.0: success.
No.1: success.
No.2: success.
cs2021@ubuntu:~/Music$ 
```

Then I captured the packets using Wireshark in victim:

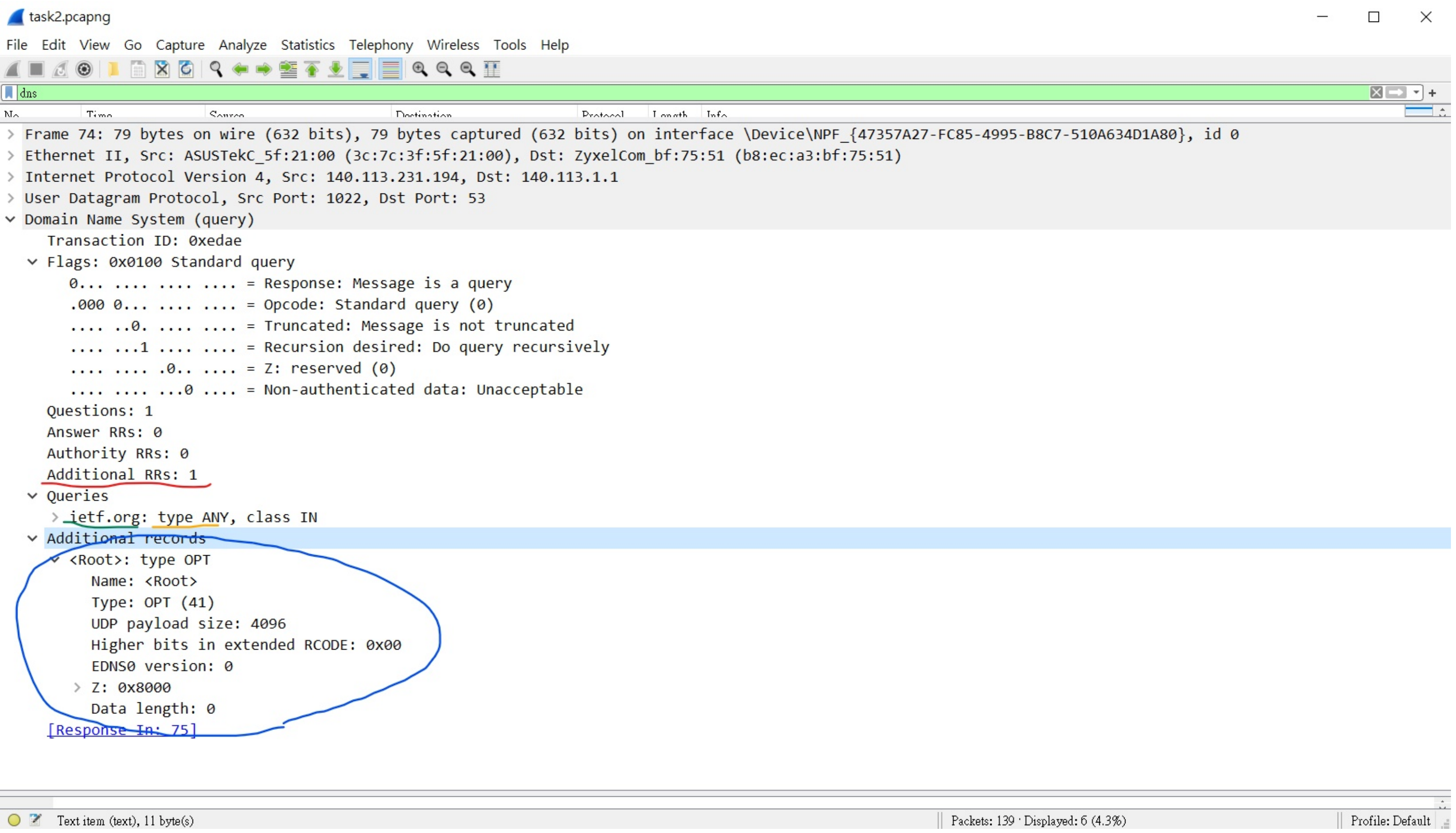


We can see that three DNS request packets sent from 140.113.231.194 and each received an DNS response packet from 140.113.1.1. The request packets has size 79 bytes and the response packets has size 920 bytes to 948 bytes. The amplification ratio has reached approximately 11.65, which has met the requirement of Task2.

My student number is 0716206, that is 0xAEDAE in hex. So my query ID is 0xedae.

2. How I amplify the DNS response:

DNS reflection attack and DNS amplification attack are both using raw socket to spoof the source IP address. The only difference between these two are they're DNS query message. In reflection attack, amplification rate R is usually 1 to 1.5; however, in amplification attack, R is usually greater then 3. Thus, I amplified the DNS response by simply modifying the DNS protocol message from a normal DNS request. Shown in following picture:



i. set the “Additional RR” bytes to 0x00 0x01 (red underline)

ii. set the “QNAME” to ietf.org (green underline)

iii. set the “QTYPE” to ANY (yellow underline)

iv. add the “Additional records” (blue circle)

In order to get a large DNS response, I set the DNS record type to ANY. This will returns all records of all types known to the server. Then I set the “Additional RR” to 1, and add a list of bytes to enable the EDNS0 , which allows DNS clients to expand and advertise up to 4096 bytes of UDP packets for certain DNS parameters.

3. A solution that defend against the DoS attack based on the DNS reflection:

From server’s point of view, we can defend against the attack by forcing the DNS clients to prove that their IP address is not spoofed. We can use some anti-spoofing techniques such as forcing a TCP transmission or forcing a retransmission.