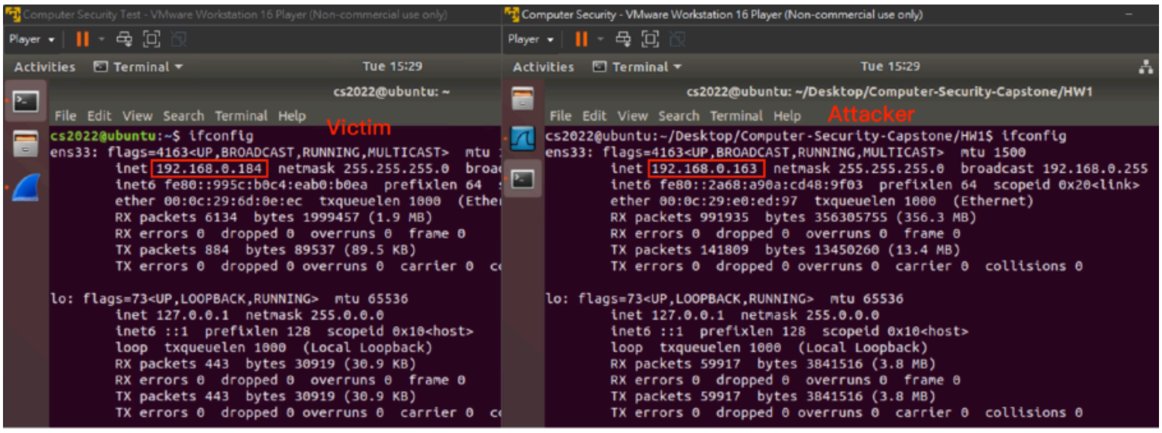
Project 1: DNS Reflection and Amplification Attacks

1. Experiment of DNS Reflection & Amplification Attacks


Computer Security Test - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Tue 15:29
cs2022@ubuntu: ~
File Edit View Search Terminal Help
Victim
Computer Security - VMware Workstation 16 Player (Non-commercial use only)
cs2022@ubuntu: ~/Desktop/Computer-Security-Capstone/HW1
Attacker

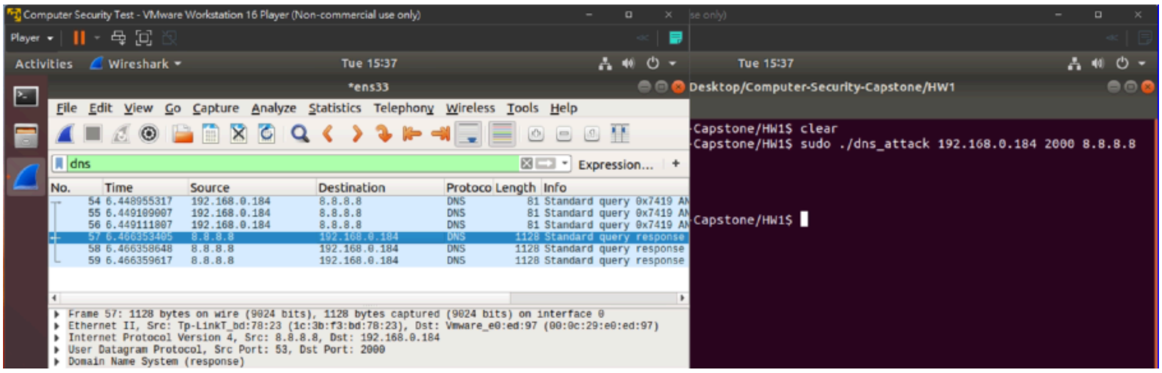
```
cs2022@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 
        inet 192.168.0.184  netmask 255.255.255.0  broa
        inet6 fe80::995c:b0c4:eab0:b0ea  prefixlen 64  
        ether 00:0c:29:6d:0e:ec  txqueuelen 1000  (Ether
        RX packets 6134  bytes 1999457 (1.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 884  bytes 89537 (89.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  c

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 443  bytes 30919 (30.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 443  bytes 30919 (30.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  c
```

```
cs2022@ubuntu:~/Desktop/Computer-Security-Capstone/HW1$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.163  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::2a68:a90a:cd48:9f03  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e0:ed:97  txqueuelen 1000  (Ethernet)
        RX packets 991935  bytes 356305755 (356.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 141809  bytes 13450260 (13.4 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 59917  bytes 3841516 (3.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59917  bytes 3841516 (3.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

I created two same virtual machines provided by the class and make one be the attacker and the other be victim of this experiment.


Computer Security Test - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Wireshark
Tue 15:37
*ens33
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
Expression...
Desktop/Computer-Security-Capstone/HW1

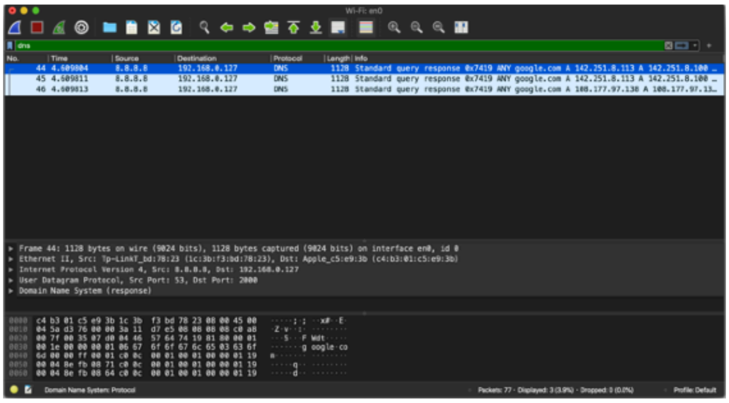
```
No.  Time         Source         Destination    Protoco Length Info
  54 6.448955317  192.168.0.184  8.8.8.8        DNS       81 Standard query 0x7419 AN
  55 6.449109007  192.168.0.184  8.8.8.8        DNS       81 Standard query 0x7419 AN
  56 6.449111807  192.168.0.184  8.8.8.8        DNS       81 Standard query 0x7419 AN
  57 6.466353405  8.8.8.8        192.168.0.184  DNS     1128 Standard query response
  58 6.466358648  8.8.8.8        192.168.0.184  DNS     1128 Standard query response
  59 6.466359617  8.8.8.8        192.168.0.184  DNS     1128 Standard query response

▸ Frame 57: 1128 bytes on wire (9024 bits), 1128 bytes captured (9024 bits) on interface 0
▸ Ethernet II, Src: Tp-LinkT_bd:78:23 (1c:3b:f3:bd:78:23), Dst: Vmware_e0:ed:97 (00:0c:29:e0:ed:97)
▸ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.0.184
▸ User Datagram Protocol, Src Port: 53, Dst Port: 2000
▸ Domain Name System (response)
```

```
Capstone/HW1$ clear
Capstone/HW1$ sudo ./dns_attack 192.168.0.184 2000 8.8.8.8


Capstone/HW1$ 
```

I ran the program on attacker (192.168.0.163) and monitor the internet packets from victim (192.168.0.184). As the result, we could discover that the victim(.184) received the DNS response from the internet which query packet send by the attacker(.163). Another things that because two VMs are bridge to same interfaces, so the Wireshark could also track the DNS query send by the attacker. As the result, I try to attack my laptop which is also under the same router as the VMs.


Wi-Fi: en0
dns
Domain Name System: Protocol
Packets: 77 · Displayed: 3 (3.9%) · Dropped: 0 (0.0%)
Profile: Default

```
No.  |Time      |Source   |Destination    |Protocol |Length|Info
  44 4.609804   8.8.8.8   192.168.0.127   DNS       1128 Standard query response 0x7419 ANY google.com A 142.251.8.113 A 142.251.8.100 …
  45 4.609811   8.8.8.8   192.168.0.127   DNS       1128 Standard query response 0x7419 ANY google.com A 142.251.8.113 A 142.251.8.100 …
  46 4.609813   8.8.8.8   192.168.0.127   DNS       1128 Standard query response 0x7419 ANY google.com A 108.177.97.138 A 108.177.97.13…

▸ Frame 44: 1128 bytes on wire (9024 bits), 1128 bytes captured (9024 bits) on interface en0, id 0
▸ Ethernet II, Src: Tp-LinkT_bd:78:23 (1c:3b:f3:bd:78:23), Dst: Apple_c5:e9:3b (c4:b3:01:c5:e9:3b)
▸ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.0.127
▸ User Datagram Protocol, Src Port: 53, Dst Port: 2000
▸ Domain Name System (response)

0000  c4 b3 01 c5 e9 3b 1c 3b  f3 bd 78 23 08 00 45 00   ·····;·;  ··x#··E·
0010  04 5a d3 76 00 00 3a 11  d7 e5 08 08 08 08 c0 a8   ·Z·v··:·  ········
0020  00 7f 00 35 07 d0 04 46  57 64 74 19 81 80 00 01   ···5···F Wdt·····
0030  00 1e 00 00 00 01 06 67  6f 6f 67 6c 65 03 63 6f   ·······g oogle·co
0040  6d 00 00 ff 00 01 c0 0c  00 01 00 01 00 00 01 19   m······· ········
0050  00 04 8e fb 08 71 c0 0c  00 01 00 01 00 00 01 19   ·····q·· ········
0060  00 04 8e fb 08 64 c0 0c  00 01 00 01 00 00 01 19   ·····d·· ········
```

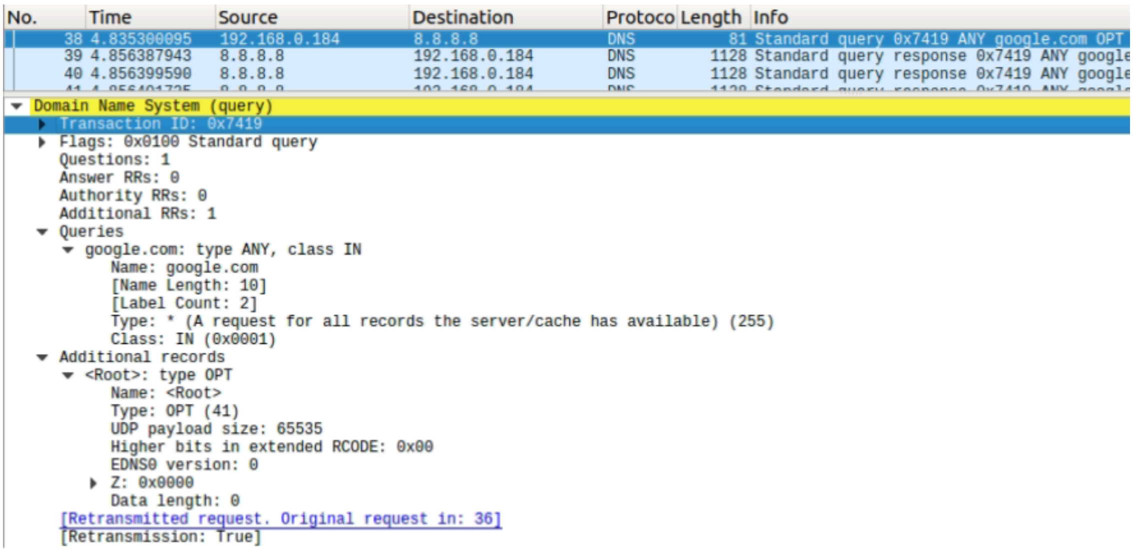
The result shows that my laptop received the DNS response.

(Using port 2000 for easily parsing to Wireshark. Experiments works for port 7 as well.)

One DNS query packet size (Sq) = 81 bytes
One DNS response packet size (Sr) = 1128 bytes

==> Amplification ratio: R = Sr / Sq = 1128 / 81 = 13.92

2. DNS Amplification Attack



```
No.   Time          Source          Destination     Protoco Length Info
   38 4.835300095   192.168.0.184   8.8.8.8         DNS         81 Standard query 0x7419 ANY google.com OPT
   39 4.856387943   8.8.8.8         192.168.0.184   DNS       1128 Standard query response 0x7419 ANY google
   40 4.856399590   8.8.8.8         192.168.0.184   DNS       1128 Standard query response 0x7419 ANY google
▾ Domain Name System (query)
  ▸ Transaction ID: 0x7419
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▾ google.com: type ANY, class IN
        Name: google.com
        [Name Length: 10]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (255)
        Class: IN (0x0001)
  ▾ Additional records
    ▾ <Root>: type OPT
        Name: <Root>
        Type: OPT (41)
        UDP payload size: 65535
        Higher bits in extended RCODE: 0x00
        EDNS0 version: 0
      ▸ Z: 0x0000
        Data length: 0
    [Retransmitted request. Original request in: 36]
    [Retransmission: True]
```

The upper image shows the DNS query packet the attacker send to the DNS server. DNS record save a lot of types of records (A, AAAA, TXT, MX….). The **‘TYPE’** tag in the query packet, could specify which type of record you want. In amplification attack, set the tag to “ANY” which cause the DNS server respond all the record it has. This could maximum the packet size.

The other Tag is **‘UDP payload size’** in additional records section. This indicate that the number of octets of the largest UDP payload that can be reassembled and delivered in the requestor's network stack. The maximum suggest size is between 1280 and 1410 for IP over Ethernet, but I set to the maximum size of the field. This makes the DNS server sent the bigger or more records in DNS response packet to the victim.

The modification of this two tags make the amplification ratio larger than 10.

3. Defend against DoS attack based on the DNS reflection

These are the approach to defend against Dos attack based on the DNS reflection:

- **PC Port blocking**

  Block the unneeded port to prevent the possibility of begin influenced by the attack.

- **Router - Packet filter, Rate limiting**

  The router could block the abnormal DNS query packet from identifying the source IP and source port.

- **DNS server - Block ‘ANY’ or IP**

  The DNS server could stop support the ‘ANY’ request (Not a good idea). Also, DNS server could stop respond to the enormous request by the same IP in a period of time.