UE18CS335 – COMPUTER NETWORK SECURITY

Lab – 4 LOCAL DNS ATTACKS LAB

Date: 22/03/2021

By:

Nitish S

PES2201800368

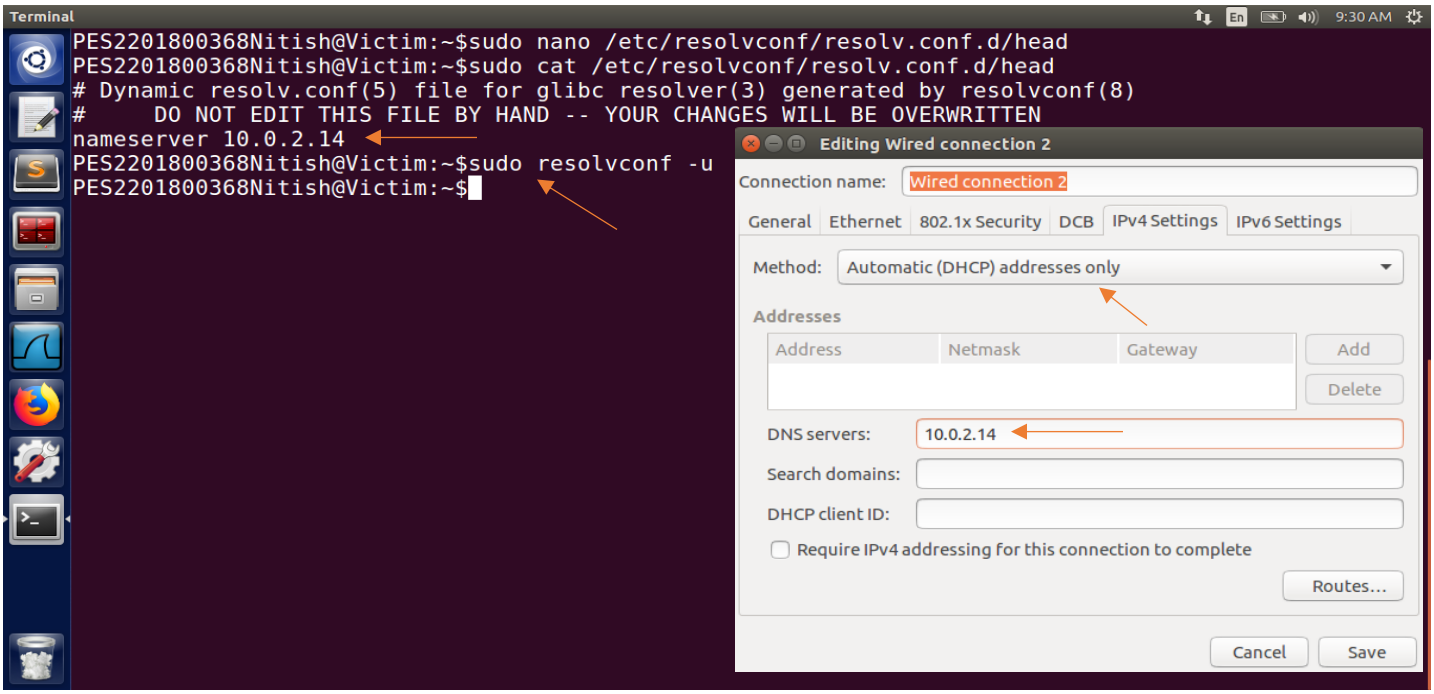
6 'A'

Setup: -

Victim Machine: 10.0.2.4, **Attacker Machine:** 10.0.2.5, **DNS Server Machine:** 10.0.2.14

PART I: SETTING UP A LOCAL DNS SERVER

TASK 1: CONFIGURE THE USER MACHINE: -



SCREENSHOT SHOWING THE CONFIGURATION DONE TO THE USER MACHINE

In order to avoid DHCP from overwriting the 'Edit Wired Connections' file, we manually set the DNS Server in the /etc/resolvconf/resolv.conf.d/head and add the line 'nameserver 10.0.2.14' to it where 10.0.2.14 is the IP address of the DNS Server. To bring the changes to effect, the command 'sudo resolvconf -u' is run.

TASK 2: SET UP A LOCAL DNS SERVER

Step 1: Configure the BIND9 Server

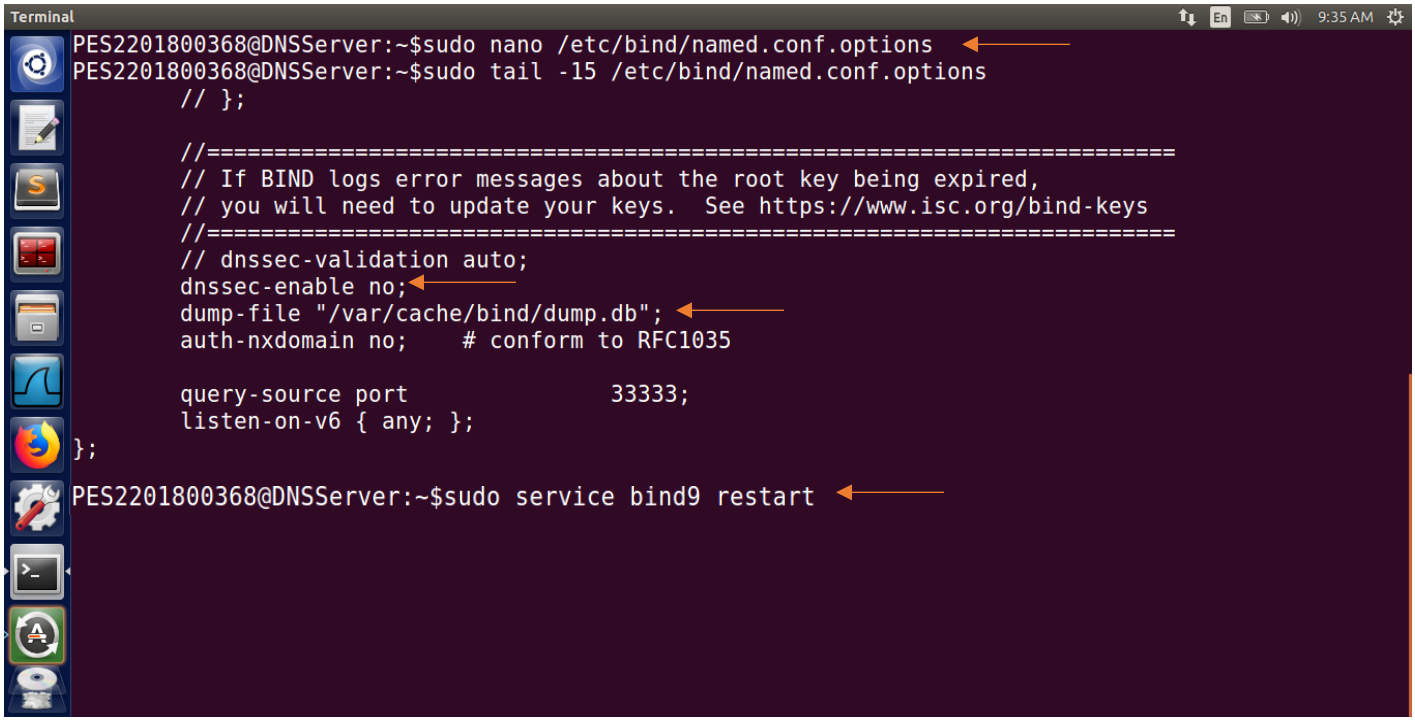
BIND9 gets its configuration from a file called /etc/bind/named.conf. This file is the primary configuration file, and it usually contains several "include" entries. One of the included files is called /etc/bind/named.conf.options. This is where we typically set up the configuration options.

Step 2: Turnoff DNSSEC

We turn the protection against spoofing in DNS server off. This is done by modifying the named.conf.options file. We comment out the dnssec-validation entry, and add a dnssec-enable entry.

Step 3: Start DNS server

SCREENSHOT SHOWING THE ABOVE THREE STEPS:



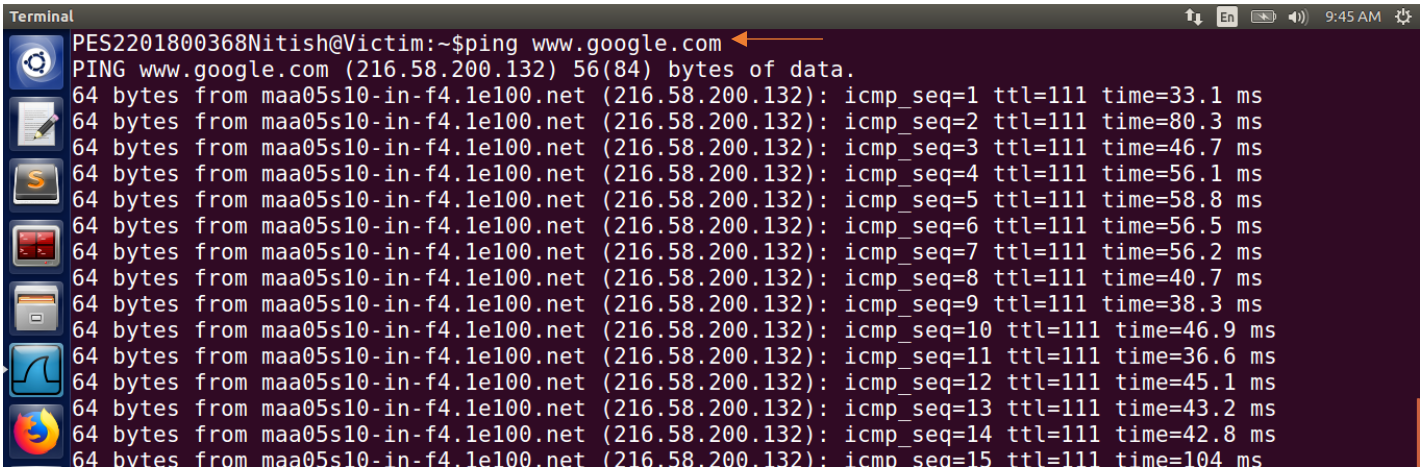
SCREENSHOT SHOWING THE CONFIGURATIONS MADE TO THE LOCAL DNS SERVER

Step 4: Use the DNS server

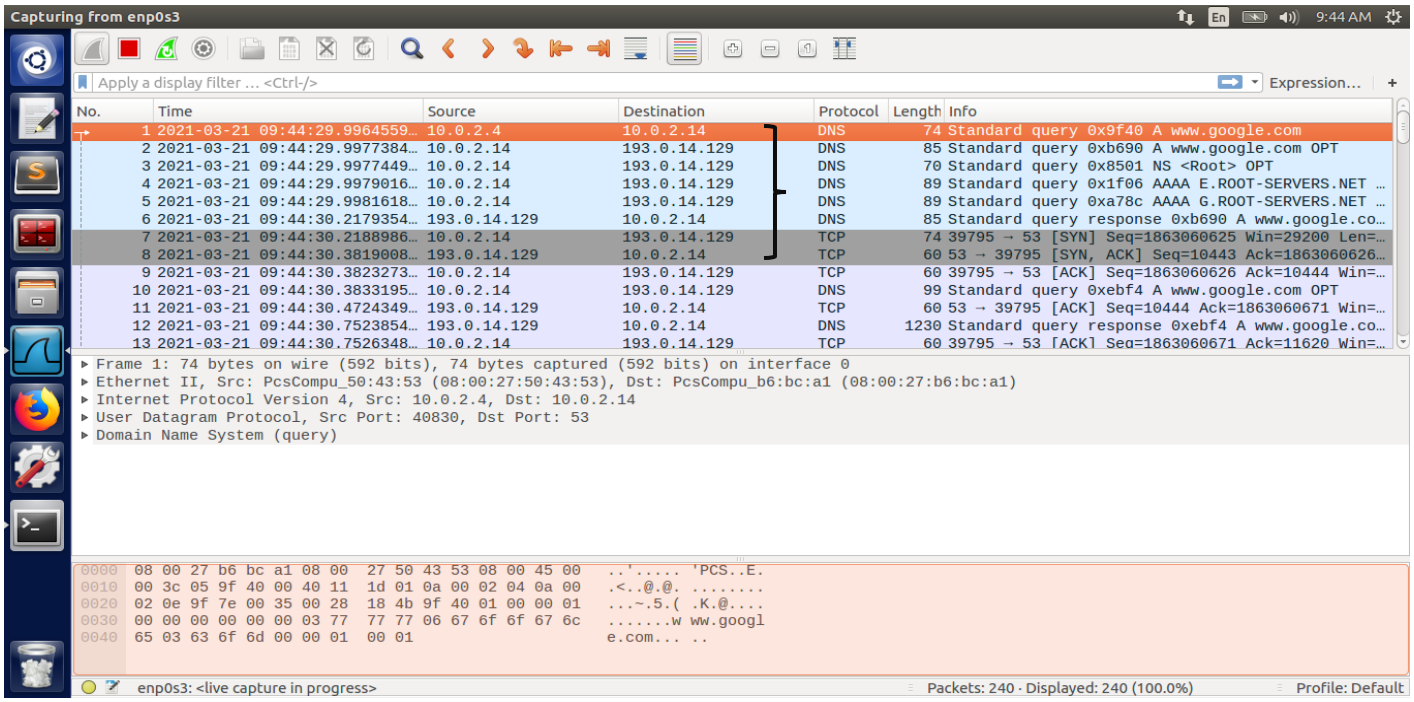
We check the configurations made to the user machine and the machine with the DNS Server. We perform a ping operation on www.google.com site.

The first time we ping the site, we can see from the wireshark screenshot that the DNS query goes to the DNS server (10.0.2.14) and the DNS server further sends the query to the root, TLD and nameservers. The consequent time when we ping, see that the DNS query is directly sent to the IP address of the hostname from the DNS Server.

The reason is the Local DNS Server has cached the IP address of the hostname which is obtained and hence is able to directly send the request to the same instead of going in an iterative way as in the first scenario.



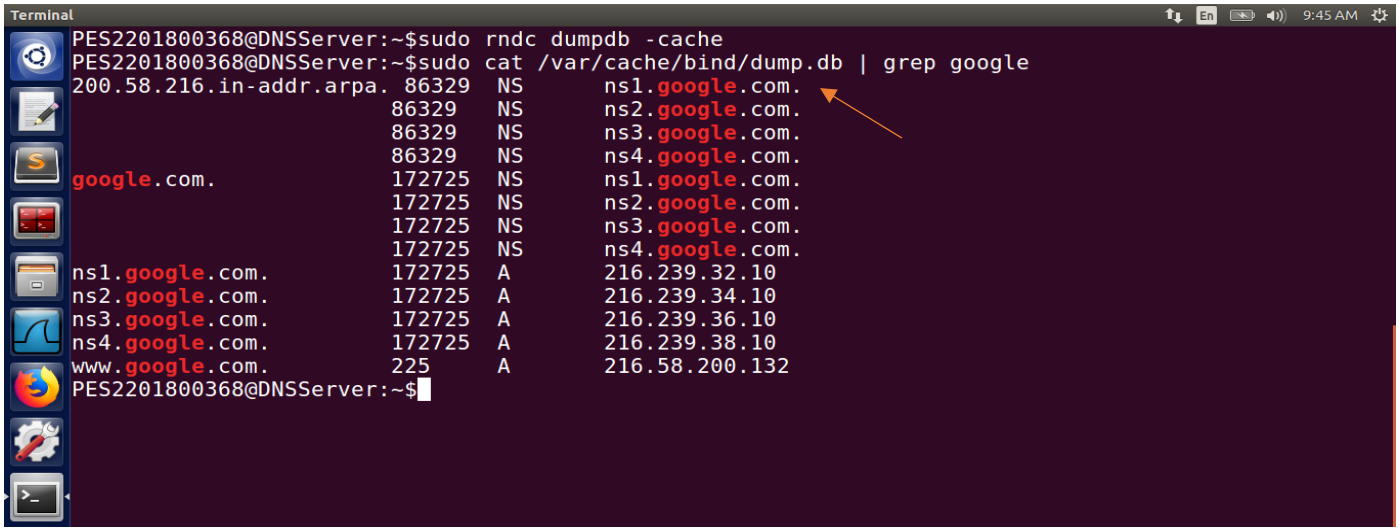
SCREENSHOT OF THE PING OPERATION FROM THE VICTIM MACHINE



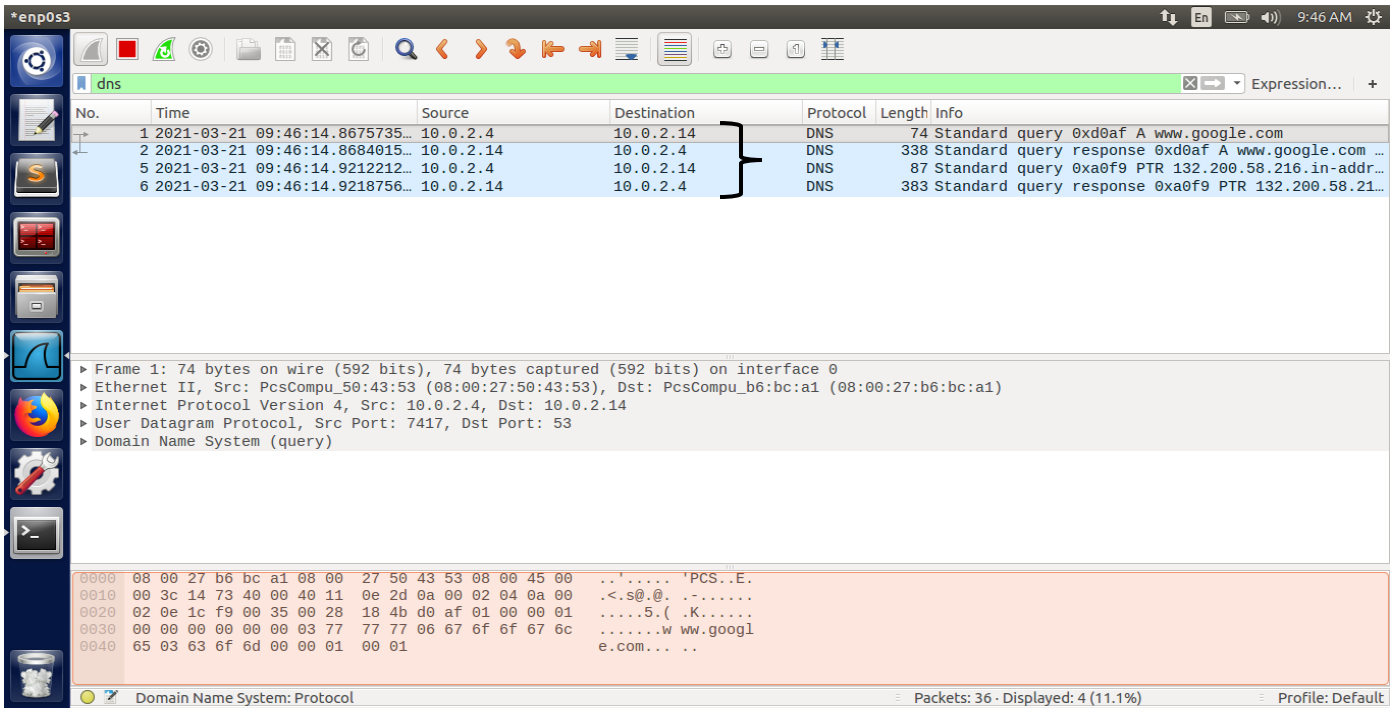
WIRESHARK CAPTURE SHOWING THE ITERATIVE APPROACH OF DNS QUERY

We see from the above screenshot that the DNS query from the victim 10.0.2.4 goes to the DNS Server(10.0.2.14) which further follows an iterative approach and sends the query to the root, TLD and finally the nameservers.

When this approach is followed the first time, the information is also cached by the local DNS Server which is seen in the below screenshot.



SCREENSHOT SHOWING THE DNS SERVER CACHE DETAILS



WIRESHARK CAPTURE OF THE 2nd PING EXEUCTION

Observation: The query directly goes to the hostname of the queried IP unlike the previous case as the records were cached by the local DNS Server. The above executions show the use of local DNS Server and caching which help in obtaining faster responses.

TASK 3: HOST A ZONE IN THE LOCAL DNS SERVER

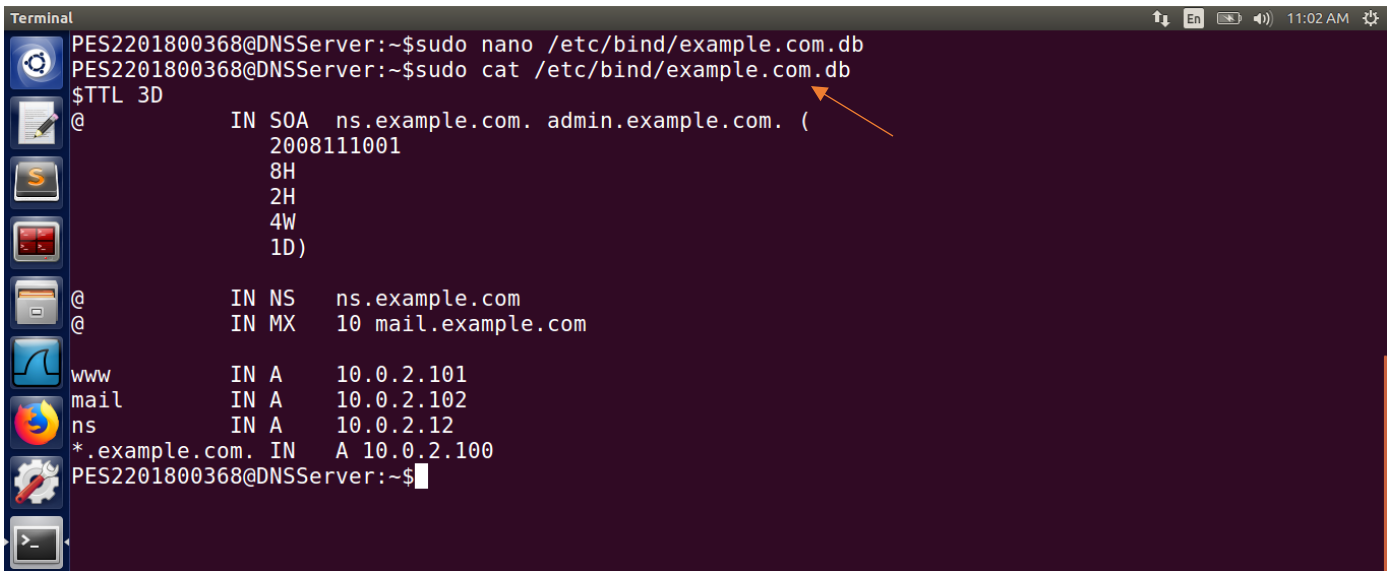
Step 1: Create Zones

We had two zone entries in the DNS server by adding the following contents to /etc/bind/named.conf as shown in the below screenshot. The first zone is for forward lookup (from hostname to IP), and the second zone is for reverse lookup (from IP to hostname).

SCREENSHOT OF THE ZONES CREATED IN THE LOCAL DNS SERVER

Step 2: Setup the forward lookup zone file

We create example.com.db zone file with the following contents in the /etc/bind/ directory where the actual DNS resolution is stored.



SCREENSHOT SHOWING THE FORWARD LOOKUP ZONE FILE

Step 3: Setup the reverse lookup zone file

We create a reverse DNS lookup file called 10.0.2.db for the example.net domain to support DNS reverse lookup, i.e., from IP address to hostname in the /etc/bind/ directory with the following contents.

SCREENSHOT SHOWING THE REVERSE LOOKUP ZONE FILE

Step 4: Restart the BIND server and test

Now we will restart the DNS server and check using "dig" command.

SCREENSHOT SHOWING THE EXECUTION OF DIG COMMAND

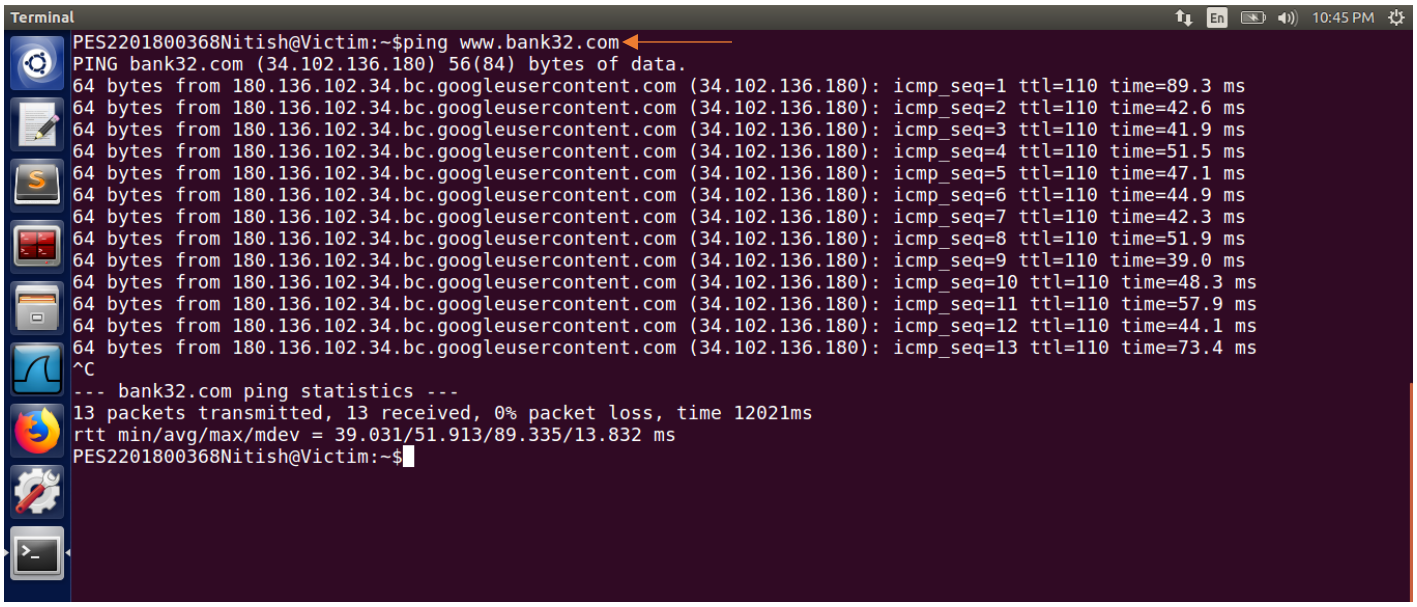
We can see that the ANSWER SECTION contains the DNS mapping. We can see that the IP address of www.example.com is now 10.0.2.101, which is what we have setup in the DNS server.

PART II: ATTACKS ON DNS

TASK 4: MODIFYING THE HOST FILE

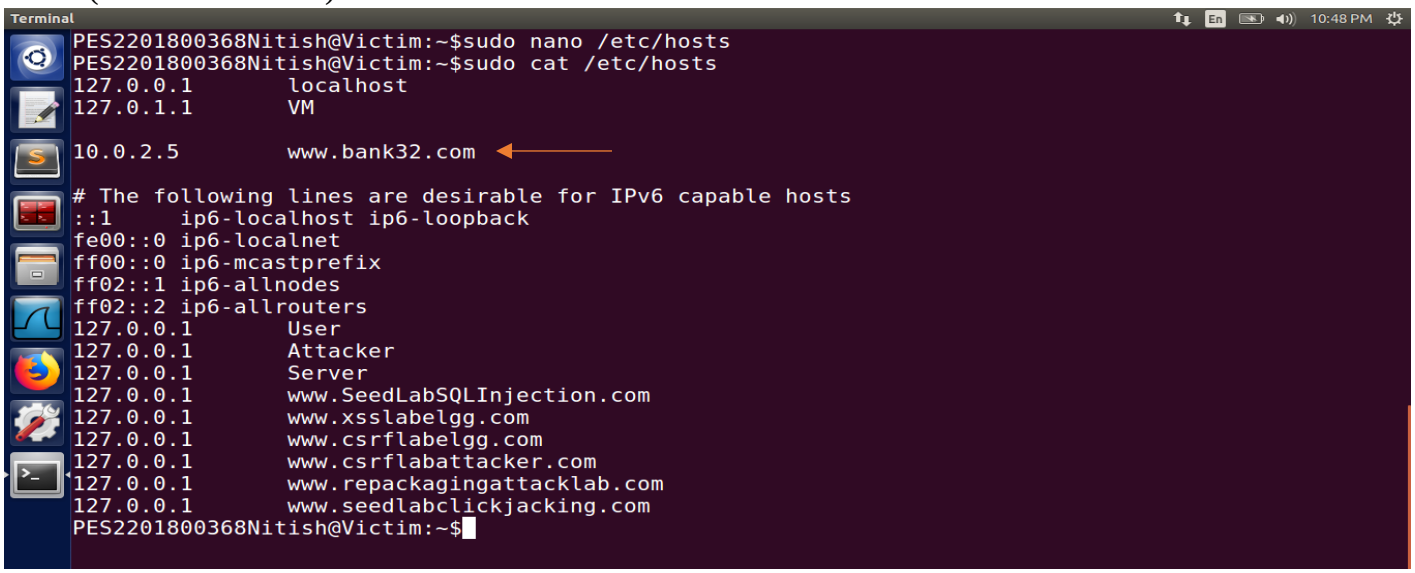
Aim: - To perform DNS attack if the victim's machine is compromised.

On the victim machine, ping www.bank32.com



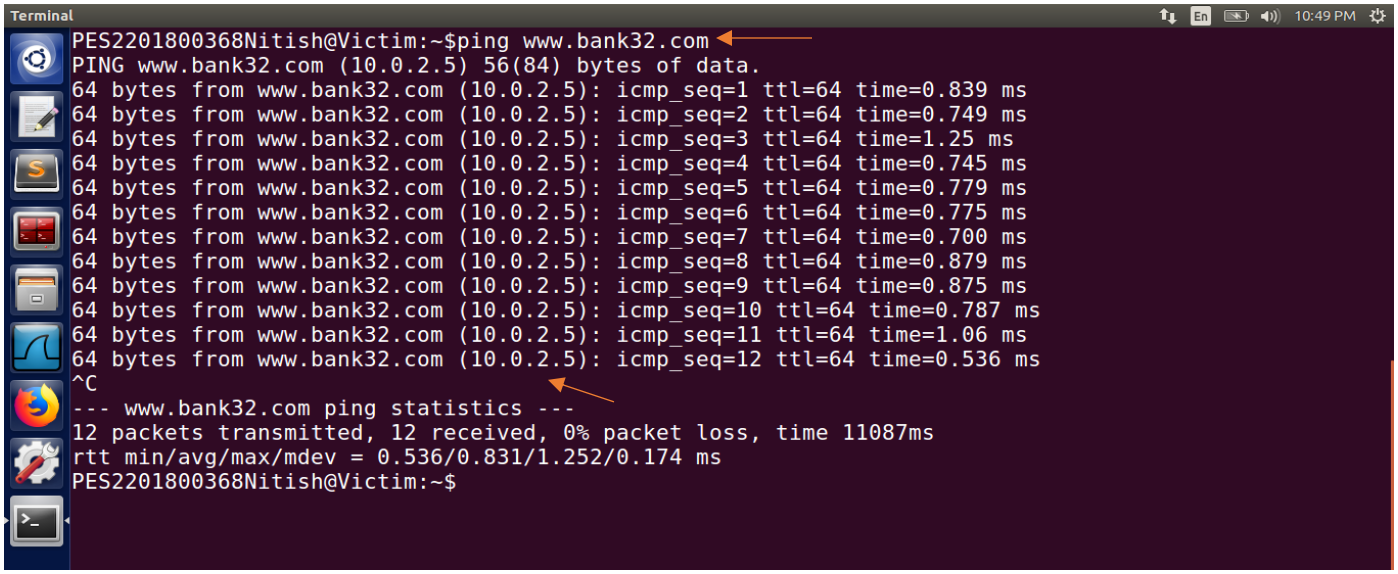
SCREENSHOT SHOWING THE EXECUTION OF PING

Now, we modify the /etc/hosts file with the IP address of www.bank32.com as 10.0.2.5 (Attacker machine).

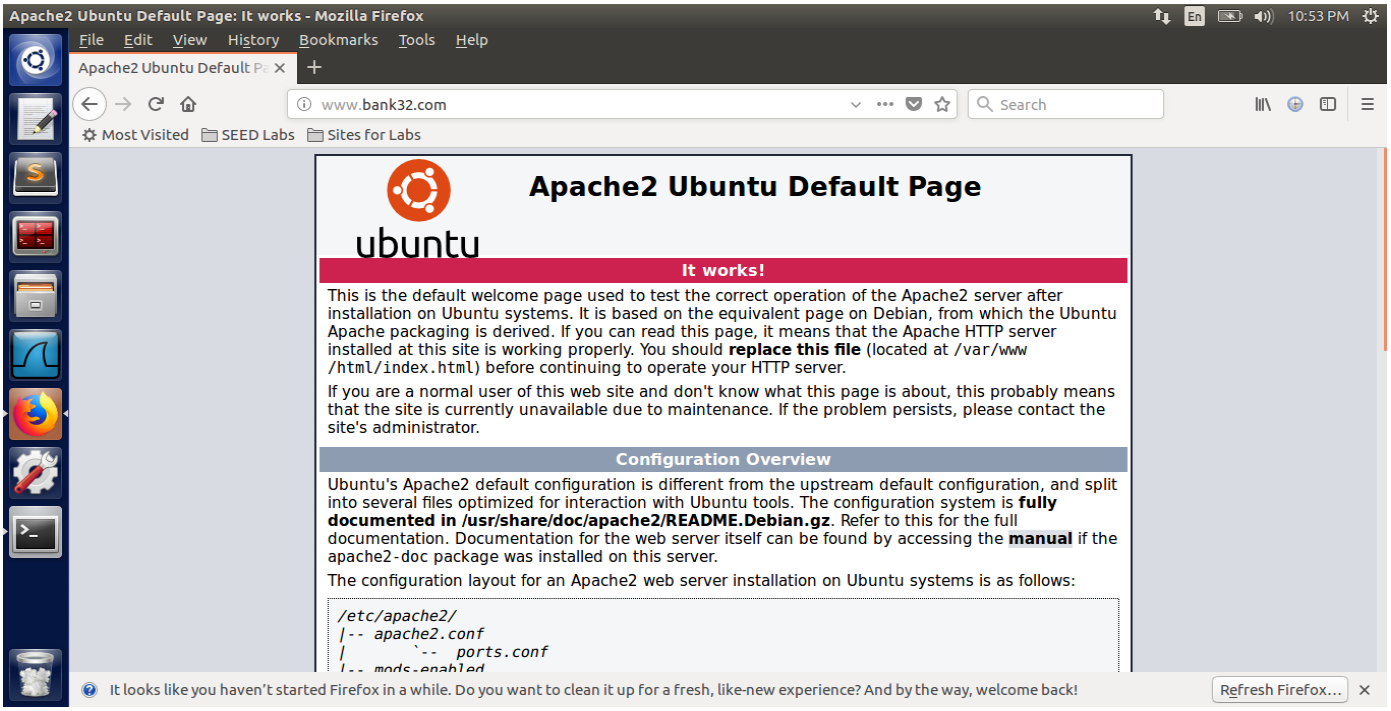


SCREENSHOT SHOWING THE MODIFICATION OF /etc/hosts FILE IN THE VICTIM MACHINE

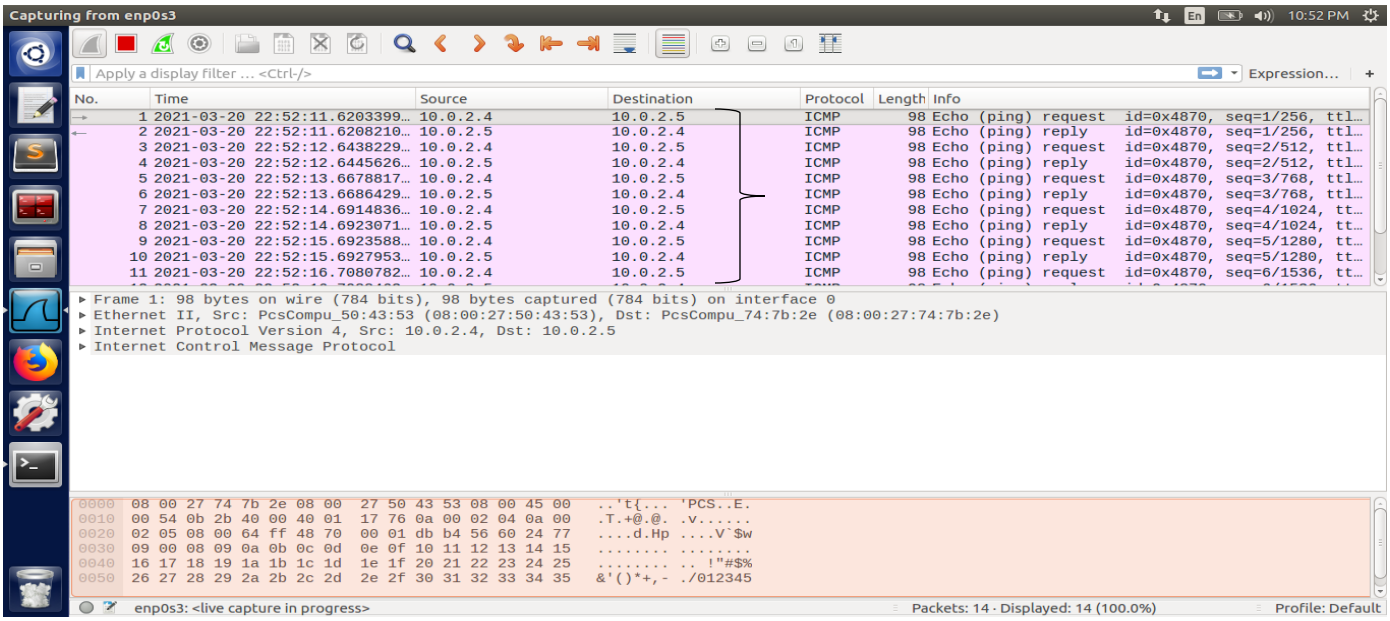
We see that when we send ping requests to www.bank32.com, we get ICMP response back from the "10.0.2.5" (Attacker machine) as well as the browser gives the default page of the Apache server running in the Attacker machine. /etc/hosts file is used for the local look up for the IP address of the look up. Hence, if the file is compromised the attacker can redirect the user to a malicious page.



SCREENSHOT SHOWING THE ICMP RESPONSE FROM ATTACKER IP TO PING COMMAND



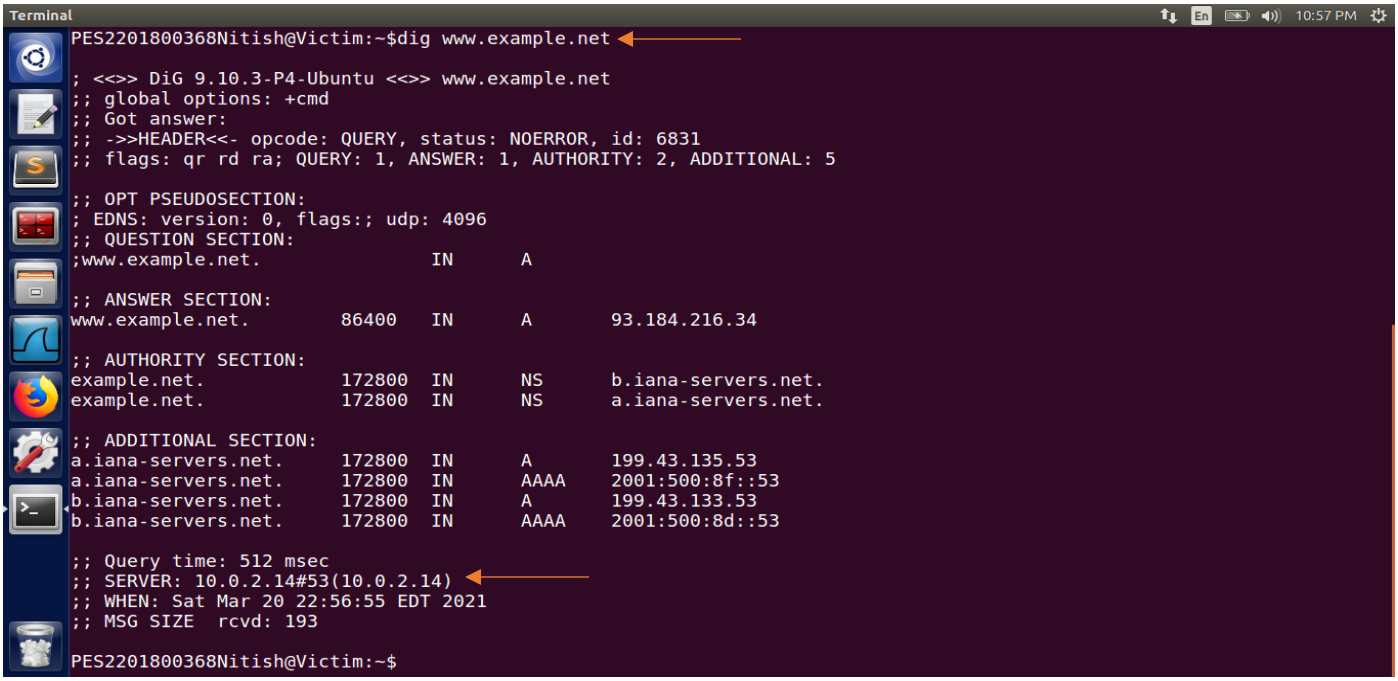
SCREENSHOT SHOWING THE BROWSER SCREENSHOT OF THE ATTACKER DEFAULT APACHE SERVER PAGE ON THE VICTIM MACHINE



WIRESHARK CAPTURE SHOWING ICMP RESPONSE FROM ATTACKER TO VICTIM MACHINE ON ACCOUNT OF THE FILE MODIFYING ATTACK

Observation: Adding new records to the /etc/hosts file in the victim machine by providing the IP addresses for some selected domains may lead to the queries from the victim machine going to malicious sites which can lead to further attacks on the system.

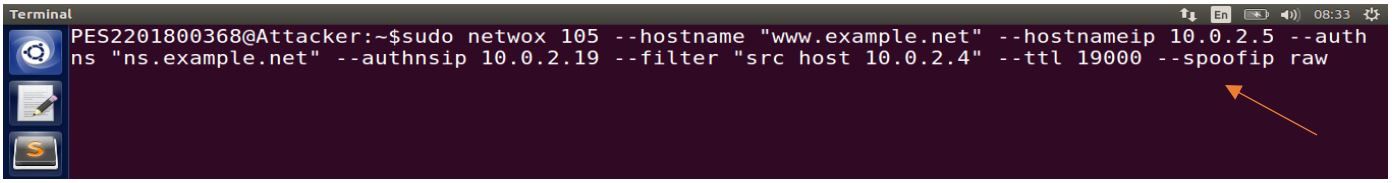
TASK 5: DIRECTLY SPOOFING RESPONSE TO USER



SCREENSHOT SHOWING EXEUCTION OF THE DIG COMMAND

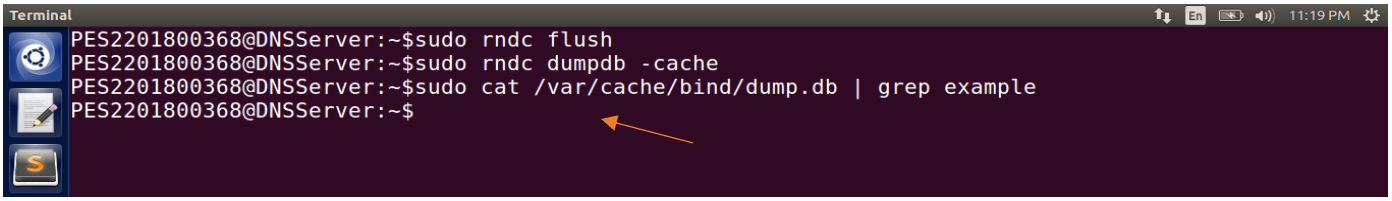
The above screenshot shows the output of the dig command for www.example.net from the victim machine. The victim machine sends out a DNS query to the local DNS server(10.0.2.14), which will eventually send out DNS query to the authoritative nameserver of the example.net domain.

In this attack, we target the DNS queries from the victim machine (10.0.2.4). In our forged reply, we map www.example.net to 10.0.2.5 and authoritative server as 10.0.2.19. The netwox tool sniffs the DNS query packet from host "10.0.2.4" (filter) and responds with a forged DNS response packet.

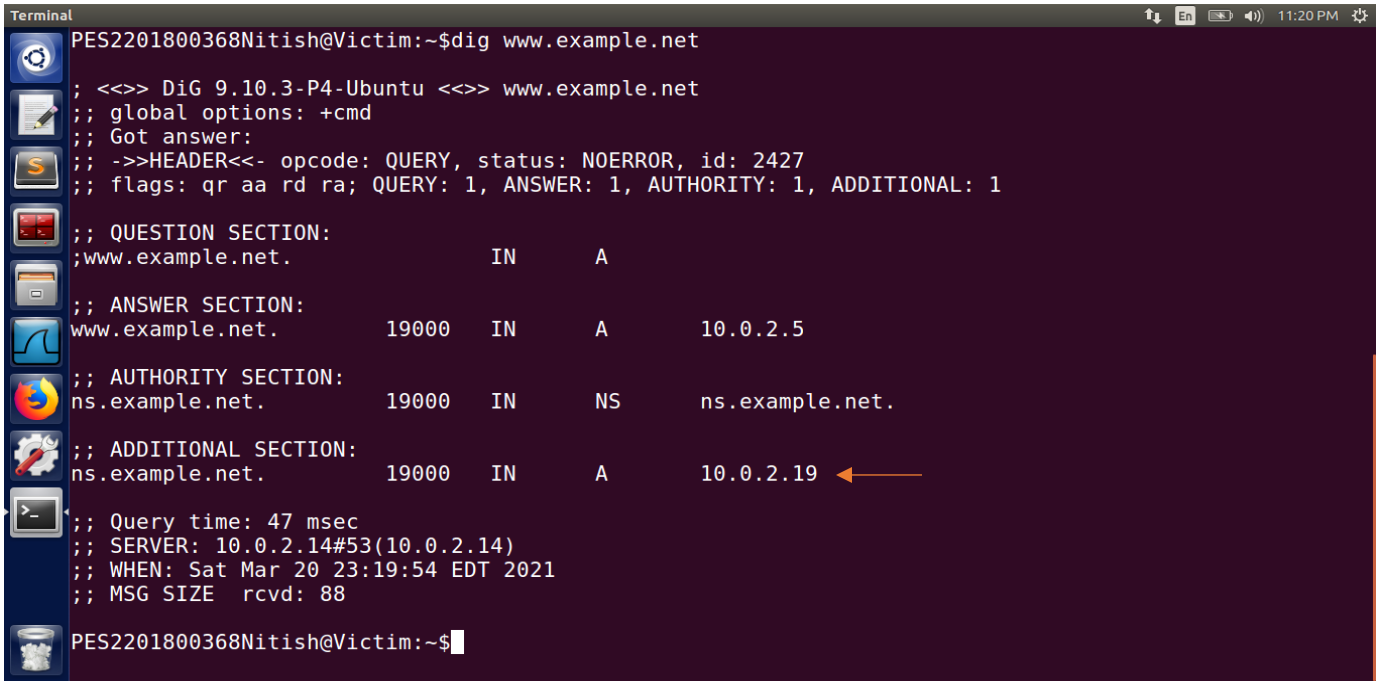


SCREENSHOT SHOWING THE LAUNCH OF ATTACK

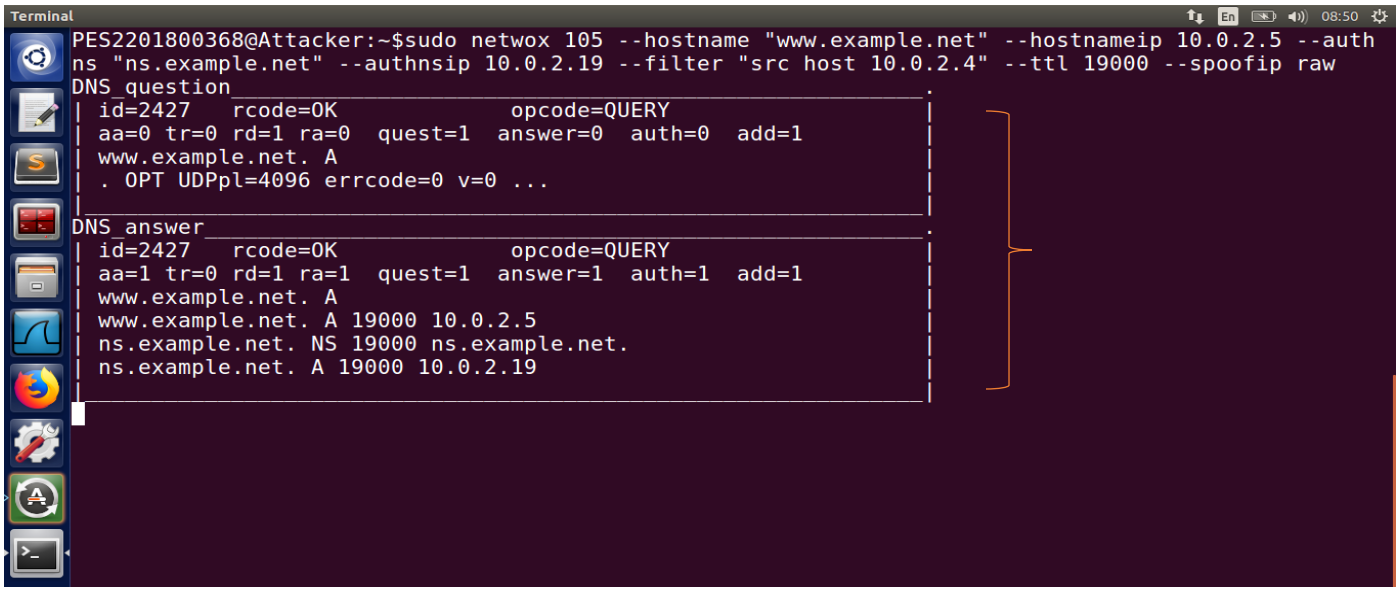
We now clean the cache in the DNS Server machine and retry the dig command for www.example.net from the victim machine and observe the response obtained.



SCREENSHOT SHOWING CLEARING OF THE CACHE AND VERIFYING THE SAME BY SEARCHING THE CACHE FILE FOR RECORDS WITH THE TERM 'example' WHICH CONFIRMS THAT CACHE IS CLEARED



SCREENSHOT SHOWING THE SUCCESS OF THE ATTACK USING THE DIG COMMAND



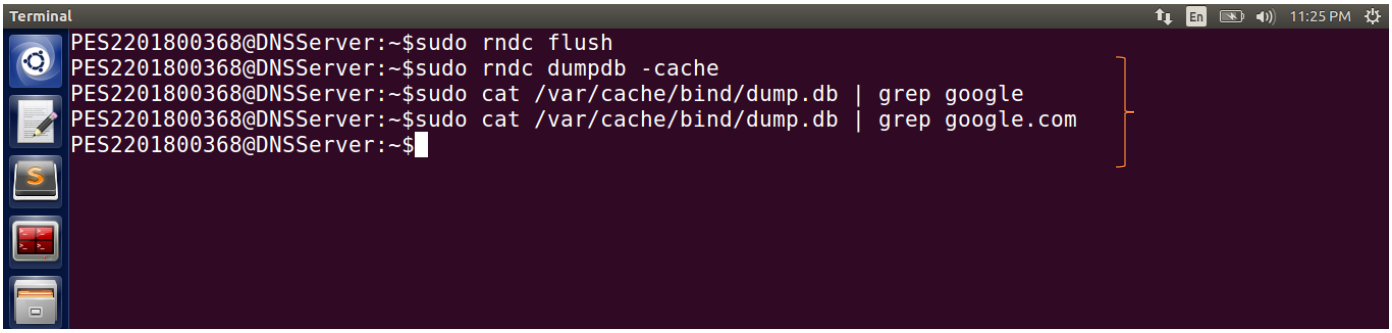
SCREENSHOT SHOWING THE RESULT OF THE ATTACK IN THE ATTACKER MACHINE ONCE THE DIG COMMAND WAS RUN IN THE VICTIM

Observation: When the user machine sends out a DNS query to its local DNS sever, the attacker can immediately send a spoofed reply, using the local DNS server as its source IP address. This response appears to the user machine to have come from the local DNS Server and hence it will be accepted. The attackers can put any arbitrary IP address in the reply which can lead to redirecting the user to malicious sites.

TASK 6: DNS CACHE POISONING ATTACK

Aim: - To target the DNS queries from the local DNS server (10.0.2.14), forge the response sent and poison the cache.

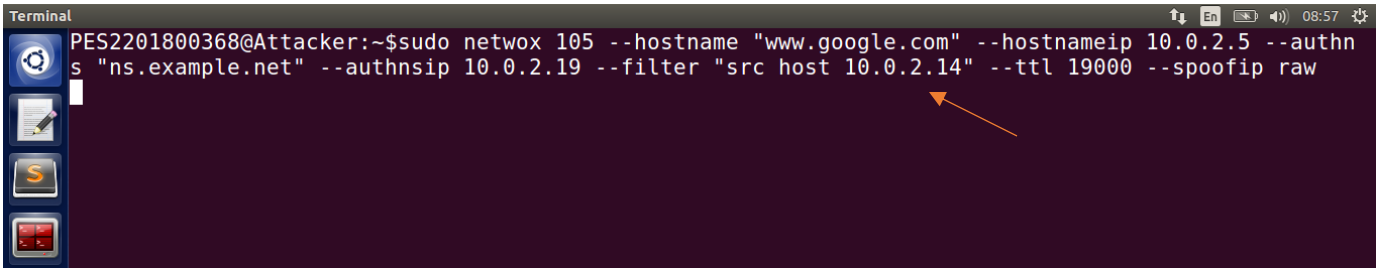
Before the start of the experiment we clear the cache in the DNS Server and verify the same.



SCREENSHOT SHOWING THE CACHE BEING CLEARED AND VERIFICATION OF THE SAME

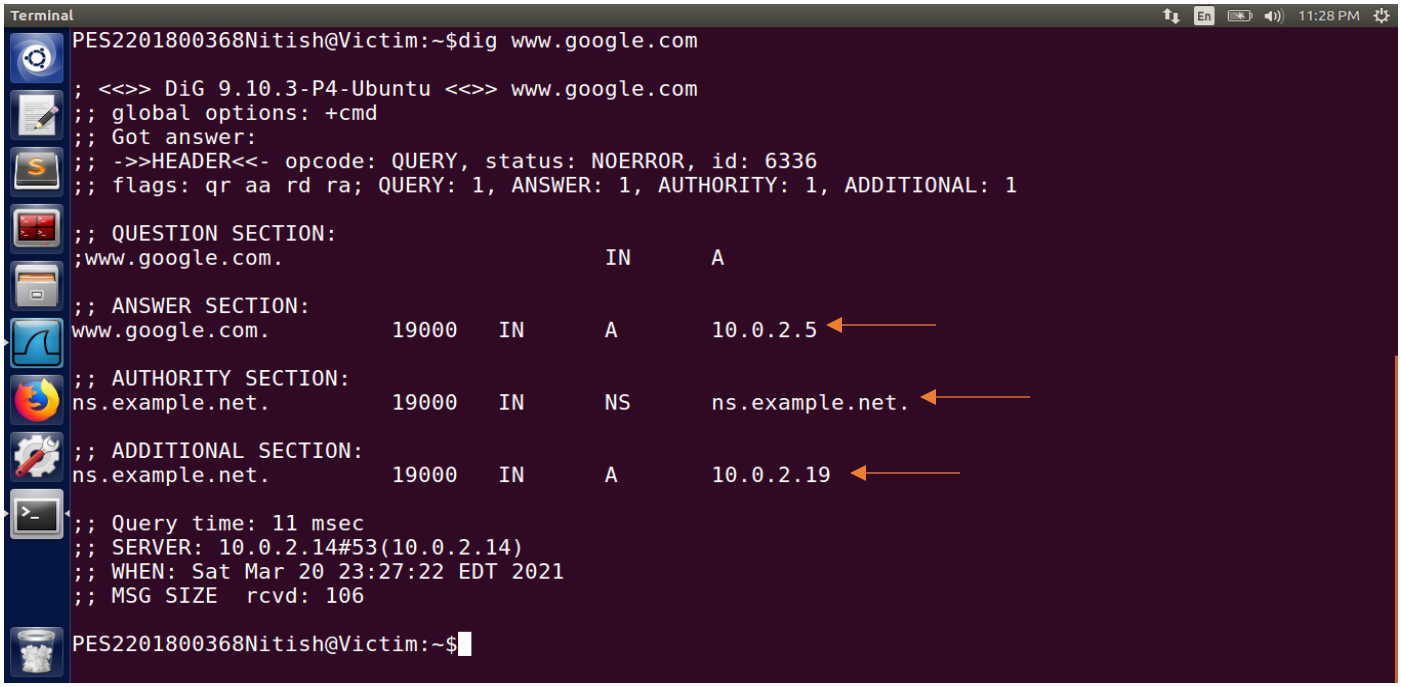
In this attack, we target the DNS queries from the local DNS server (10.0.2.14). In our forged

reply, we map www.google.com to 10.0.2.5 and authoritative server as 10.0.2.19. The netwox tool sniffs the DNS query packet from the DNS server "10.0.2.5" (filter) and sends the forged DNS response packets to the DNS server.



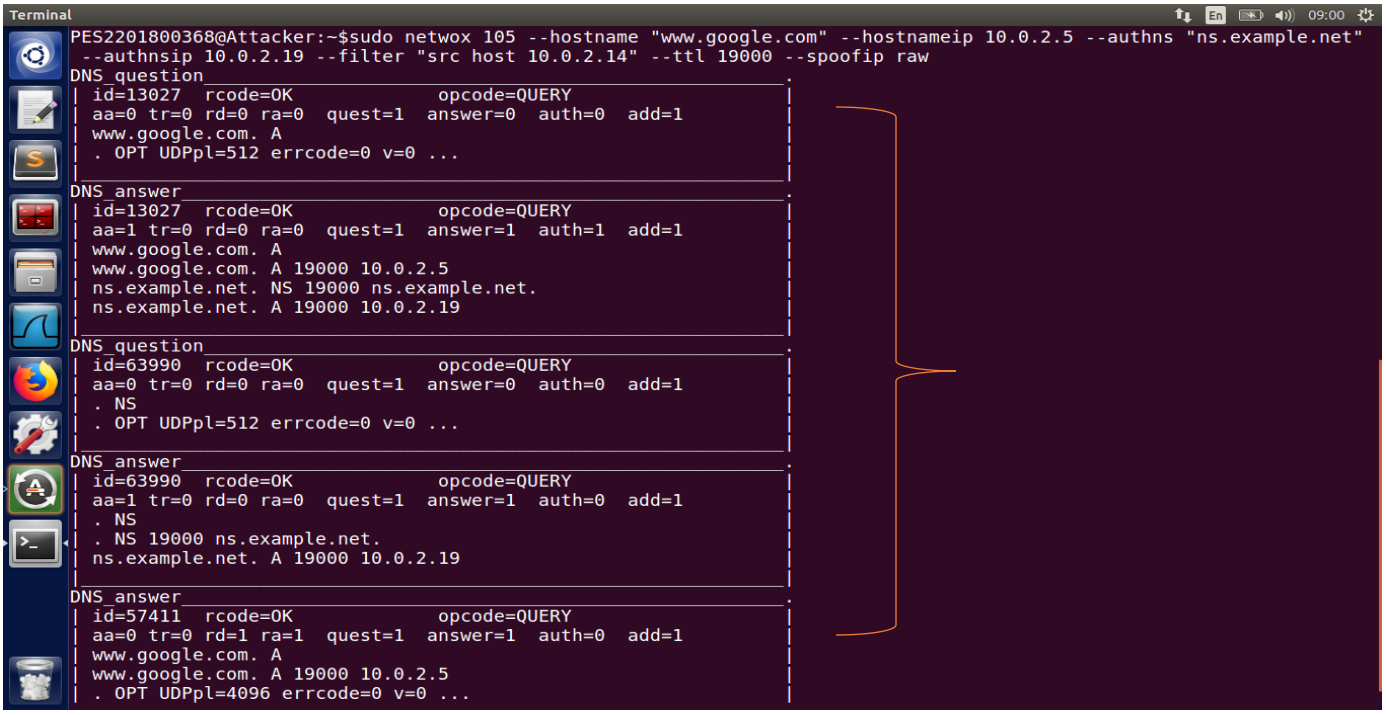
SCREENSHOT OF THE LAUNCHED ATTACK

When the attack program is running, we run "dig www.google.com" on behalf of the user. From the below screenshot, we see that the victim machine gets forged reply from the DNS server with Answer section, Authority section and Additional section.

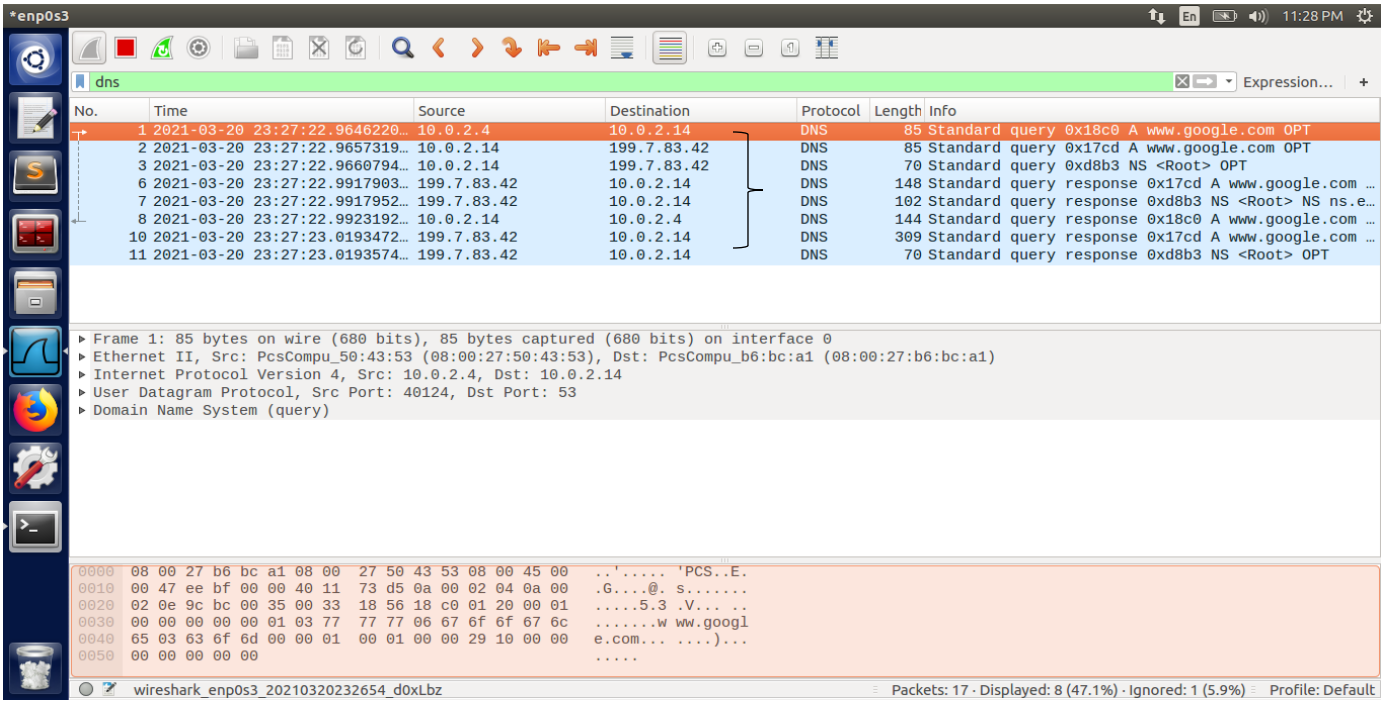


SCREENSHOT SHOWING THE DIG COMMAND EXECUTION WHEN THE ATTACK IS RUNNING

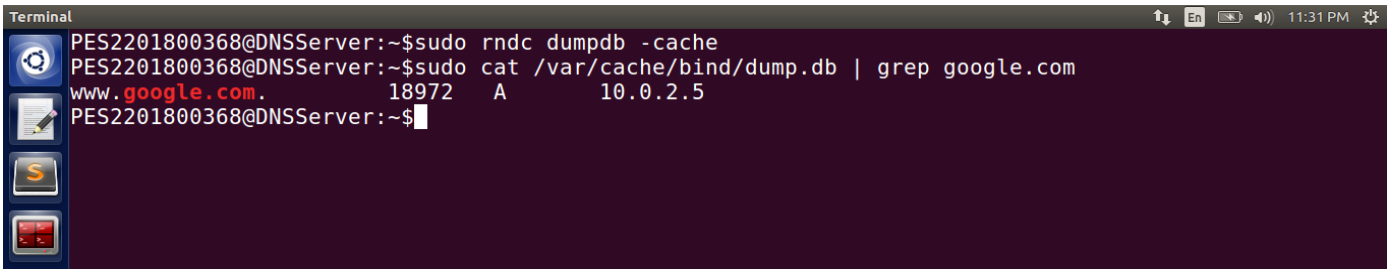
We now check the same on wireshark and verify the contents of the cache on the DNS Server.



SCREENSHOT SHOWING THE CONTENT OF SENDING A DNS QUERY FROM THE VICTIM MACHINE ON THE ATTACKER MACHINE



WIRESHARK CAPTURE SHOWING THE DNS SERVER SENDING FORGED RESPONSES AFTER THE CACHE WAS POISONED SUCCESSFULLY



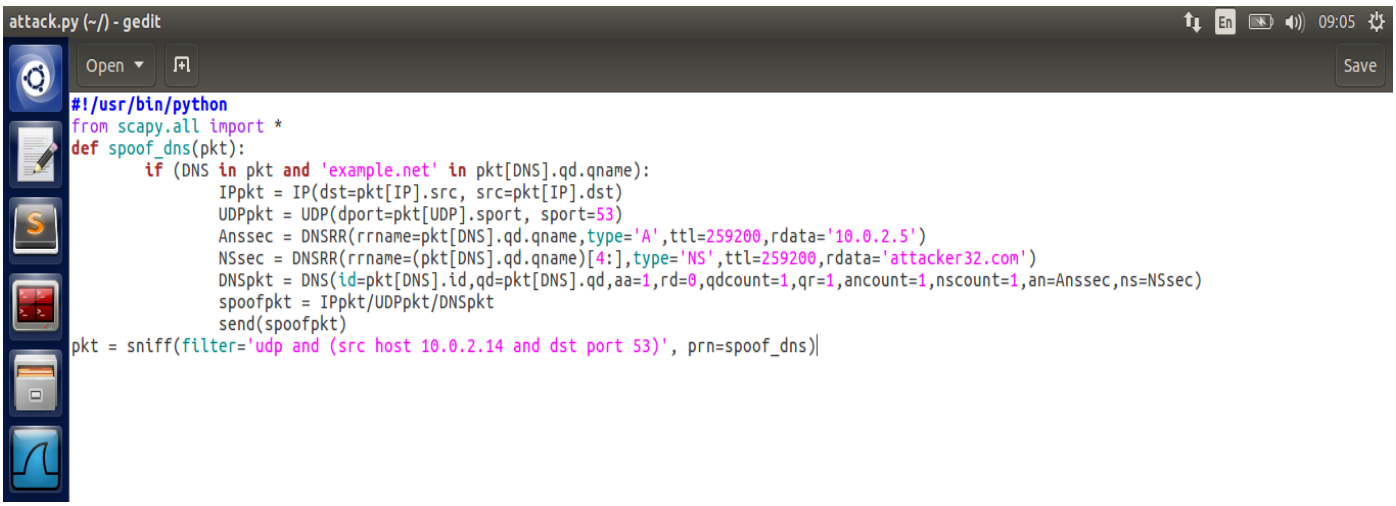
SCREENSHOT SHOWING THE CACHE DETAILS ON THE DNS SERVER

Observations from the above attack: Cache poisoning attack, When the local DNS server sends out iterative queries to get an answer from the DNS servers on the Internet, attackers can send out spoofed replies to the local DNS server. As long as the spoofed replies arrive earlier than the legitimate ones, they will be accepted and cached. When the user performs a query further, the poisoned cached details are sent which may lead the user to malicious sites.

TASK 7: DNS CACHE POISONING: TARGETING THE AUTHORITY SECTION

Aim: - To launch DNS Cache poisoning using Authority section in DNS replies.

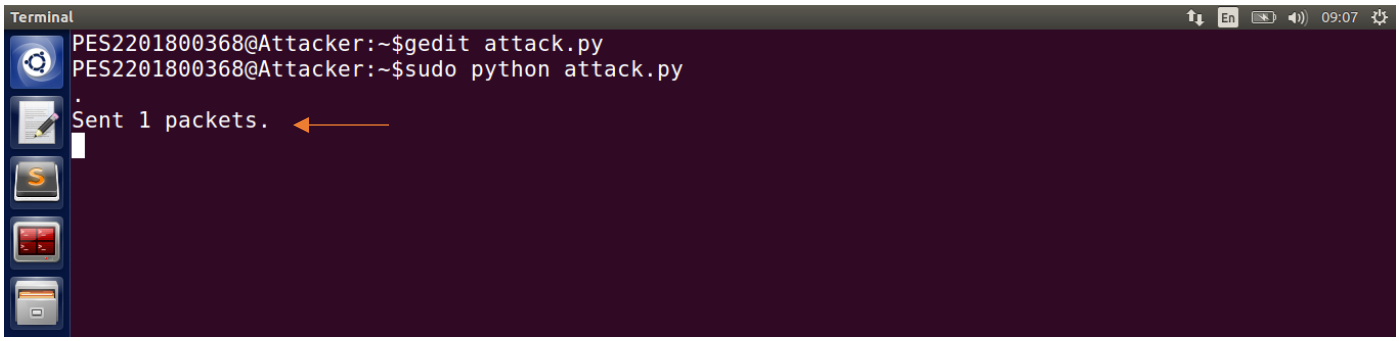
In addition to spoofing the answer (in the Answer section), we also add "ns.attacker32.com" the Authority section. When this entry is cached by the local DNS server, ns.attacker32.com will be used as the nameserver for future queries of any hostname in the example.net domain. To achieve this task, we will write a Scapy code to sniff DNS requests from DNS server and send spoofed DNS response with answer section and authoritative section.



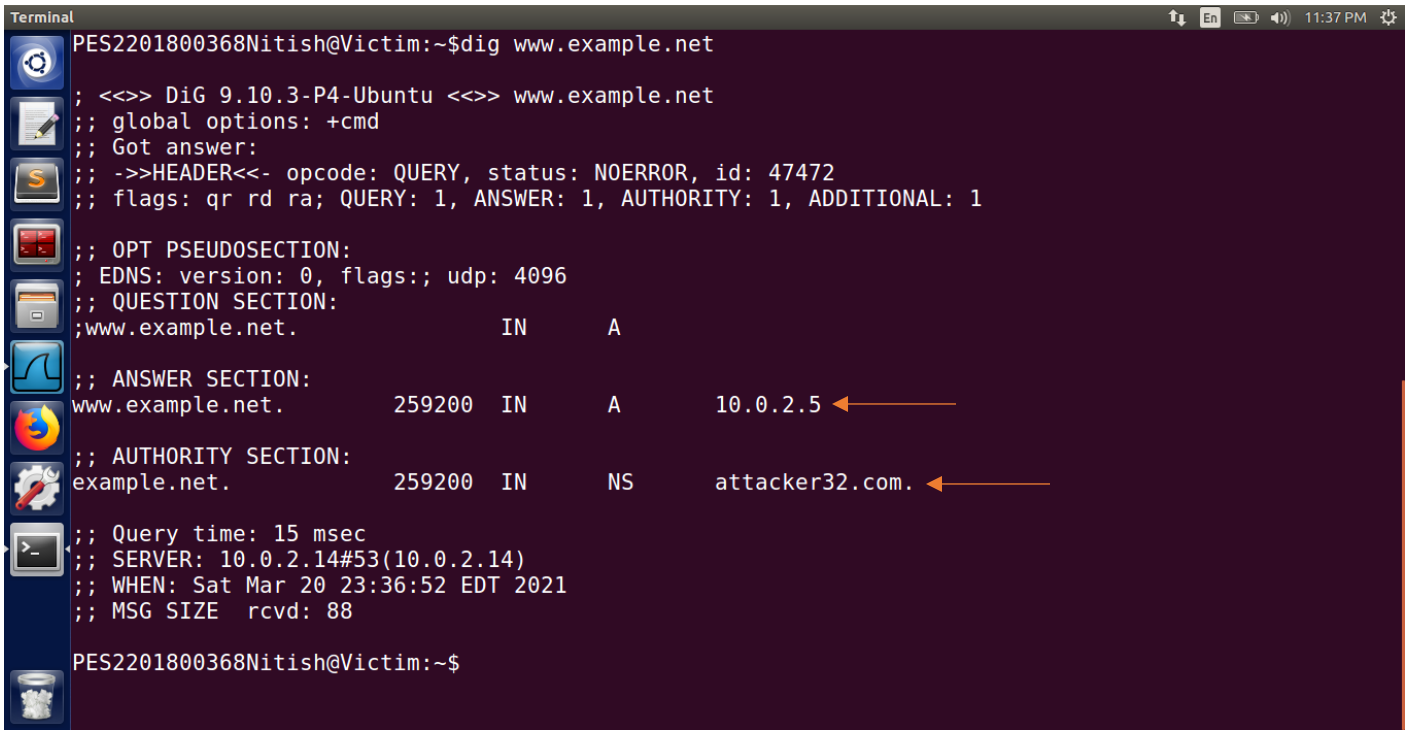
SCREENSHOT SHOWING THE SCAPY CODE

We clear the cache before the attack in run.

In the above code, we sniff packets with UDP protocol, source IP address as 10.0.2.14 (DNS server) and destination port as 53. When the DNS server sends DNS query requests, our program sniffs for the packets, creates new DNS response packets with Answer section, Authority section. For answer and authority sections, we create DNS Response records with Resource Record name, Record type as 'Answer (A)' or 'NameServer (NS)', resource data as IP address or domain name.

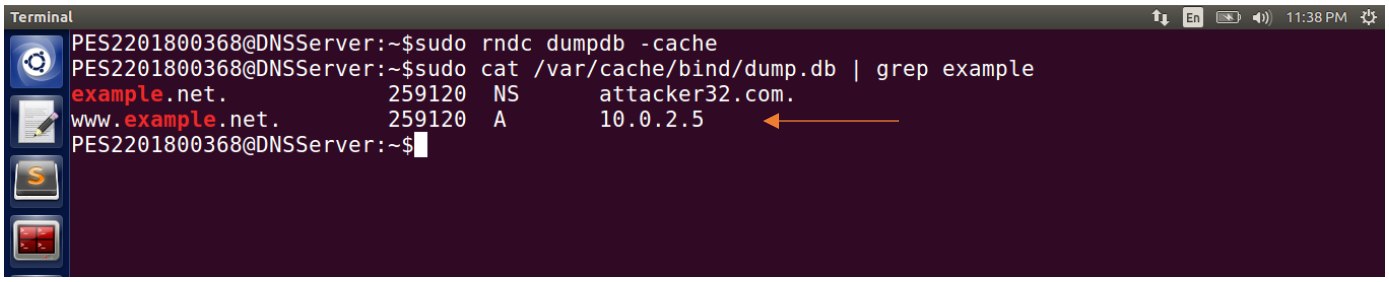


SCREENSHOT SHOWING THE LAUNCHED ATTACK



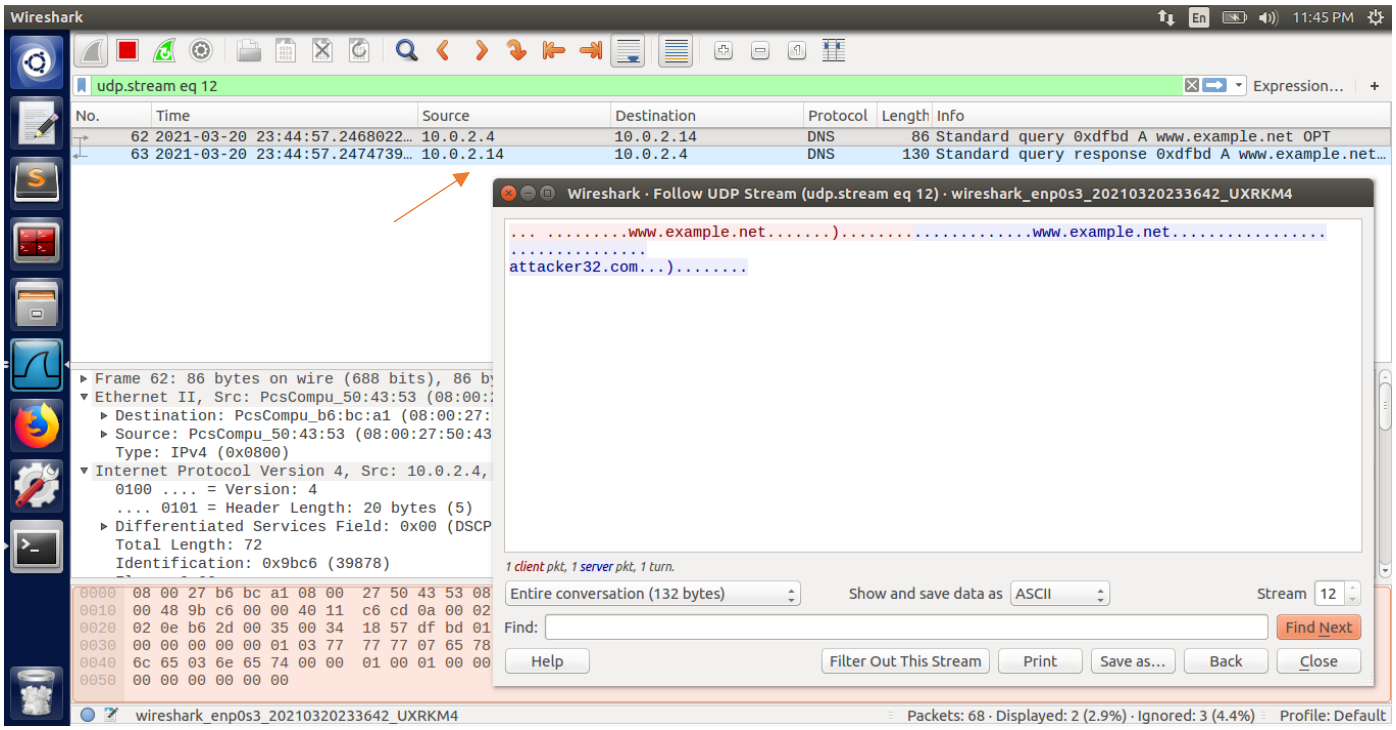
SCREENSHOT SHOWING THE EXECUTION OF DIG COMMAND ON THE USER MACHINE

Observation: From the below screenshot, we see that the victim machine gets a forged reply from the DNS server with the Answer section and Authority section. The authority section contains the forged response of "ns.attacker32.com".



SCREENSHOT OF THE CACHE CONTENTS AFTER THE CACHE POISONING ATTACK

Using Wireshark, we can see that DNS query request is sent to our DNS server (10.0.2.14). As the server knows the nameserver of the domain (due to cache poisoning), it sends query requests for attacker32.com. It shows that our cache poisoning attack is successful.

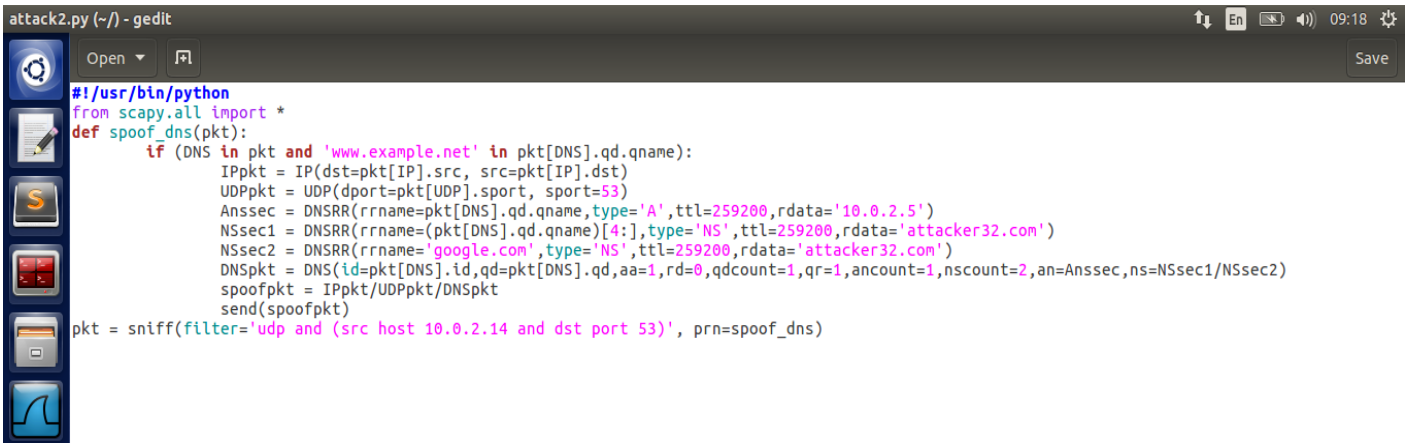


WIRESHARK CAPTURE

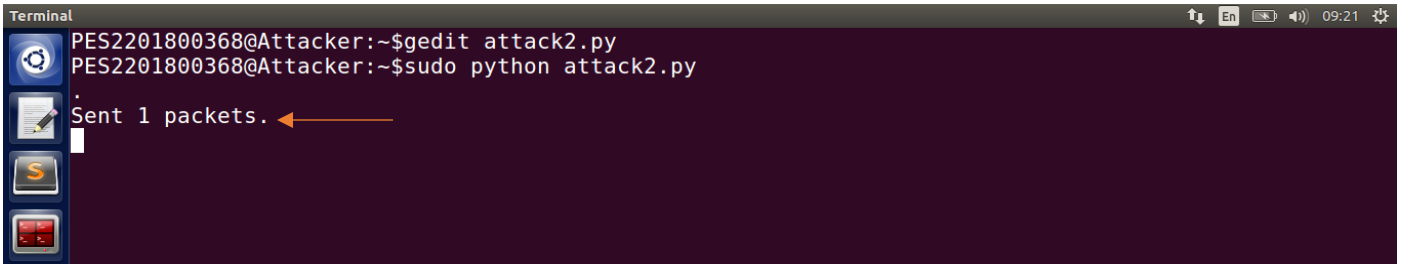
TASK 8: TARGETING ANOTHER DOMAIN

Aim: - To extend the impact of the cache poisoning with a nameserver for the domain to other domains.

We would like to add an additional entry in the Authority section so attacker32.com is also used as the nameserver for google.com. The following code sends DNS response with two authoritative entries similarly as we did for the previous task. We create a Name Server Resource Record for "google.com" with "attacker32.com" as the domain.



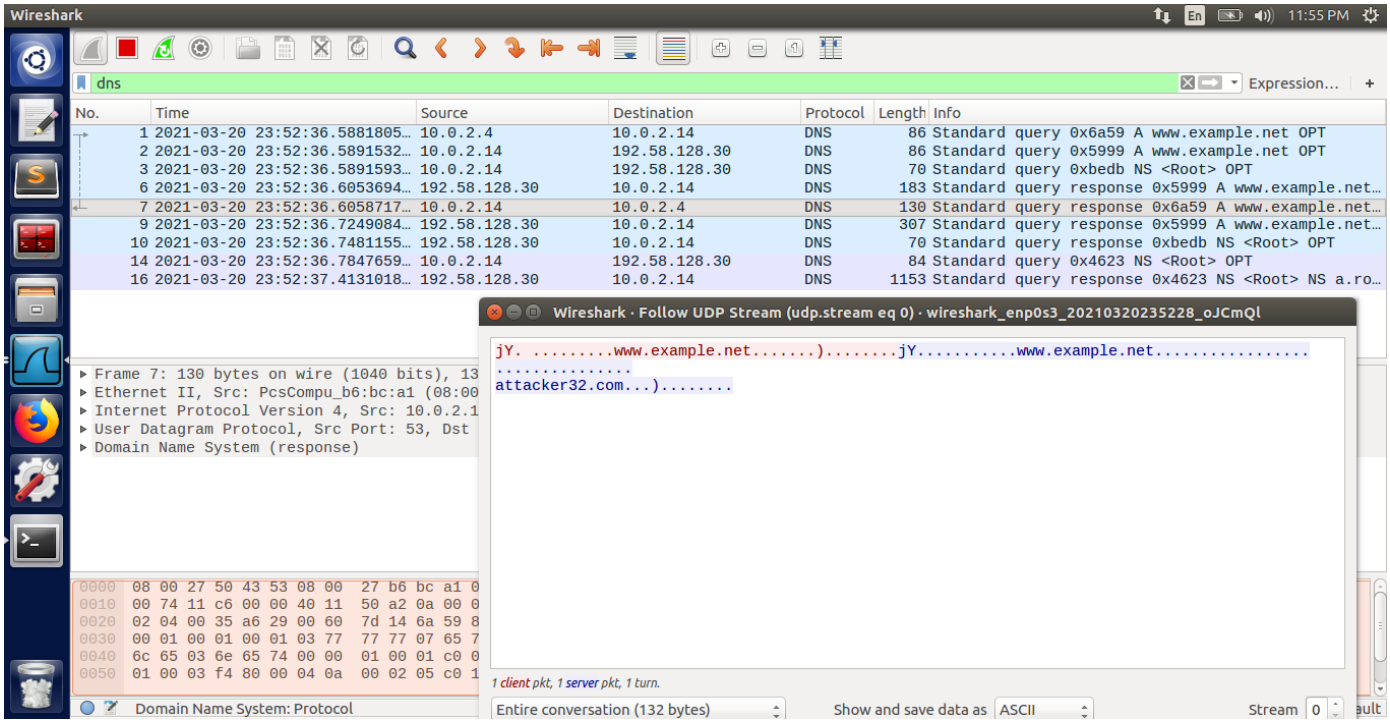
SCREENSHOT OF THE SCAPY CODE



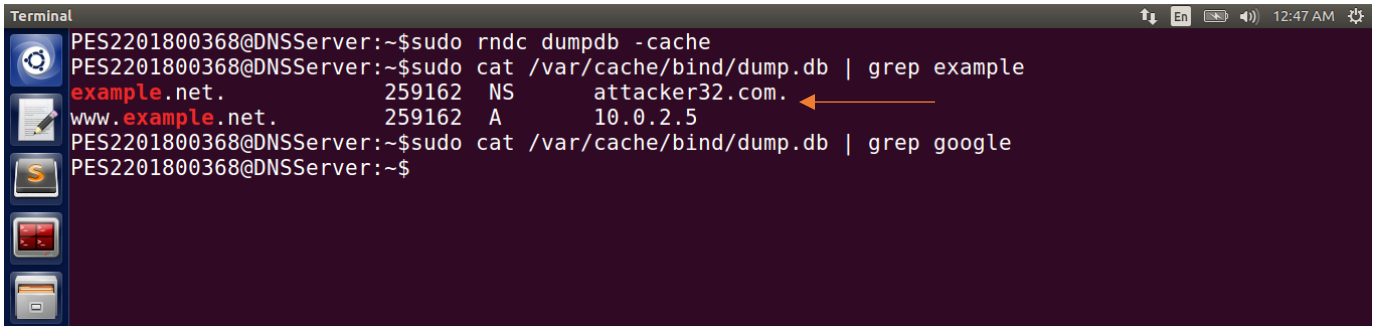
SCREENSHOT OF LAUNCHING THE ATTACK

SCREENSHOT OF THE DIG COMMAND EXECUTION AFTER LAUNCHING THE ATTACK

After running the code, we run "dig www.example.net" on behalf of the user. From the above screenshot, we see that the victim machine gets forged reply from the DNS server with the Answer section and Authority section. The authority section contains the forged response for "example.net", but not for the "google.com".



WIRESHARK CAPTURE OF THE DNS QUERY & RESPONSE AFTER THE ATTACK



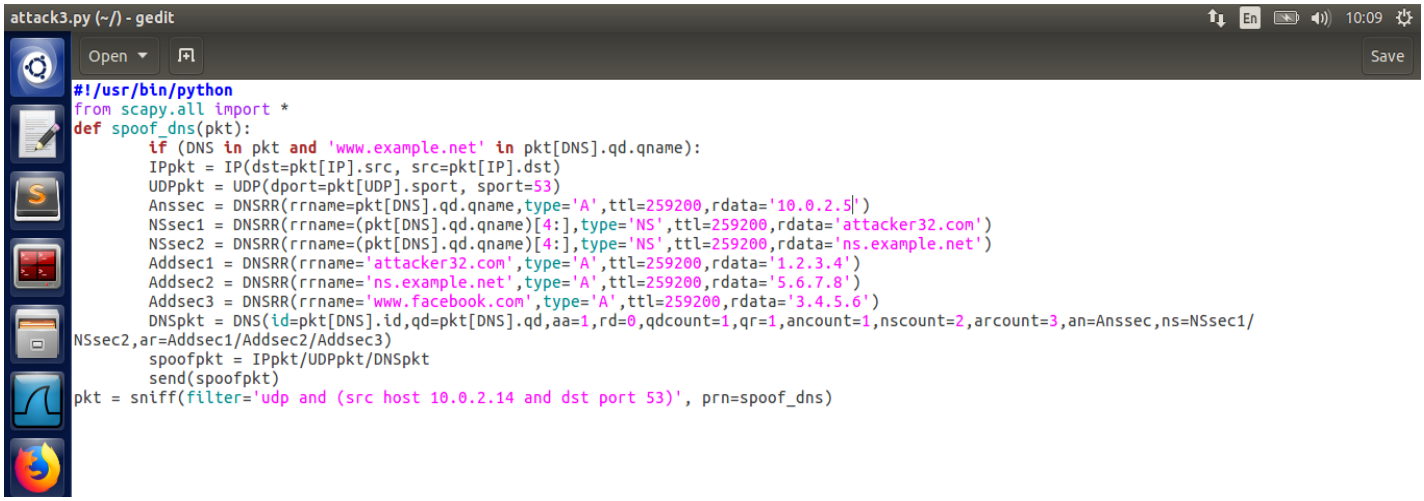
SCREENSHOT SHOWING THE CONTENTS OF THE CACHE

Observation: We can see that the forged DNS response in the code contains the entries for two authoritative nameservers: "example.net" and "google.com". The second record for the authority section is fraudulent and hence it is discarded and not cached. The decision is based on zones. The query is sent to example.net zone, so the DNS resolver will use example.net to decide whether the data in the information is inside this zone or outside. The first record is inside the zone, so it is accepted. Also, though "attacker.com" is not in the same zone is "example.net", it is fine because a domain's authoritative name servers do not necessarily need to be a name inside the domain. However, the second record is "google.com", which is not inside the zone of "example.net", so it will be discarded.

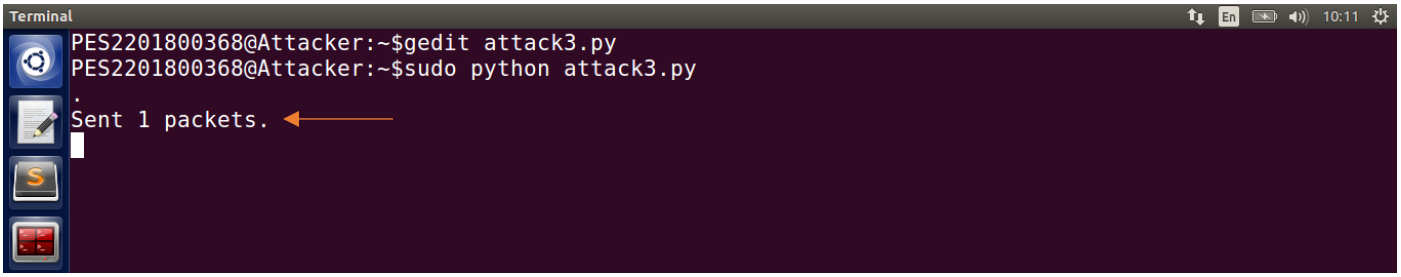
TASK 9: TARGETING THE ADDITIONAL SECTION

Aim: - To spoof some entries in the Additional section and check whether they will be successfully cached by the target local DNS server.

When responding to the query for www.example.net, we add the additional entries in the spoofed reply, along with the entries in the Answer section. We need to add the Answer section in the DNS packet as Additional record (ar).
The code shows that we create Additional sections with the Resource Record name (domains/nameservers) and Resource data (IP address). The resource record names are "attacker32.com", "ns.example.com", "www.facebook.com".

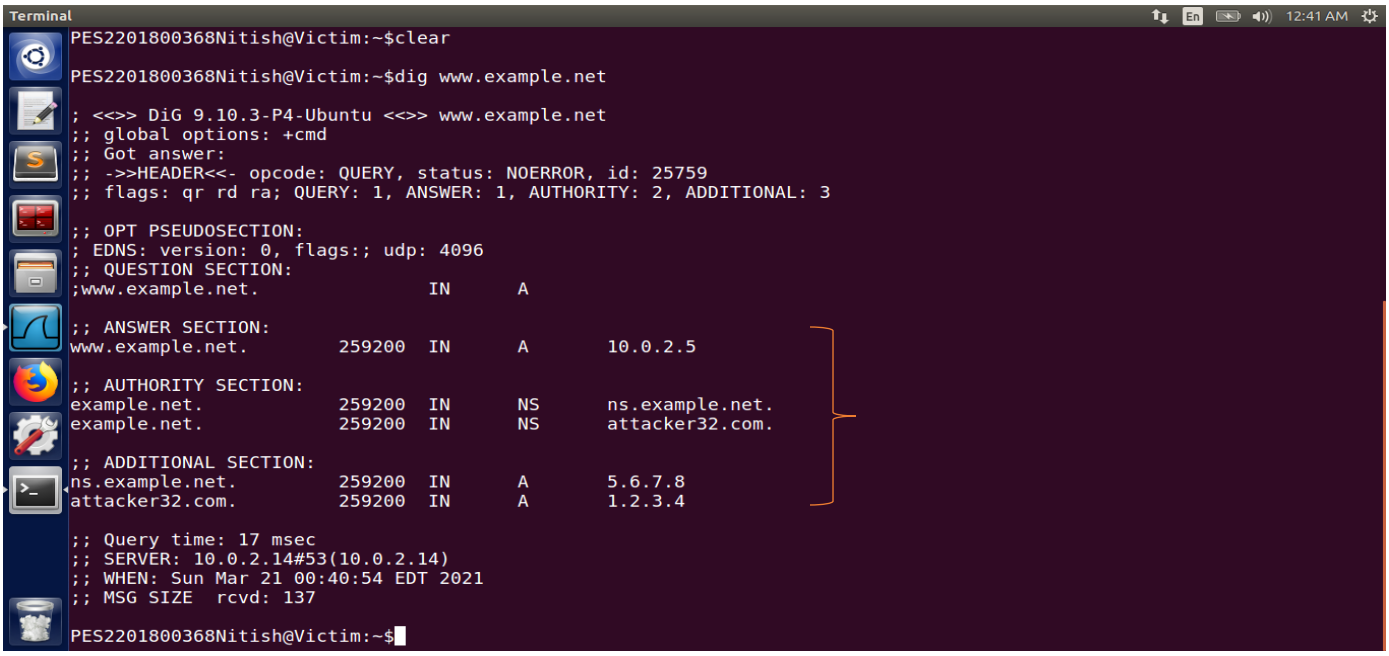


SCREENSHOT SHOWING THE SCAPY CODE

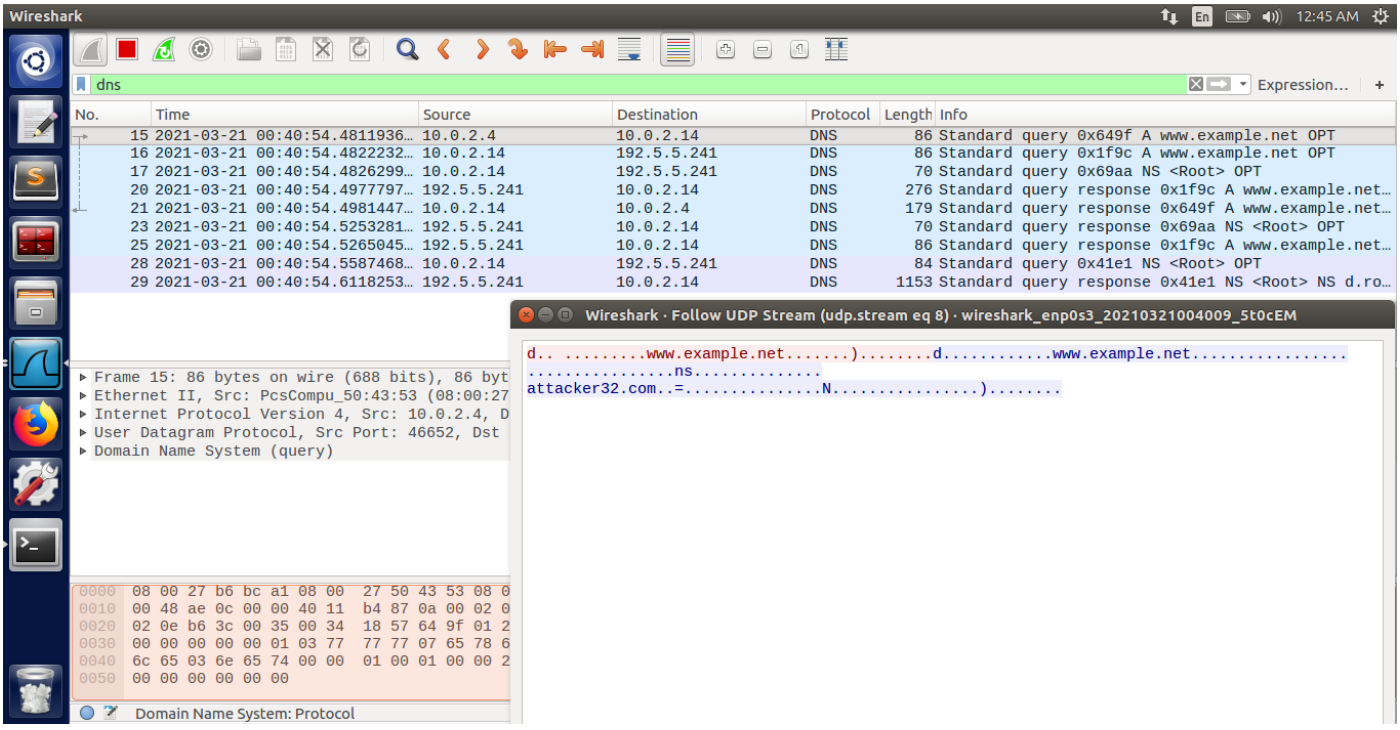


SCREENSHOT SHOWING THE LAUNCH OF THE ATTACK

After running the script, we run the command "dig www.example.net" on behalf of the user. From the below screenshot, we see that the victim machine gets forged reply from the DNS server with Answer section, Authority section and Additional section. The authority section contains the forged responses for "example.net". The additional section contains the forged responses for "ns.example.net" and "attacker32.com", but not for www.facebook.com.

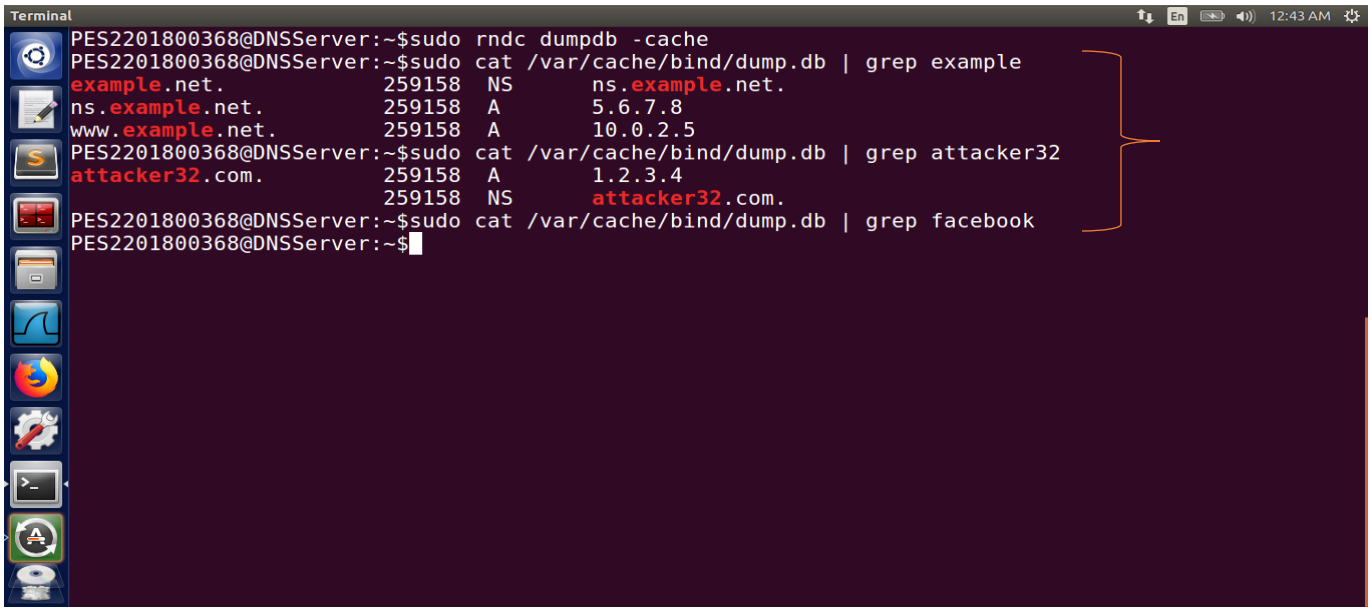


SCREENSHOT SHOWING THE EXECUTION OF THE DIG COMMAND AFTER LAUNCHING THE ATTACK



WIRESHARK CAPTURE OF THE DNS QUERY AND RESPONSE AFTER THE ATTACK

We also provide fake IP addresses for "ns.example.net", "attacker32.com" and www.facebook.com. The two accepted additional records are part of the authority section and hence, the additional information of IP addresses for the two domains are accepted whereas www.facebook.com is out of zone and hence, the record gets discarded.



SCREENSHOT SHOWING THE CONTENTS OF THE CACHE AFTER THE ATTACK