CNS LAB

LAB - 3

Local DNS Attack Lab

NAME : SIRI S

SEMESTER : 5

SECTION :H

SRN : PES1UG19CS485

Lab Setup:

ATTACKER: 10. 0. 2. 4

DNS SERVER: 10. 0. 2. 5

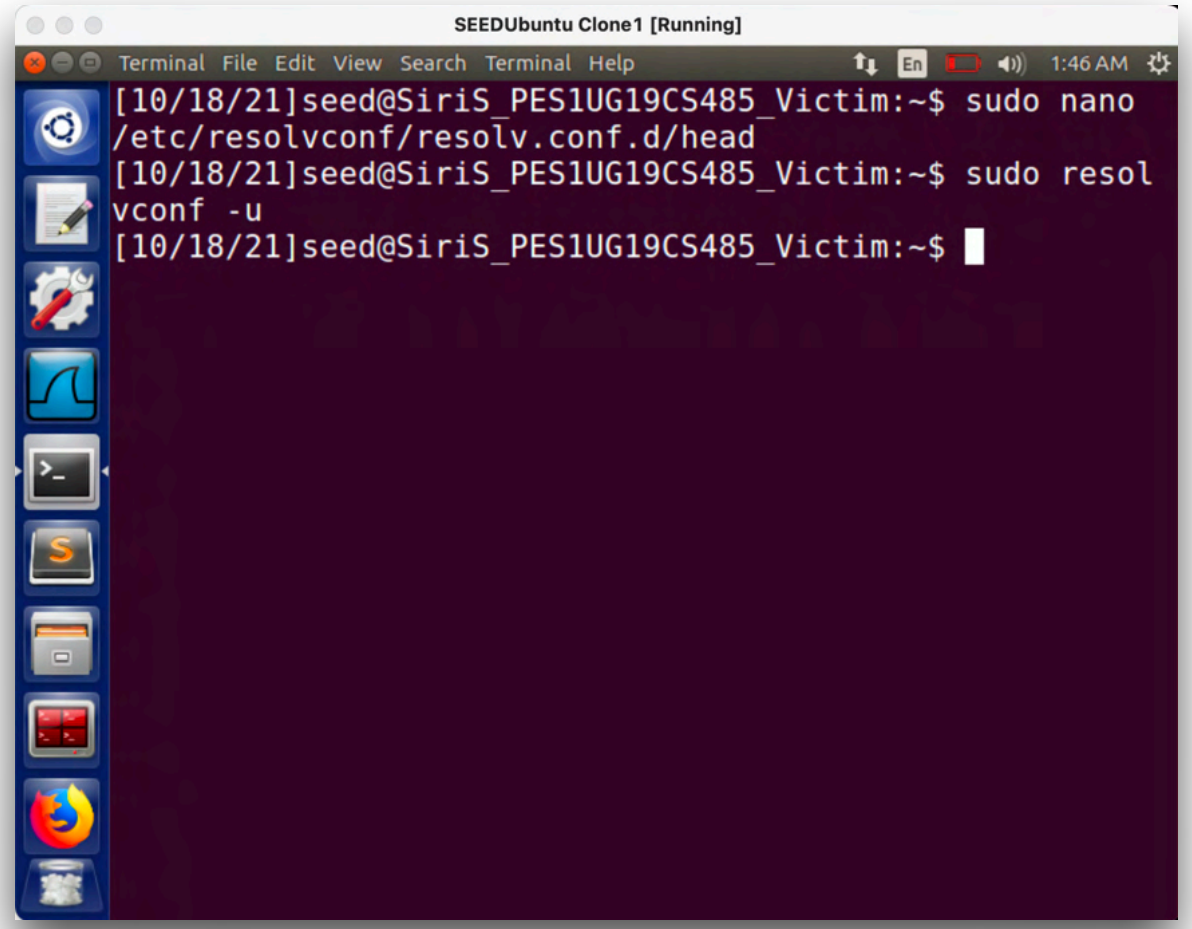
USER/VICTIM: 10. 0. 2. 6

Part I: Setting Up a Local DNS Server

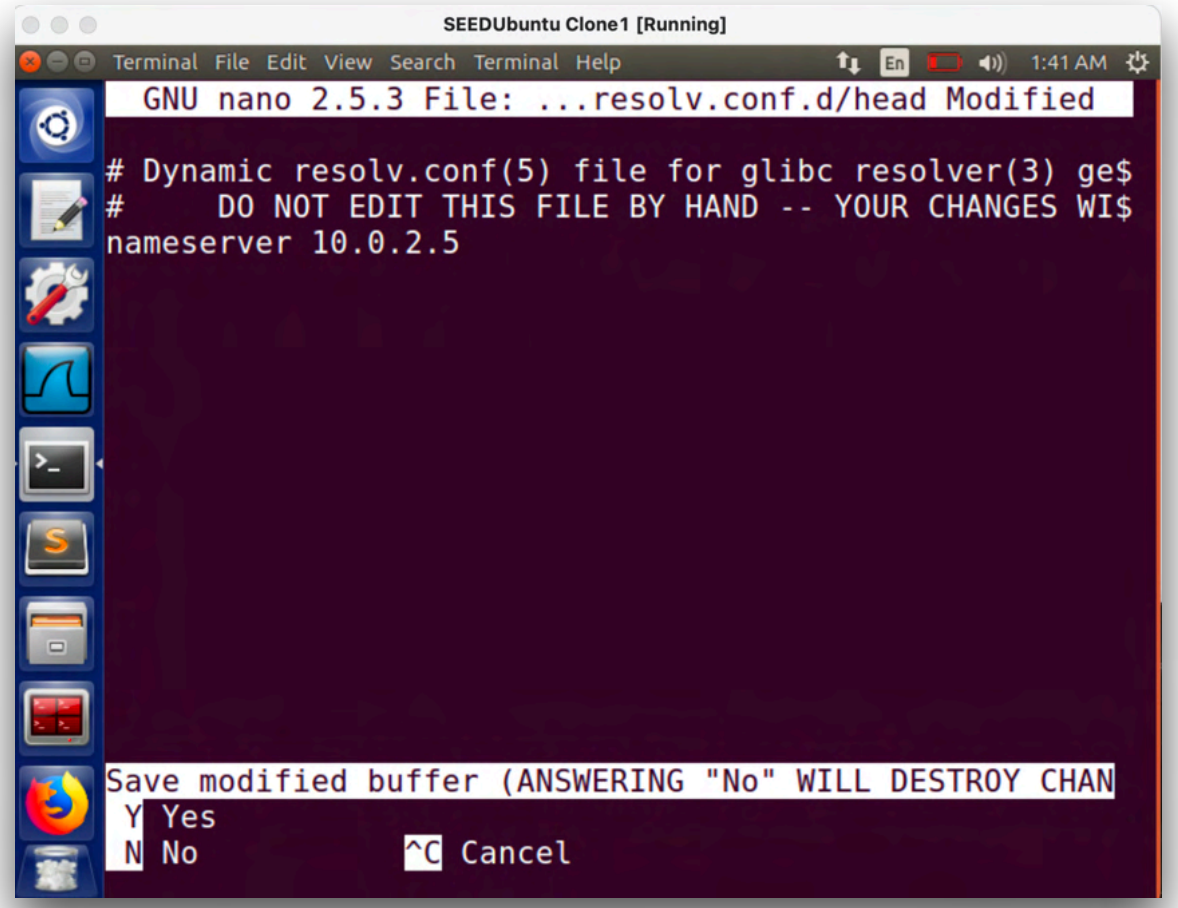
Task 1: Configure the User Machine

Our VM uses the Dynamic Host Configuration Protocol (DHCP) to obtain network configuration parameters, such as IP address, local DNS server, etc. DHCP clients will overwrite the **/etc/resolv.conf** file with the information provided by the DHCP server.

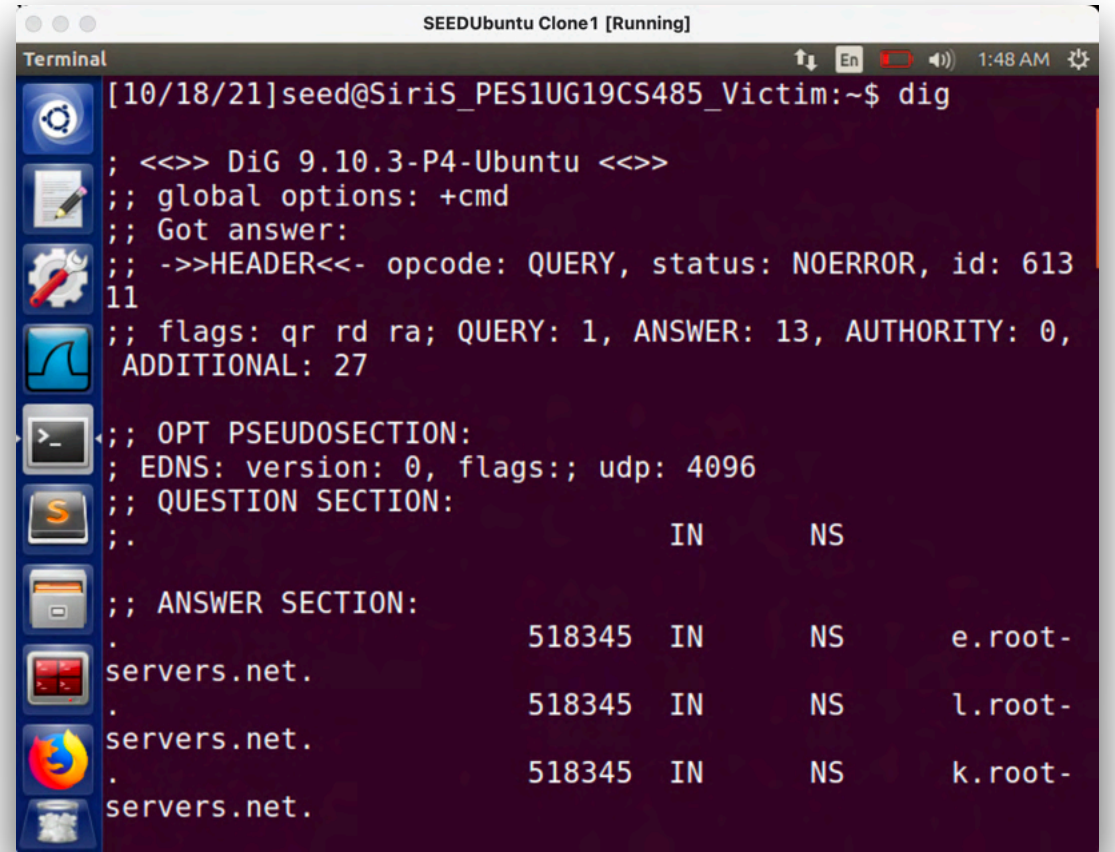
One way to avoid this is by to get our information into **/etc/resolv.conf** without worrying about the DHCP is to add the entry to the **/etc/resolvconf/resolv.conf.d/head** file:

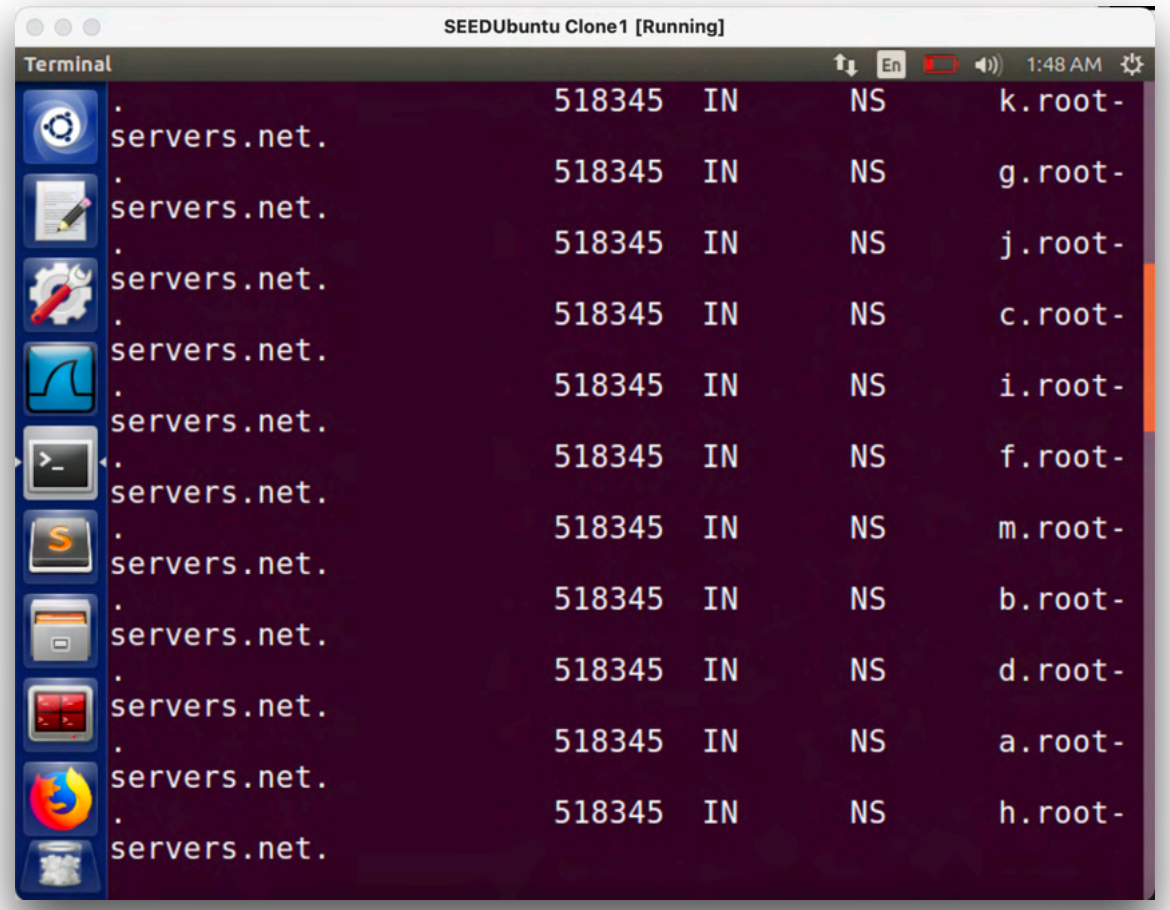


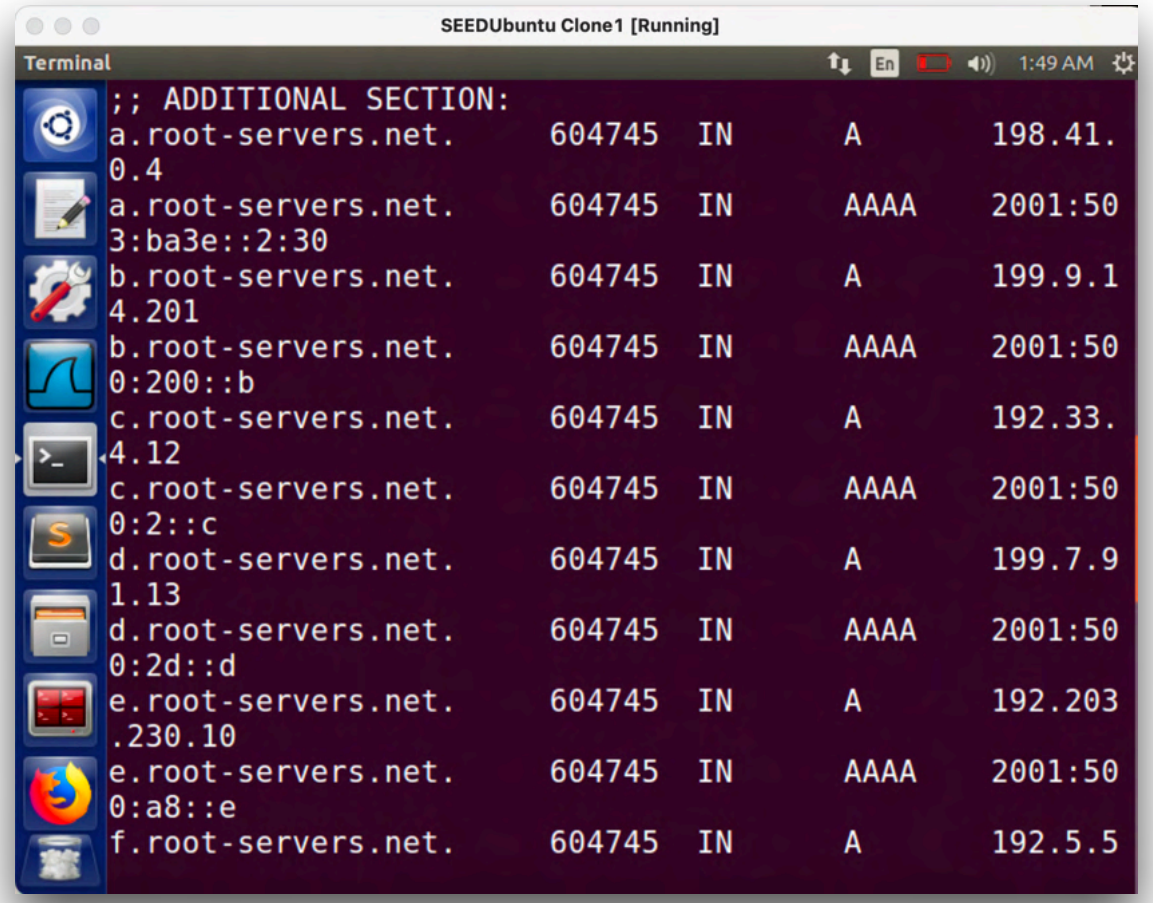
Entering the IP address of DNS server into the user:

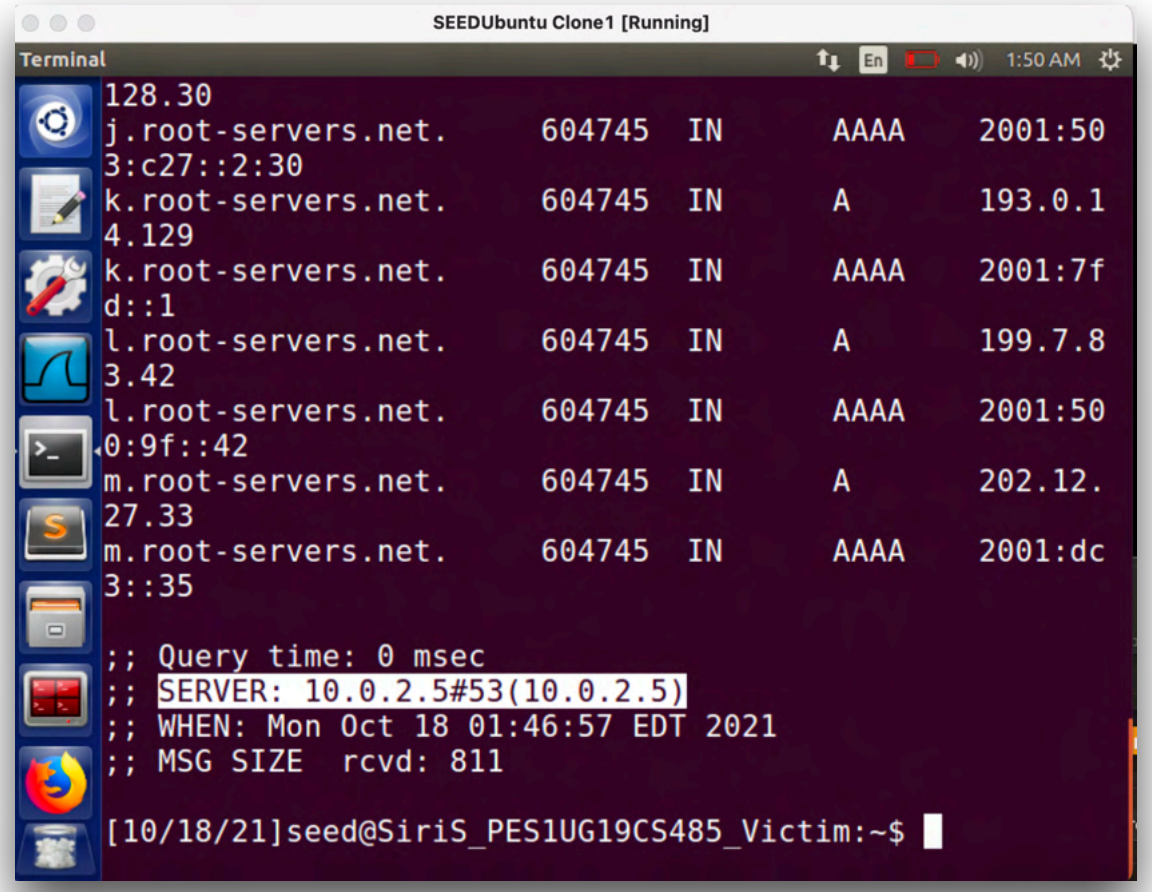


Using *dig* command to provide evidences to show that the response is indeed from the server:



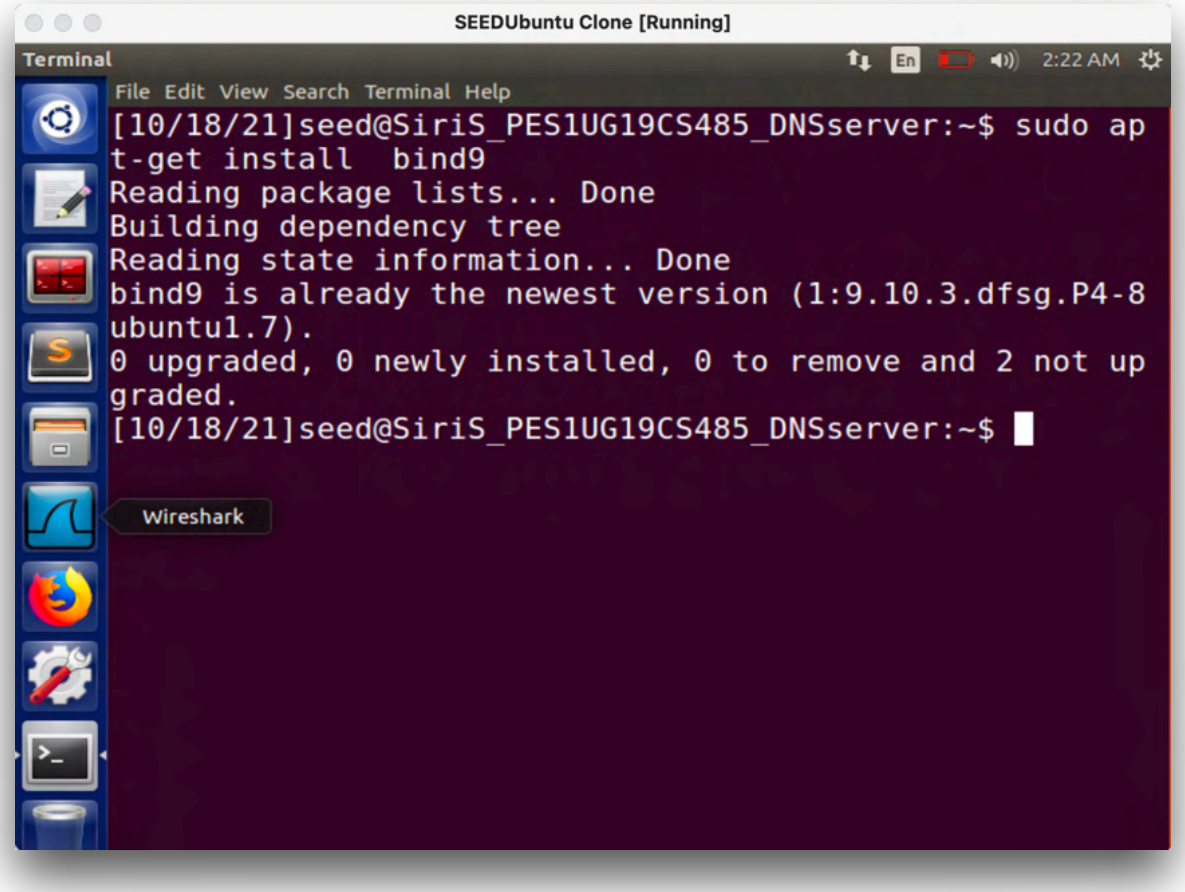

SEEDUbuntu Clone1 [Running]
Terminal
En
1:48 AM
. 518345 IN NS k.root-servers.net.
. 518345 IN NS g.root-servers.net.
. 518345 IN NS j.root-servers.net.
. 518345 IN NS c.root-servers.net.
. 518345 IN NS i.root-servers.net.
. 518345 IN NS f.root-servers.net.
. 518345 IN NS m.root-servers.net.
. 518345 IN NS b.root-servers.net.
. 518345 IN NS d.root-servers.net.
. 518345 IN NS a.root-servers.net.
. 518345 IN NS h.root-servers.net.


SEEDUbuntu Clone1 [Running]
Terminal
En
1:49 AM
;; ADDITIONAL SECTION:
a.root-servers.net. 604745 IN A 198.41.0.4
a.root-servers.net. 604745 IN AAAA 2001:503:ba3e::2:30
b.root-servers.net. 604745 IN A 199.9.14.201
b.root-servers.net. 604745 IN AAAA 2001:500:200::b
c.root-servers.net. 604745 IN A 192.33.4.12
c.root-servers.net. 604745 IN AAAA 2001:500:2::c
d.root-servers.net. 604745 IN A 199.7.91.13
d.root-servers.net. 604745 IN AAAA 2001:500:2d::d
e.root-servers.net. 604745 IN A 192.203.230.10
e.root-servers.net. 604745 IN AAAA 2001:500:a8::e
f.root-servers.net. 604745 IN A 192.5.5



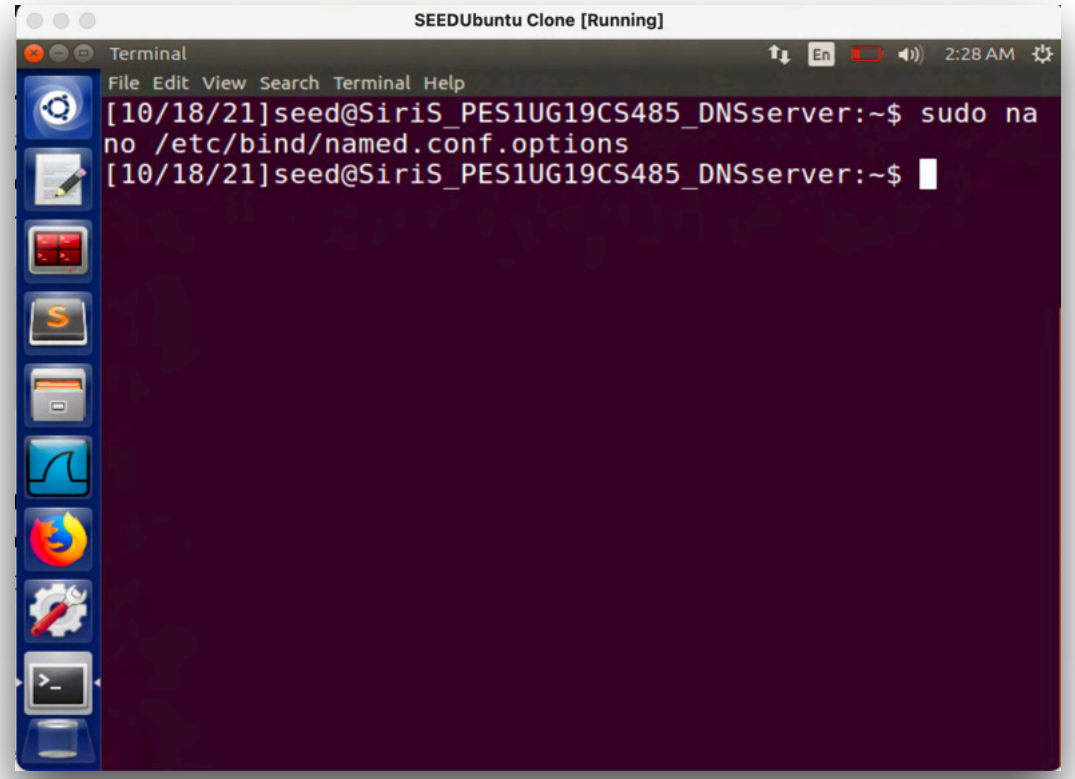
<u>Task 2:</u> Set Up a Local DNS Server

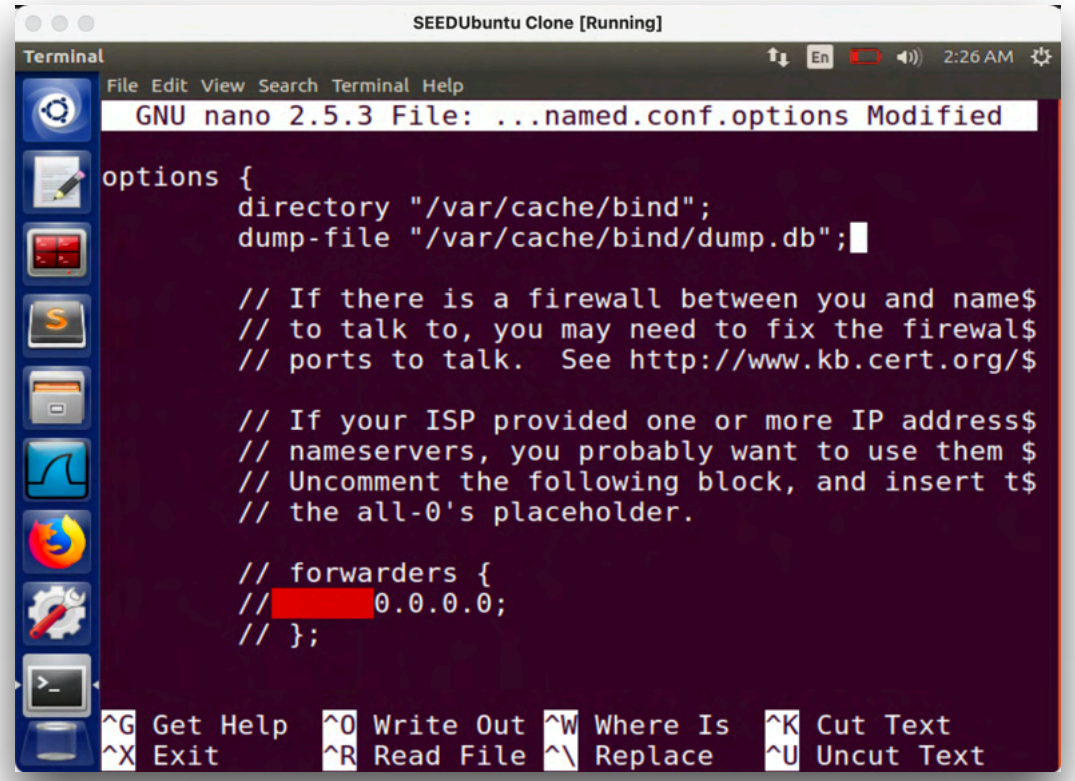
The most widely used DNS server software is called BIND (Berkeley Internet Name Domain). The BIND 9 server program is already installed in our pre-built Ubuntu VM image.



<u>Step 1:</u> Configure the BIND 9 server.

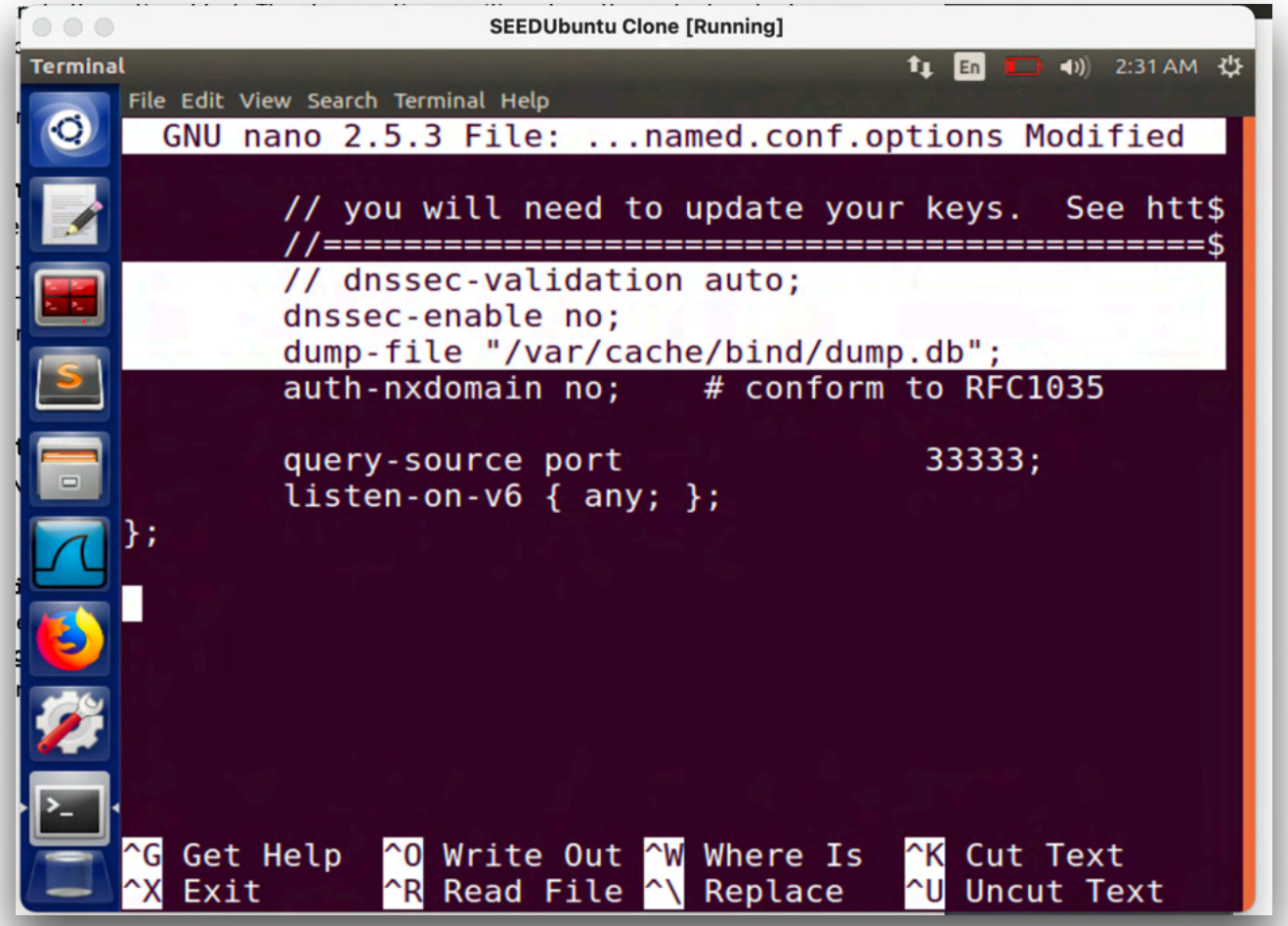
This specifies where the cache content should be dumped to if BIND is asked to dump its cache. If this option is not specified, BIND dumps the cache to a default file called **/var/cache/bind/ named_dump.db.**





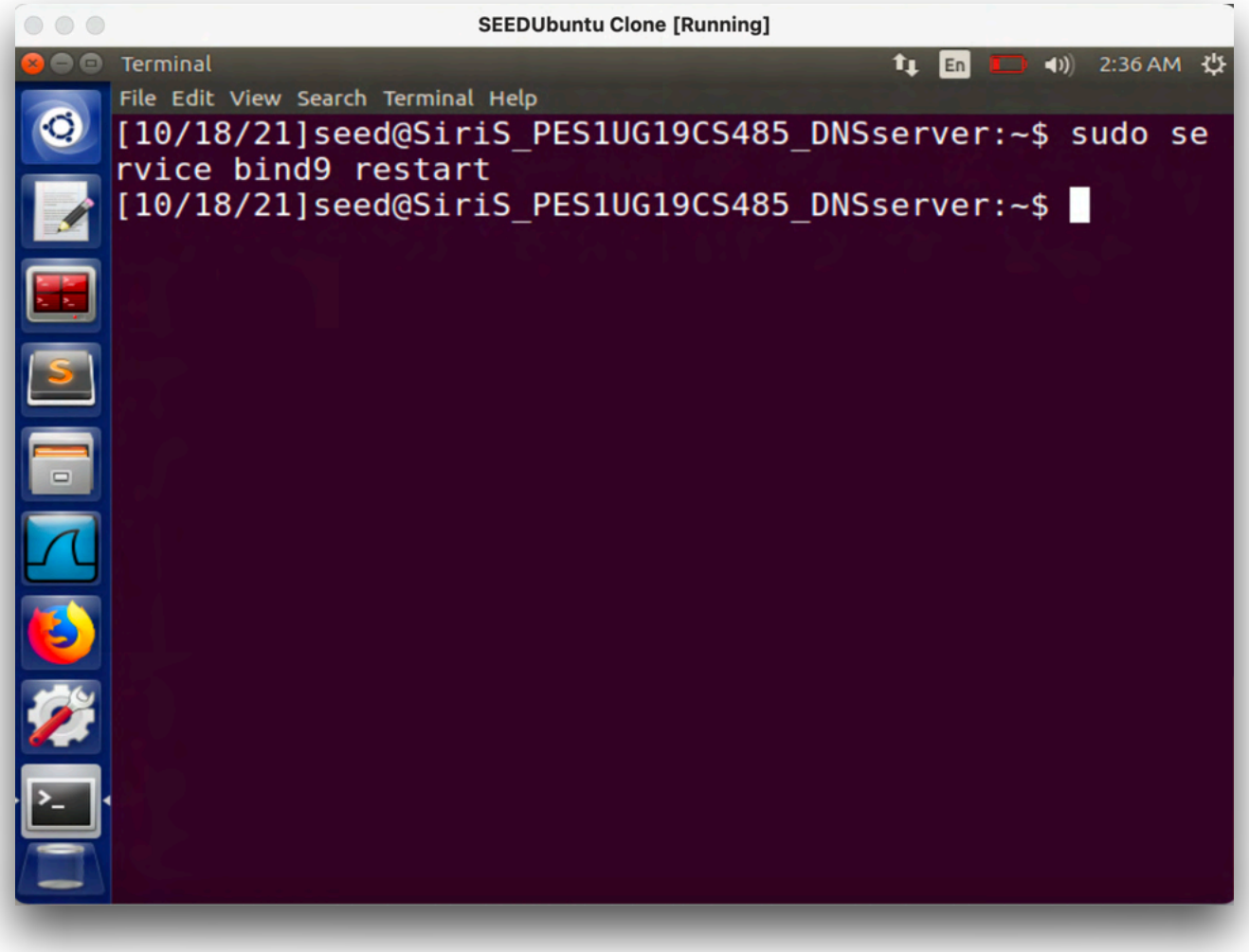
Step 2: Turn off DNSSEC.

DNSSEC is introduced to protect against spoofing attacks on DNS servers. To show how attacks work without this protection mechanism, we need to turn the protection off.



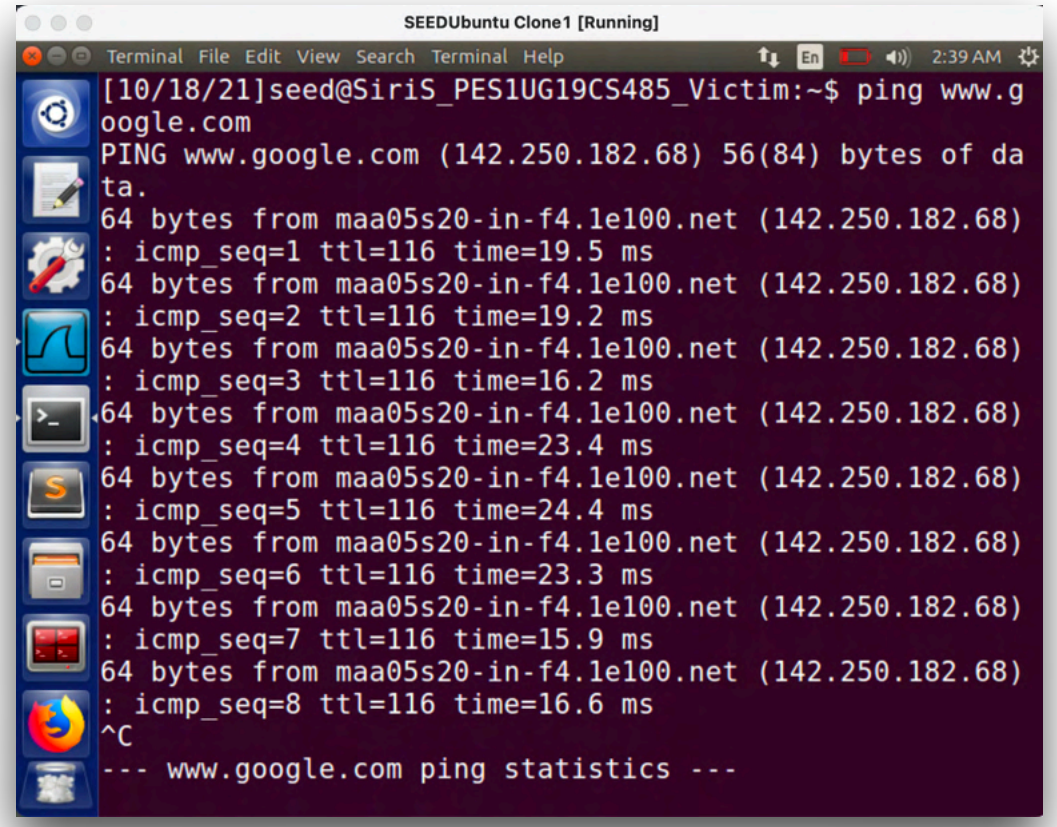
<u>Step 3:</u> Start DNS server.

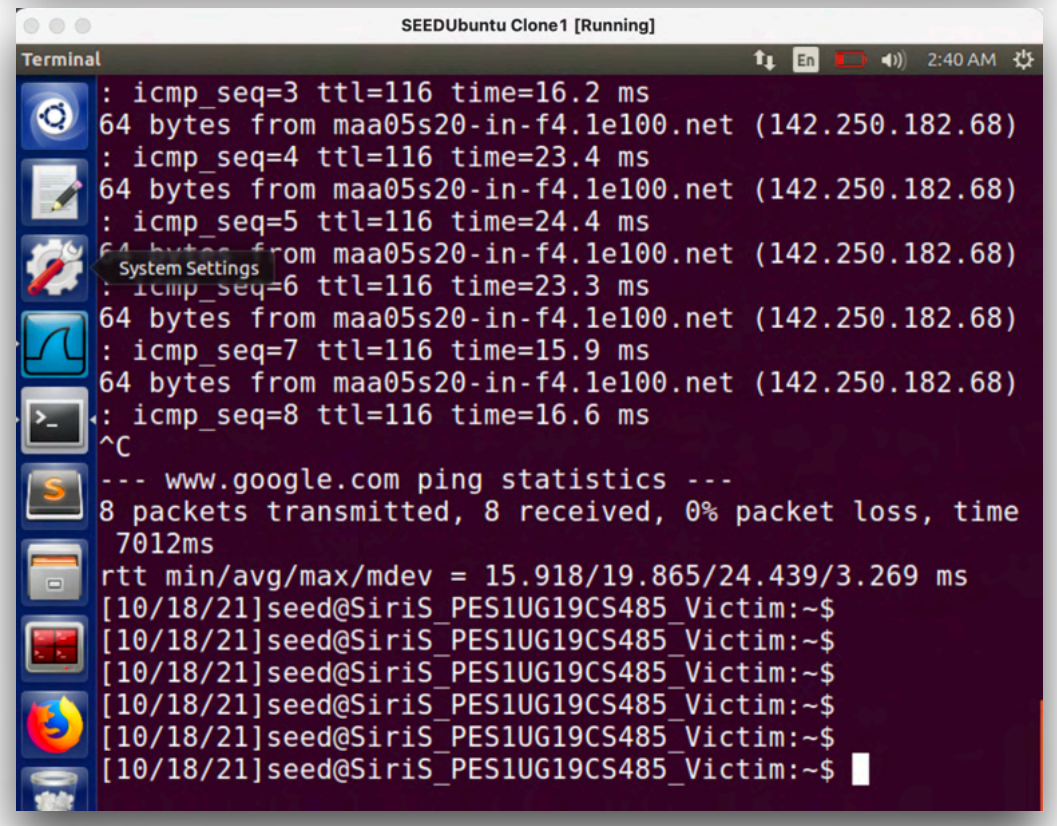
Every time a modification is made to the DNS configuration, the DNS server needs to be restarted.



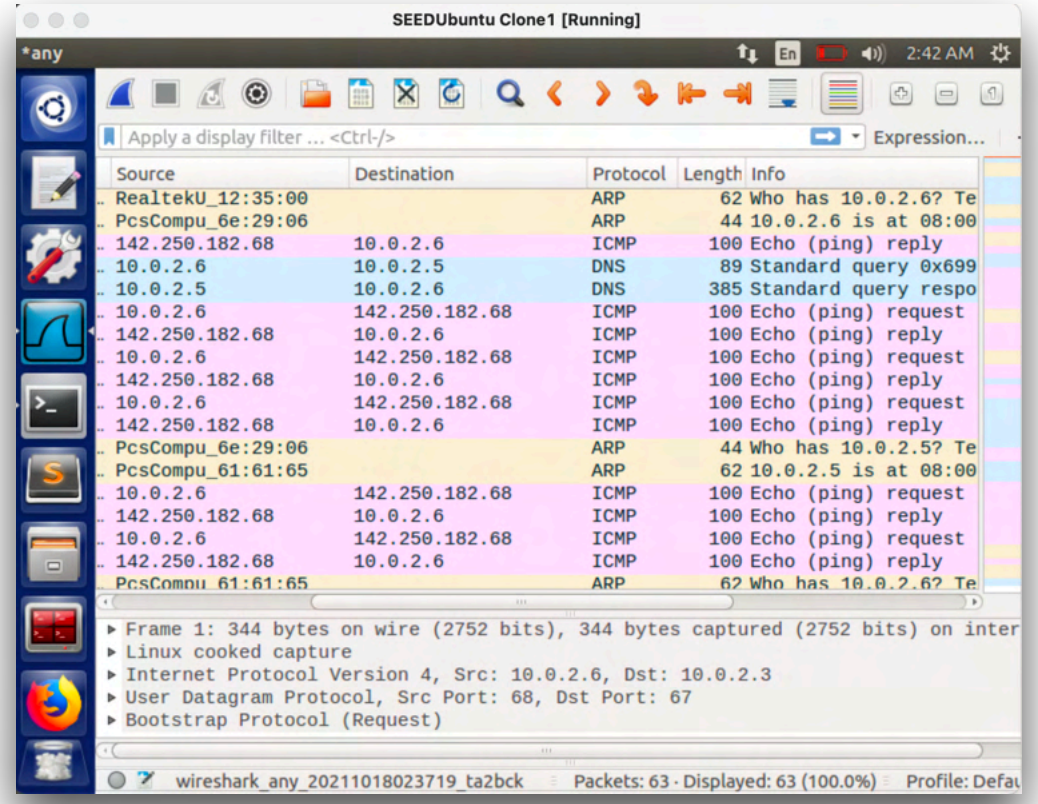
Step 4: Use the DNS server.

Pinging www.google.com from the victim machine (10. 0. 2. 6) so we can see that it sends a DNS query to the DNS server (10. 0. 2. 5) and the DNS server further sends the query to the root, TLD and nameservers.



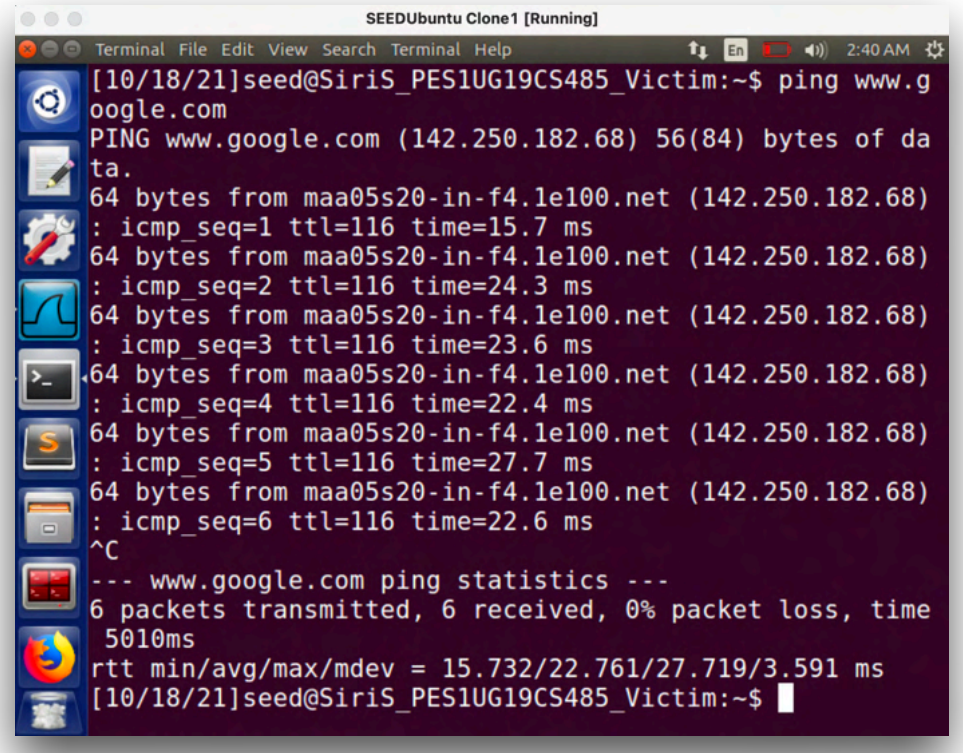


Wireshark screenshot for the first ping:

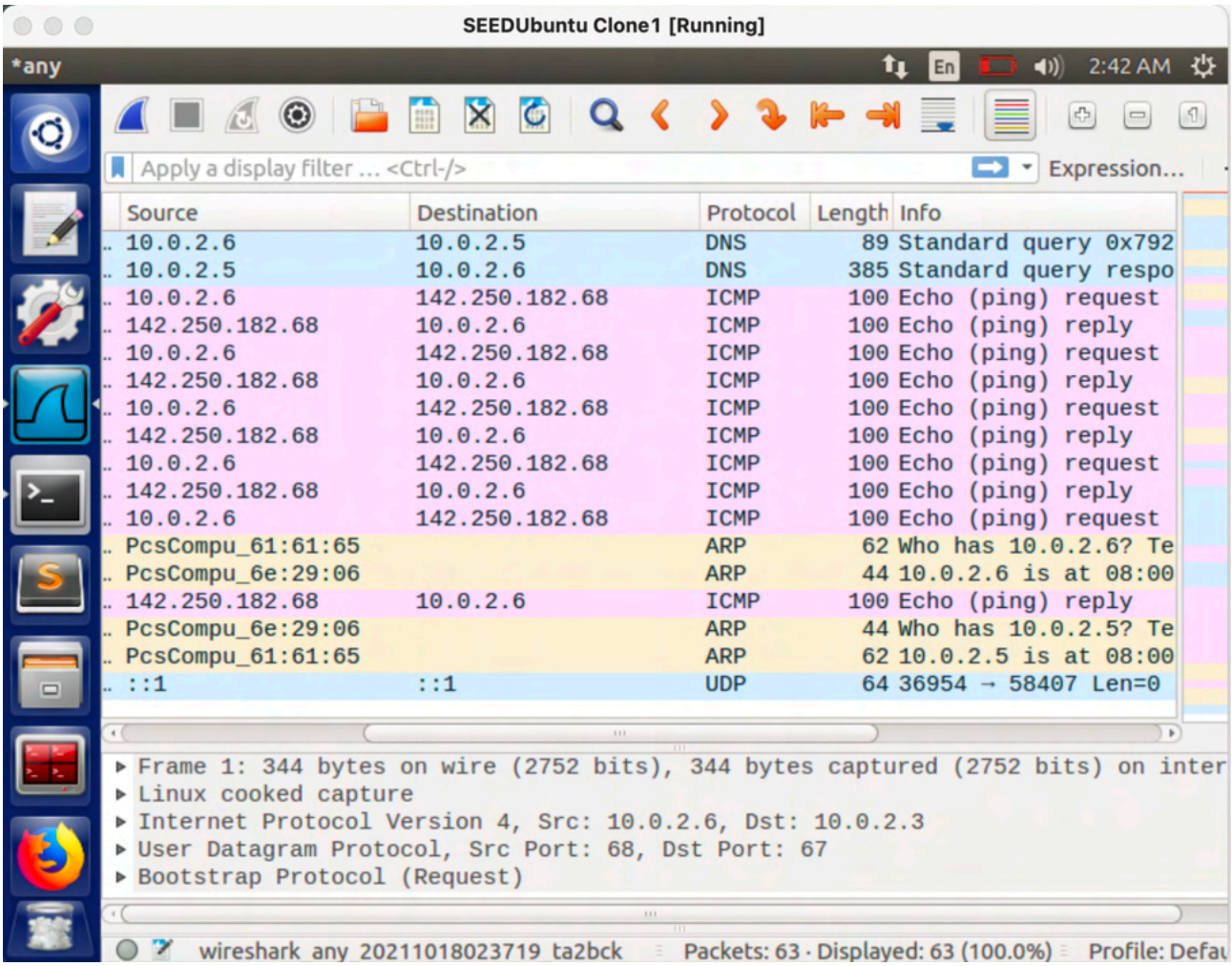


We can see from the screenshot that the Victim is sending a standard DNS query to the DNS server which is further sending an ICMP packet.

Upon pinging the same host again, we will see that on receiving the DNS query request, the DNS server will send a DNS reply with the IP address of the host. It shows that the DNS server has cached the information about the root servers, TLDs and nameservers.



Wireshark screenshot for the second ping:

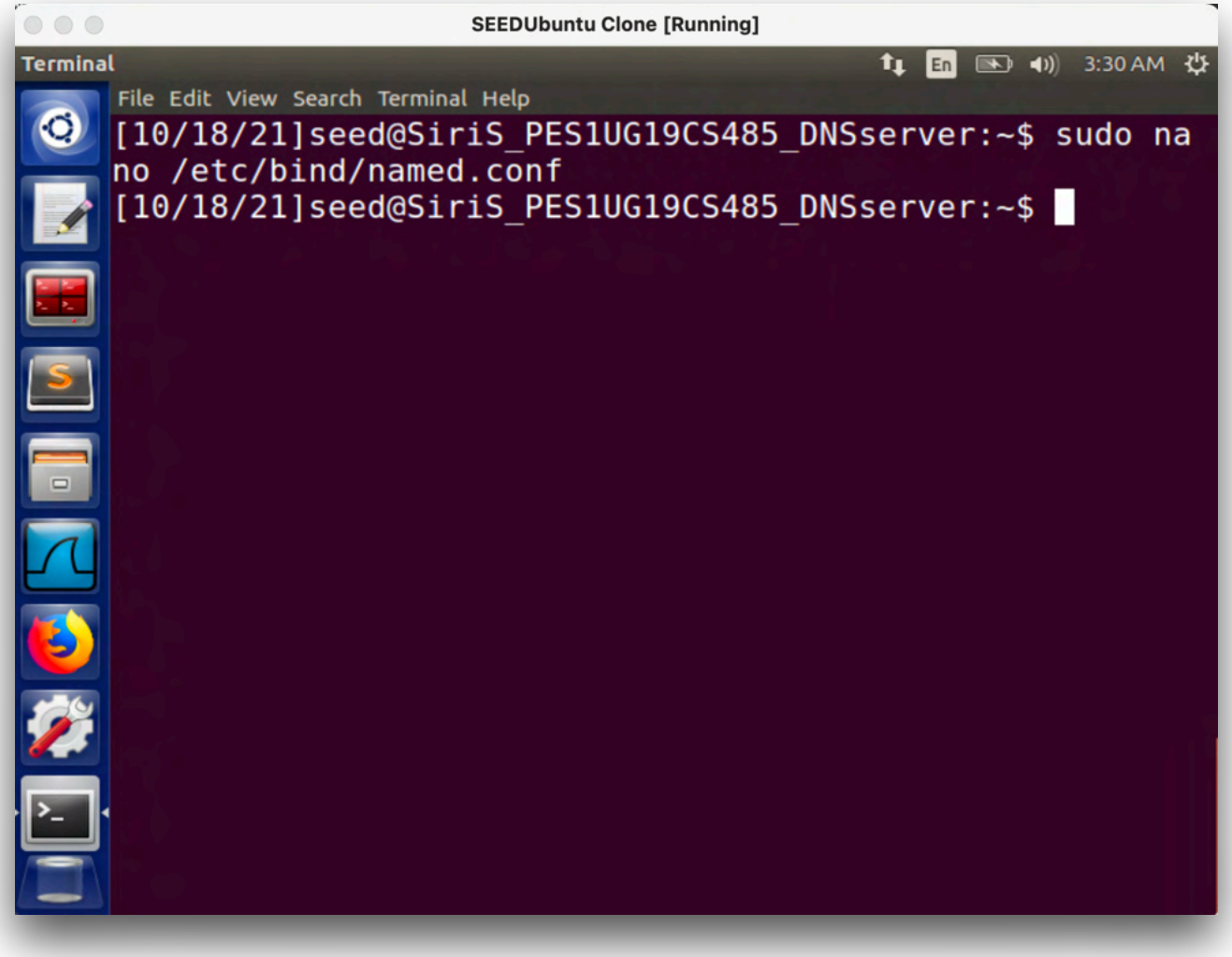


<u>Task 3:</u> Host a Zone in the Local DNS Server

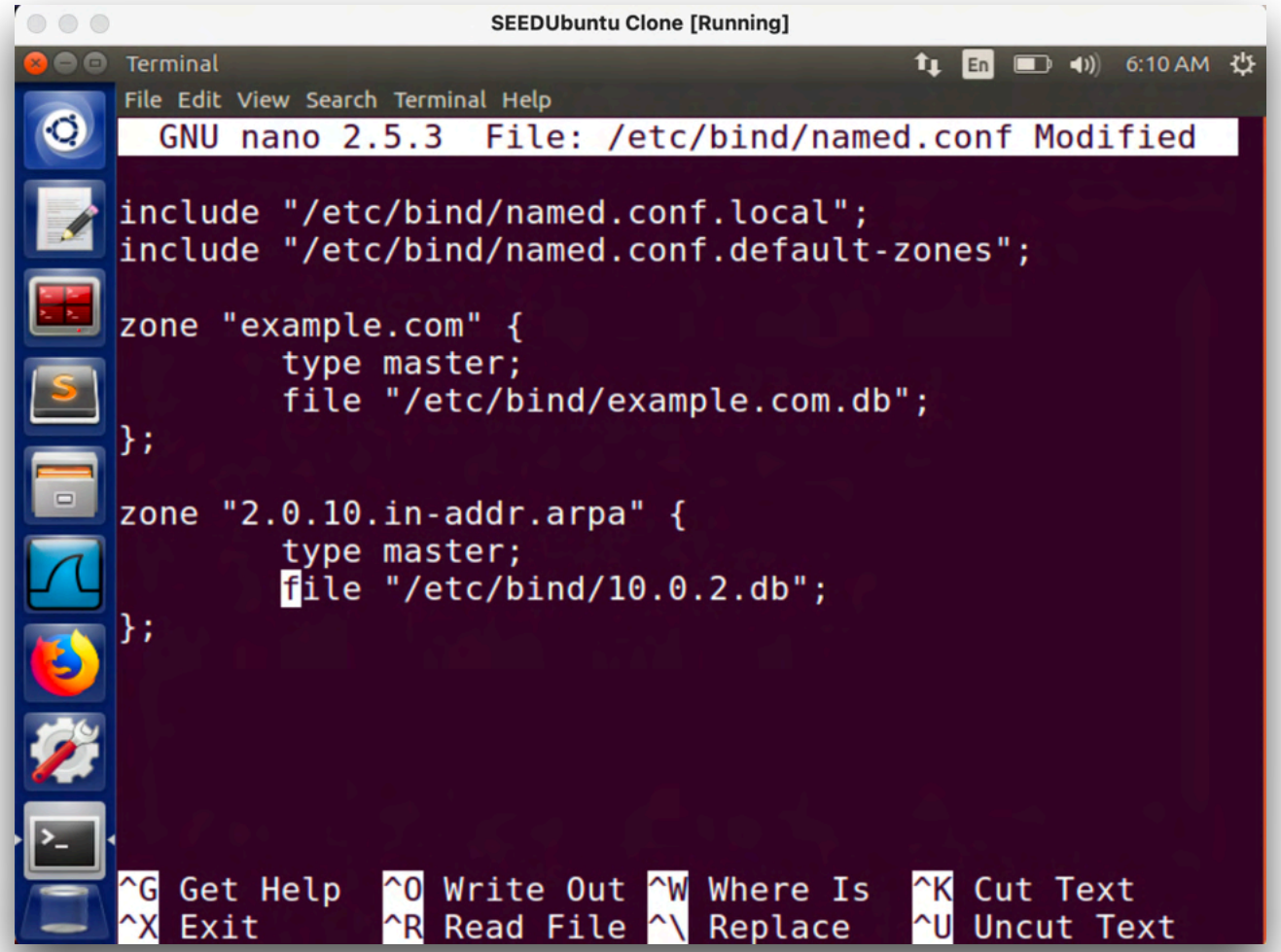
Setting up an authoritative server for the example.com domain. This domain name is reserved for use in documentation, and is not owned by anybody, so it is safe to use it.

<u>Step 1:</u> Create zones.

Creating two zone entries in the DNS server by adding the following contents to **/etc/bind/named.conf**.
The first zone - forward lookup (from hostname to IP)
The second zone - reverse lookup (from IP to hostname).

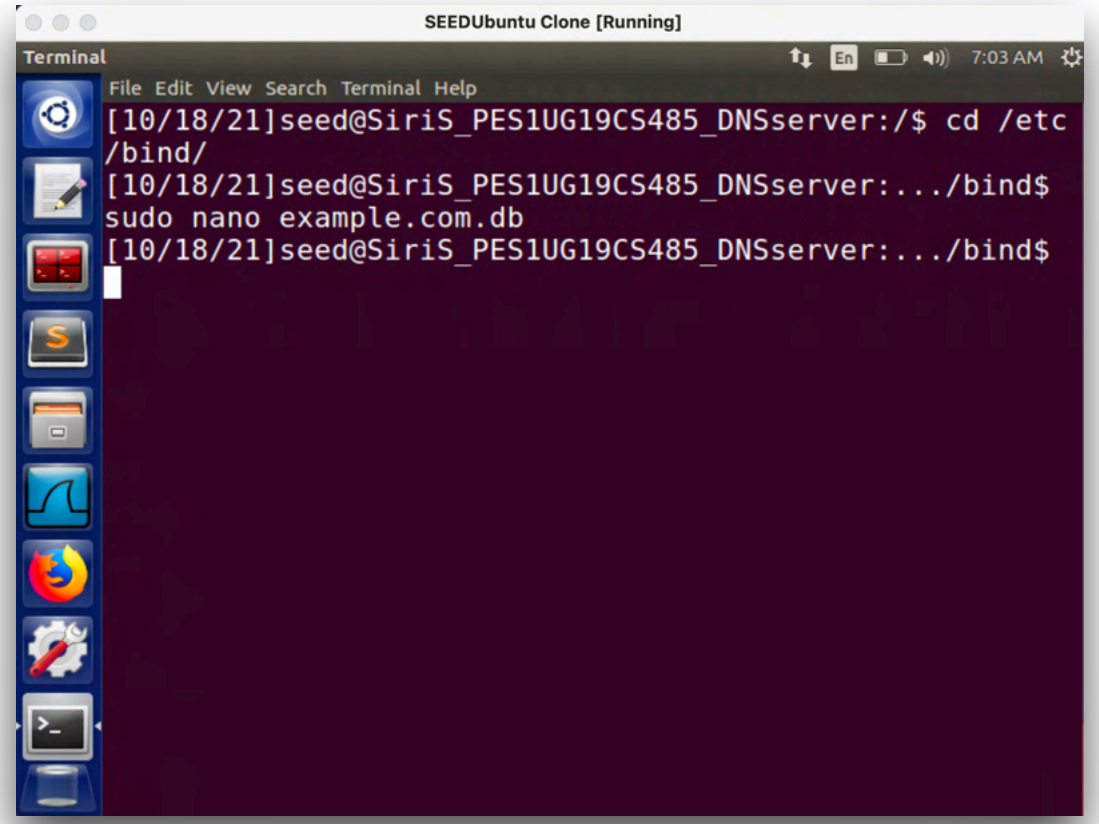


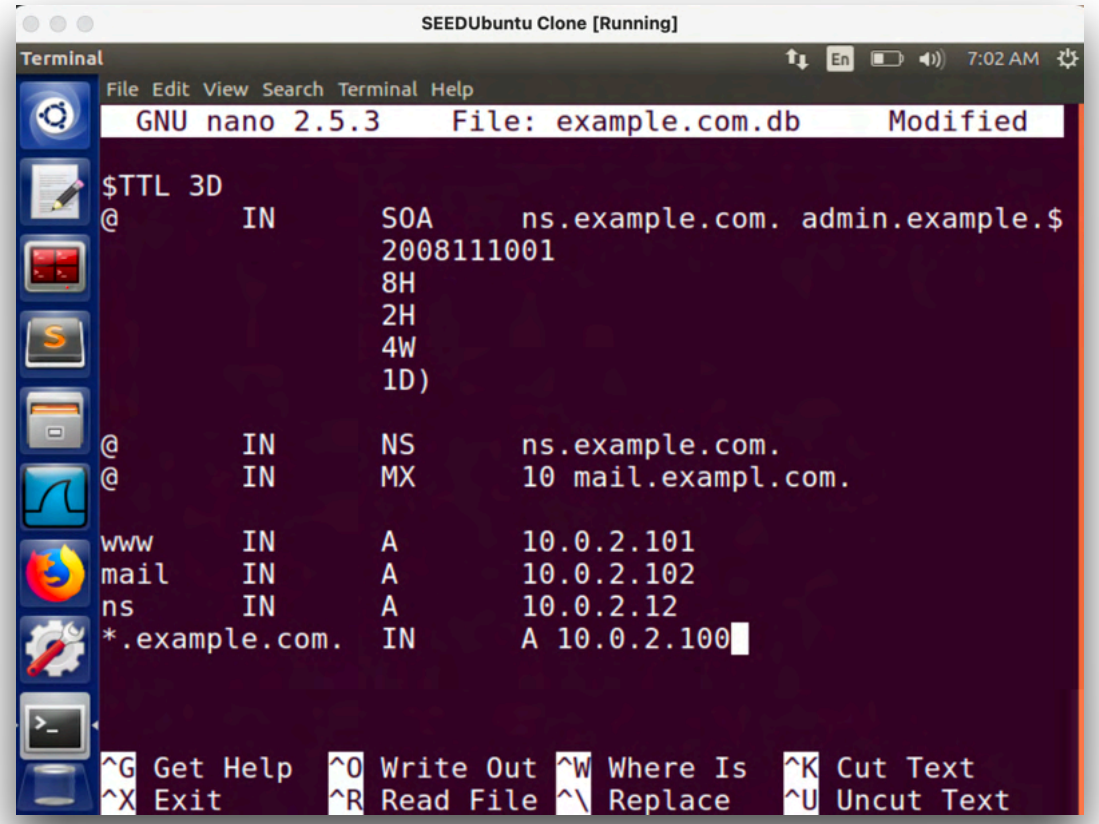
Adding the two zones into named.conf



Step 2: Setup the forward lookup zone file

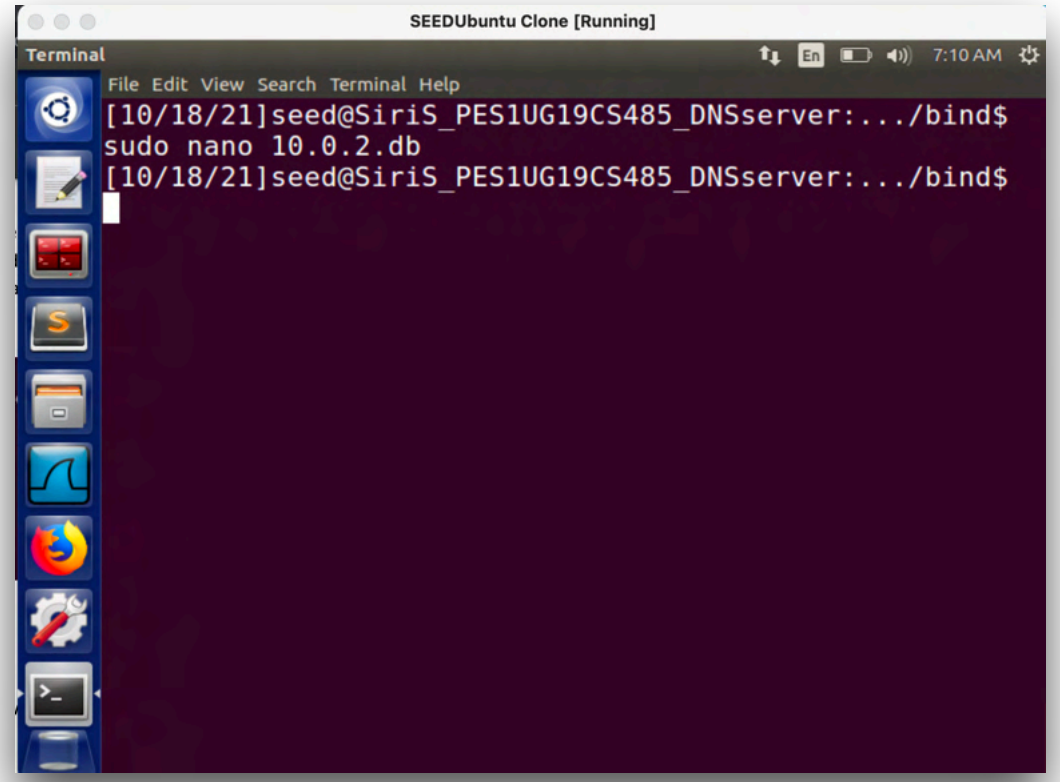
We create an **example.com.db** zone file with the following contents in the **/etc/bind/** directory where the actual DNS resolution is stored.

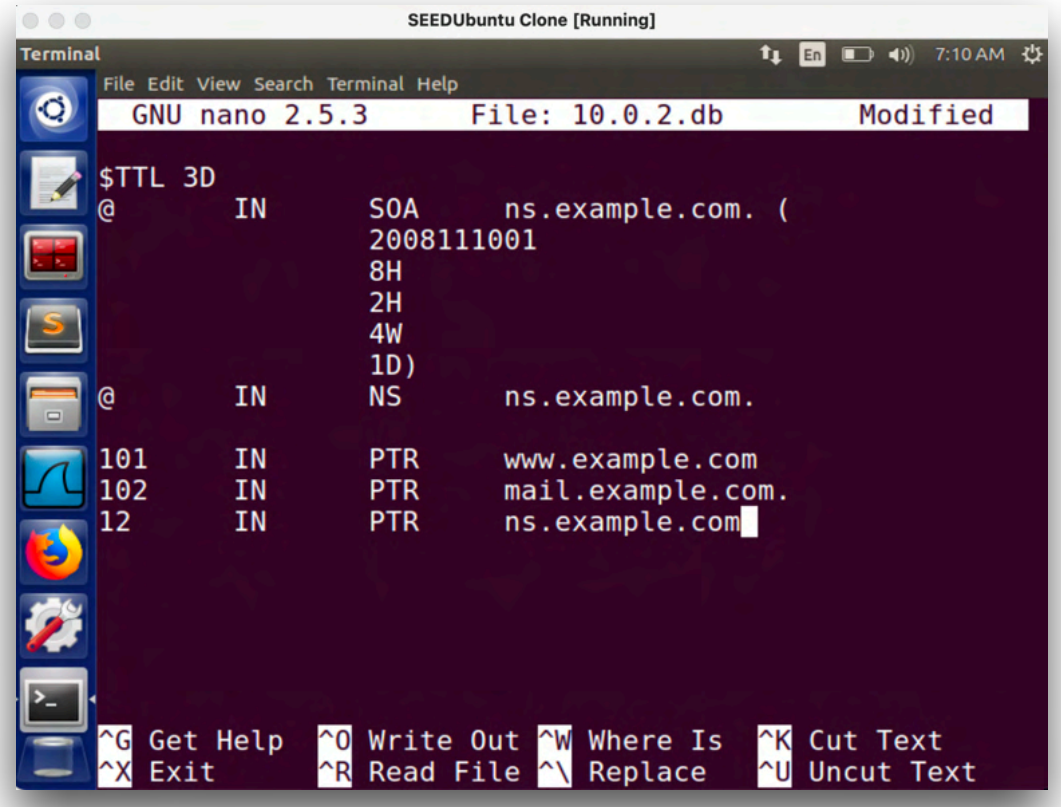




<u>Step 3:</u> Setup the reverse lookup zone file.

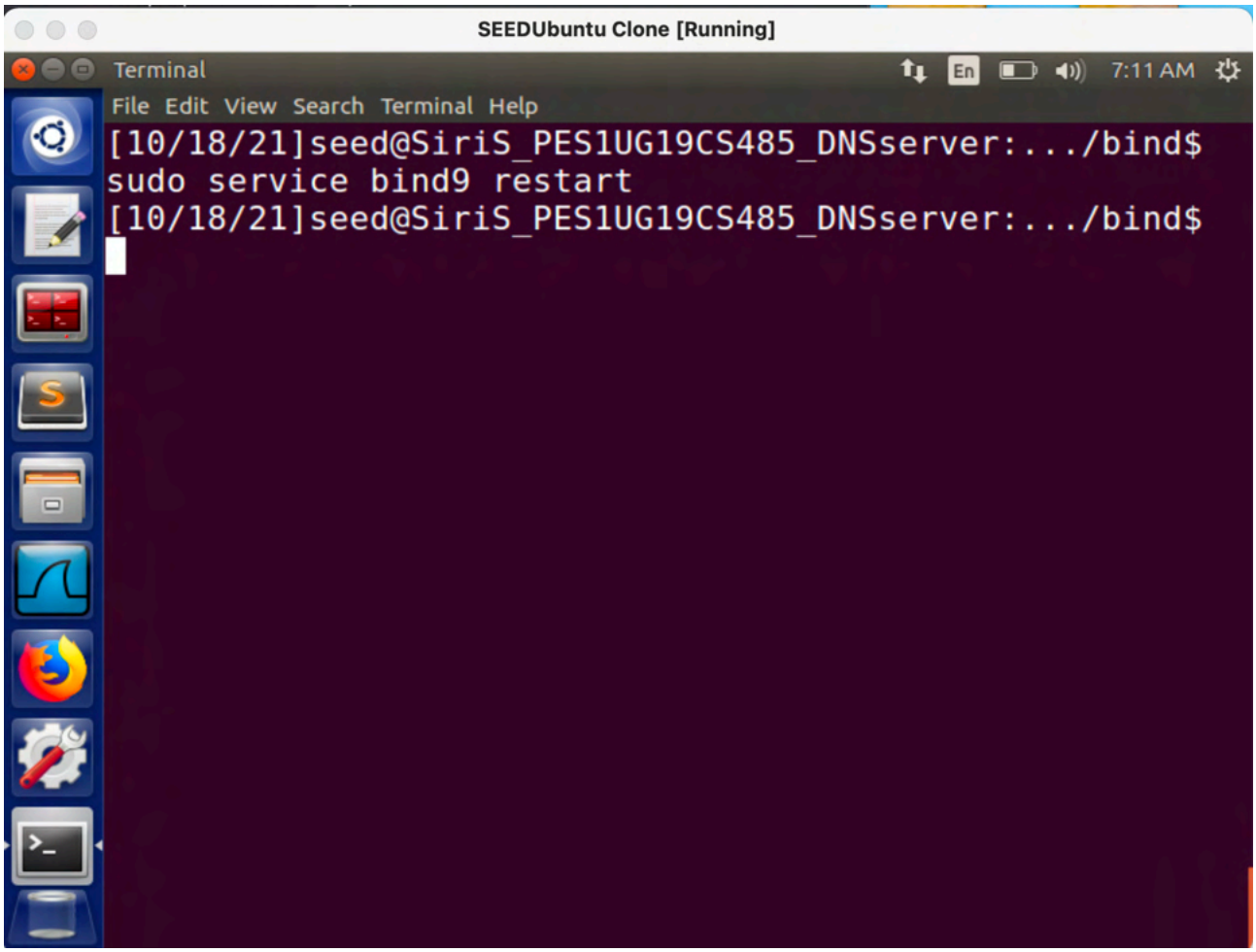
We create a reverse DNS lookup file called **10.0.2.db** for the example.net domain to support DNS reverse lookup, i.e., from IP address to hostname in the **/etc/bind/** directory with the following contents.



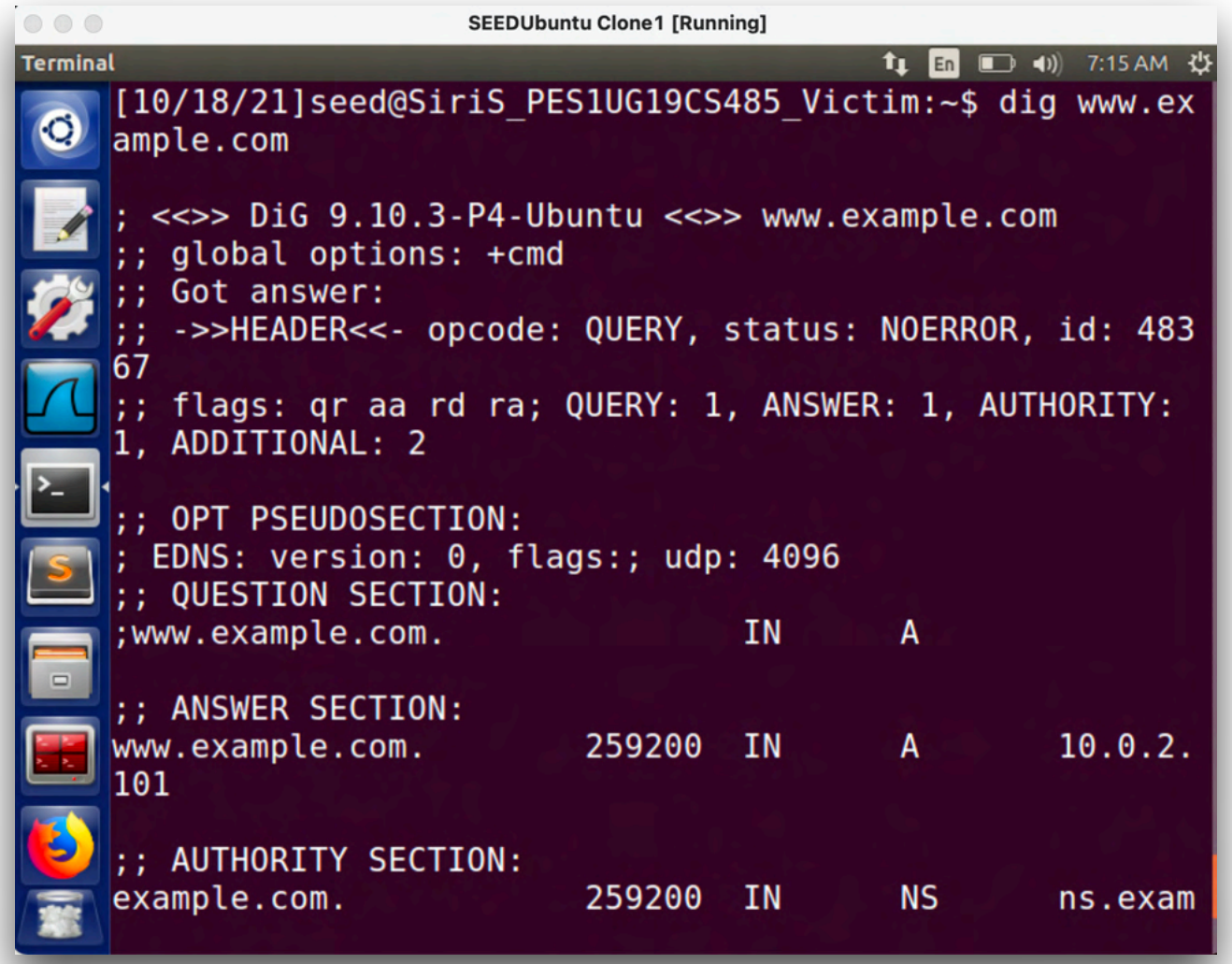


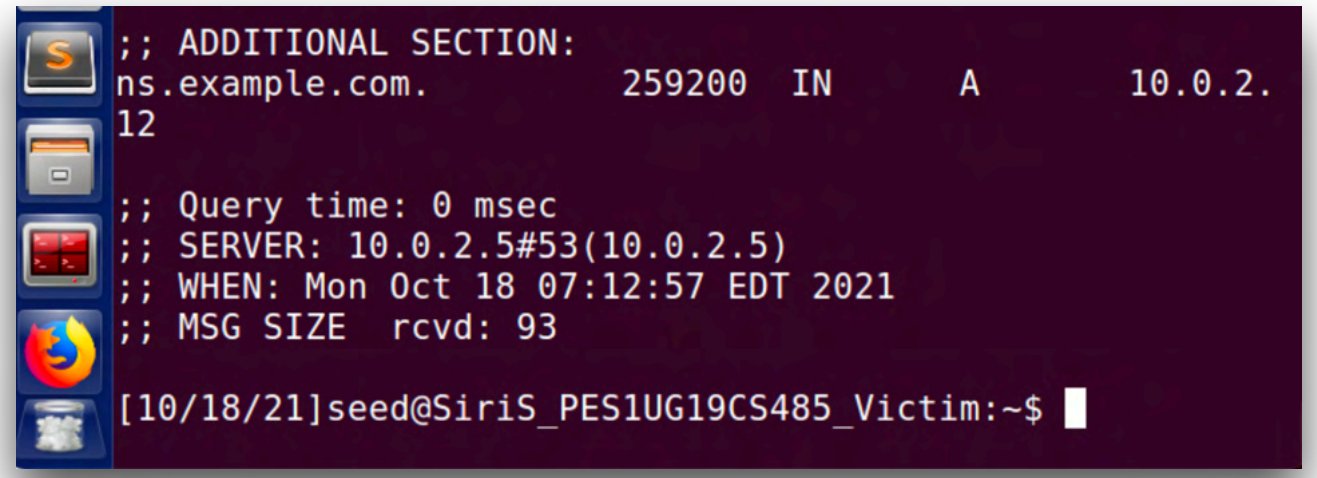
<u>Step 4:</u> Restart the BIND server and test.

Every time a modification is made to the DNS configuration, the DNS server needs to be restarted.



By using the *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 set up in the DNS server.

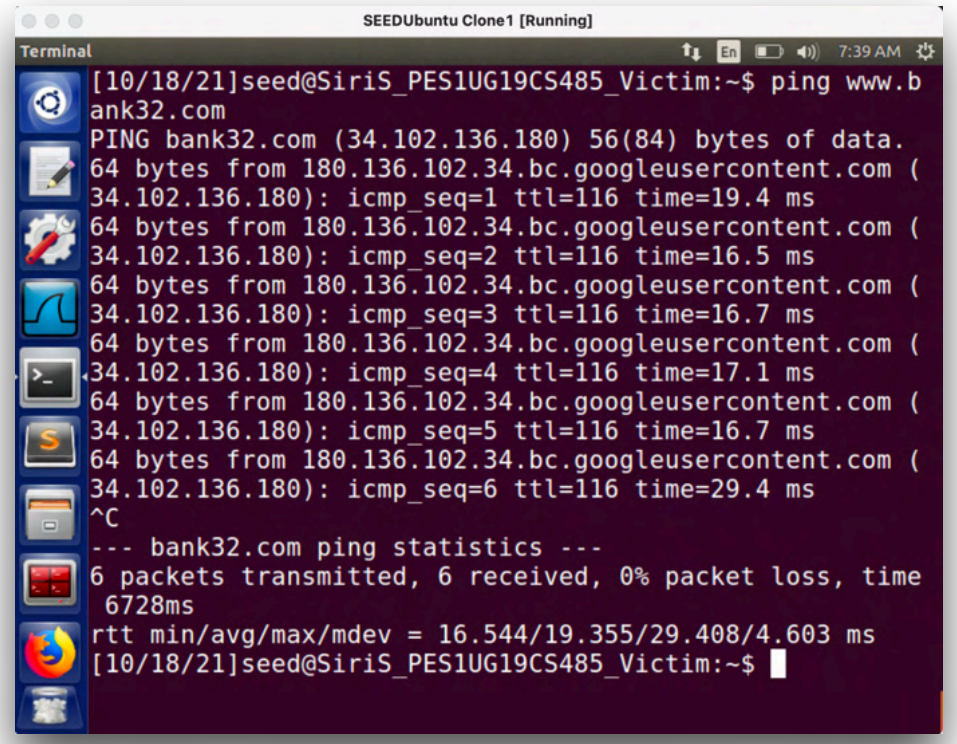


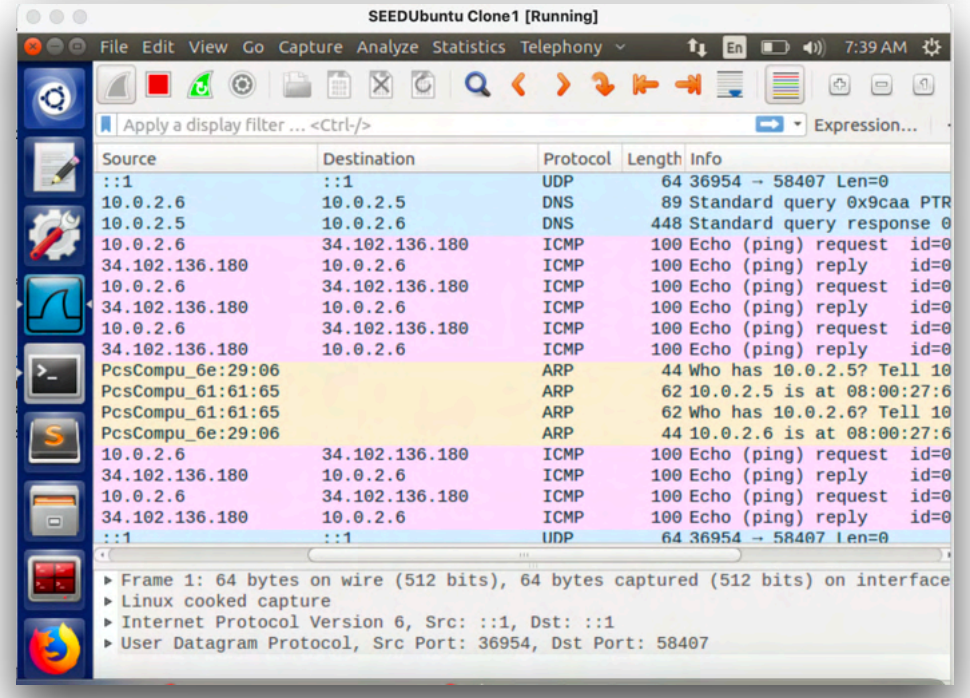


Part II: Attacks on DNS

Task 4: Modifying the Host File

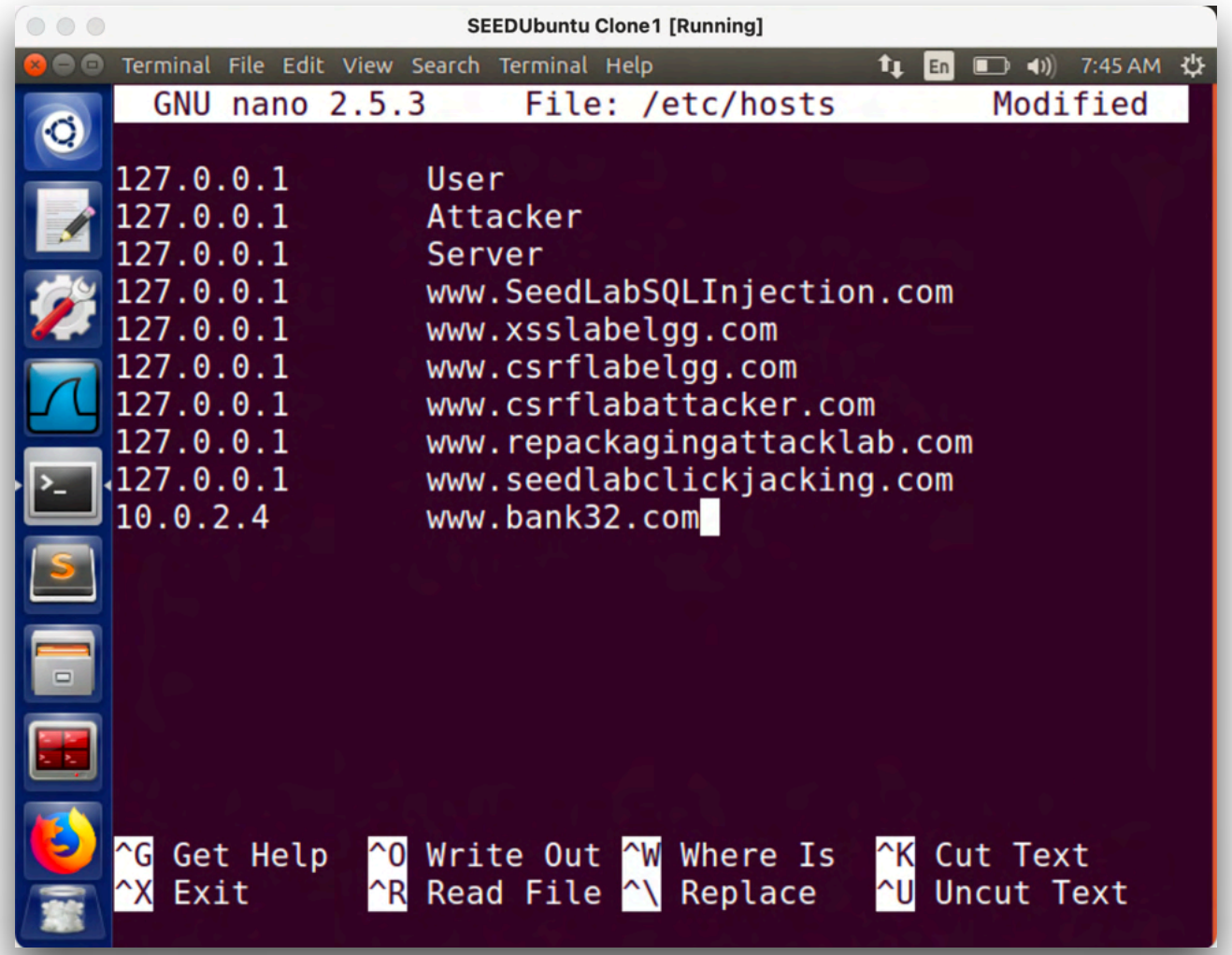
Before the attack:



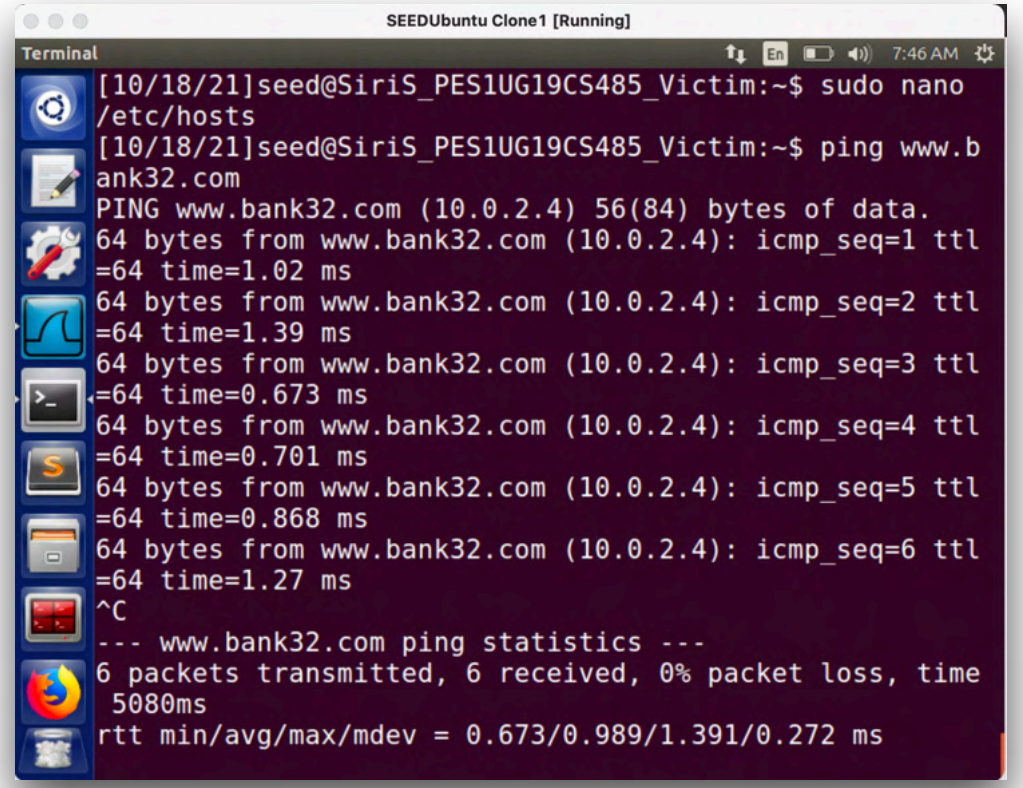


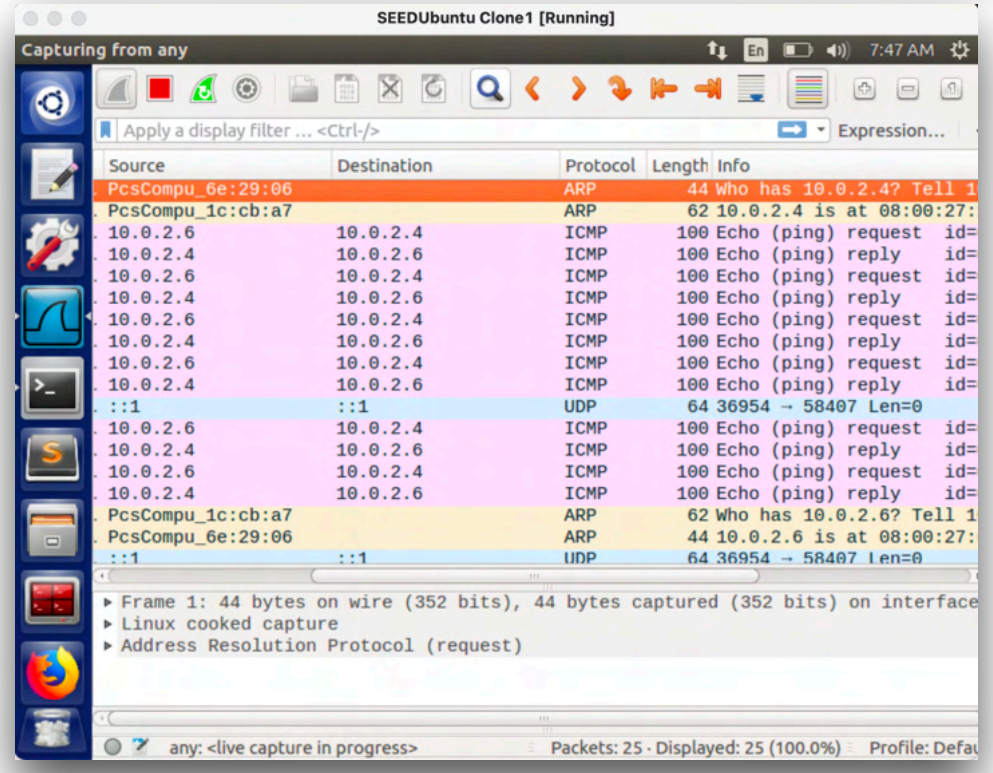
We are getting an ICMP response back from www.bank32.com

Attack: We modify the **/etc/hosts** file with the IP address of www.bank32.com as 10. 0. 2. 4 (Attacker machine).



After the attack:

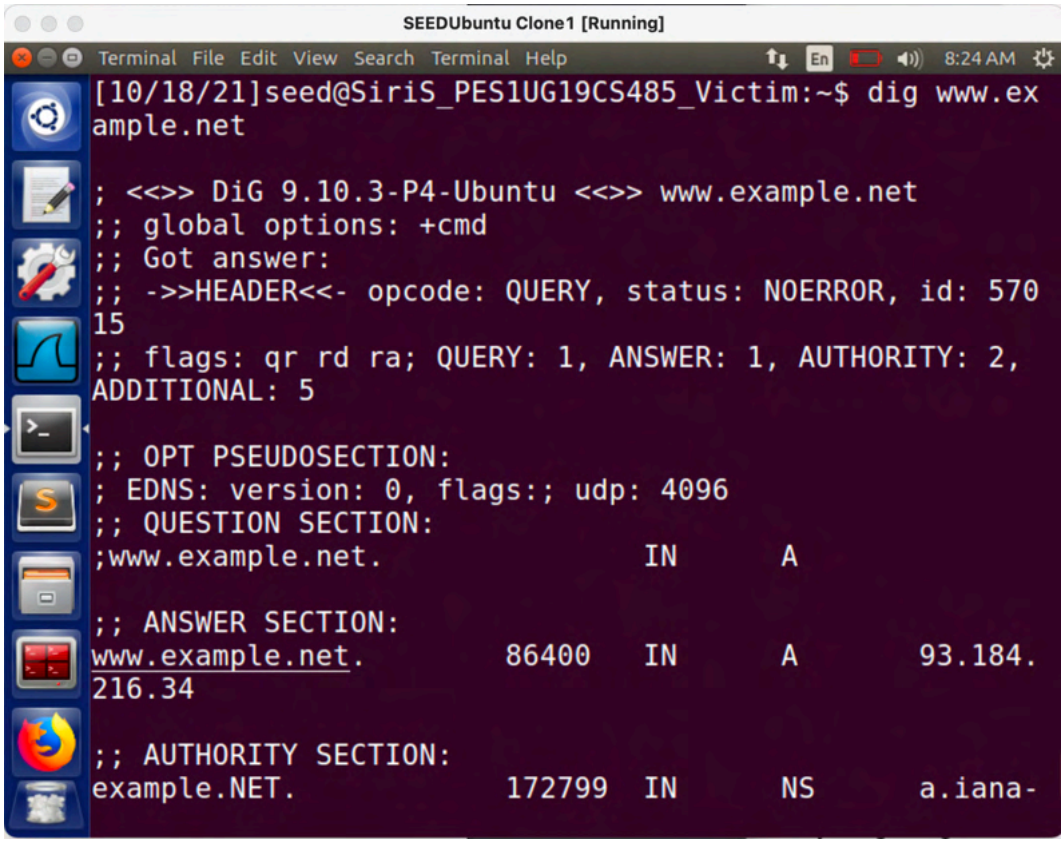


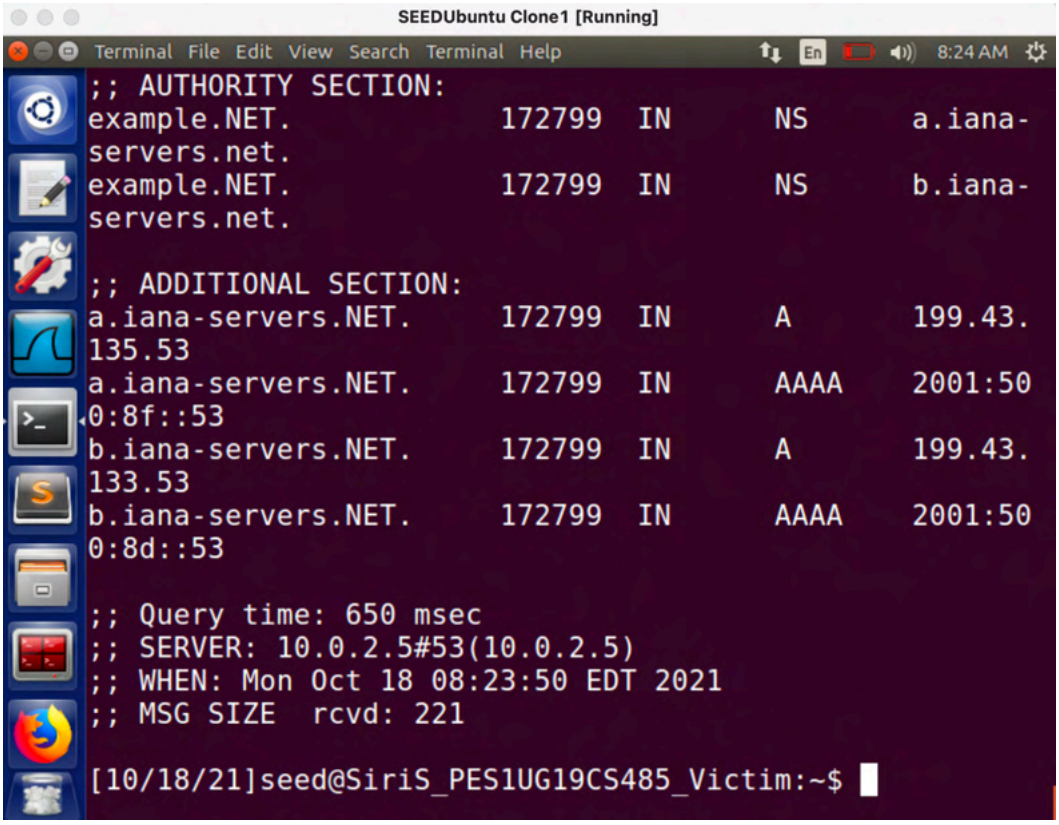


We get ICMP response back from the “10. 0. 2. 4” (Attacker machine) as well as the browser gives the default page of the Apache server running in the Attacker machine. If the file is compromised the attacker can redirect the user to a malicious page.

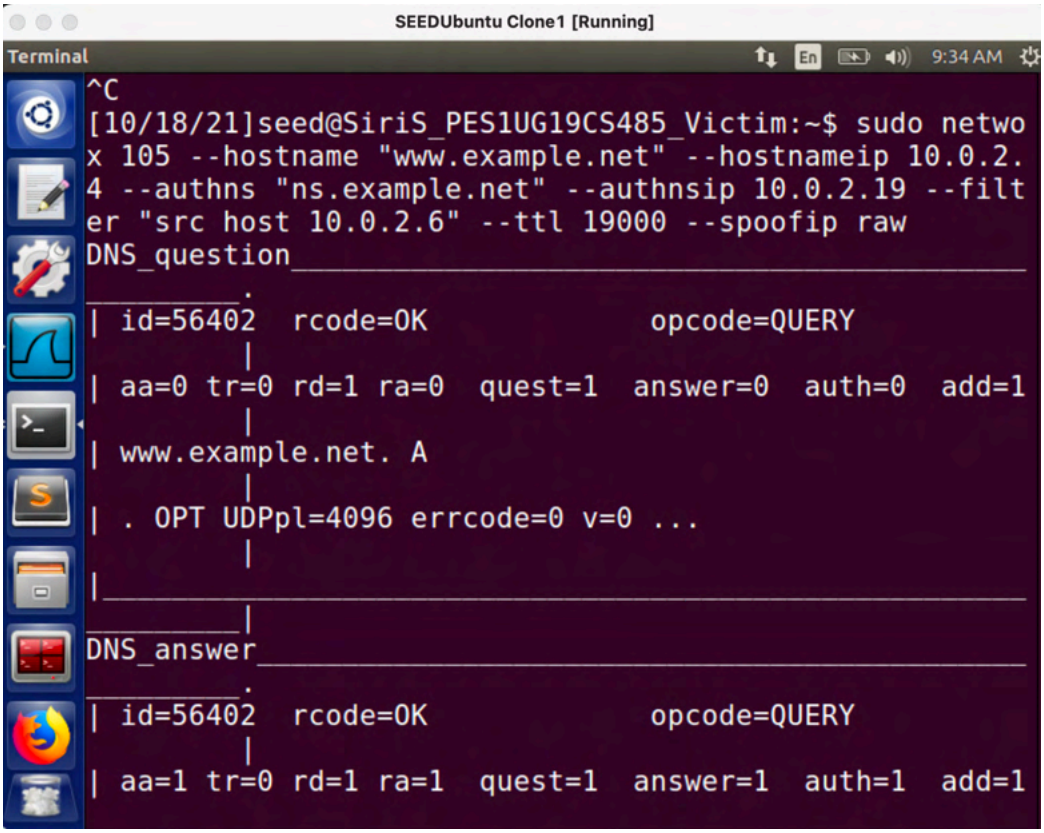
Task 5: Directly Spoofing Response to User

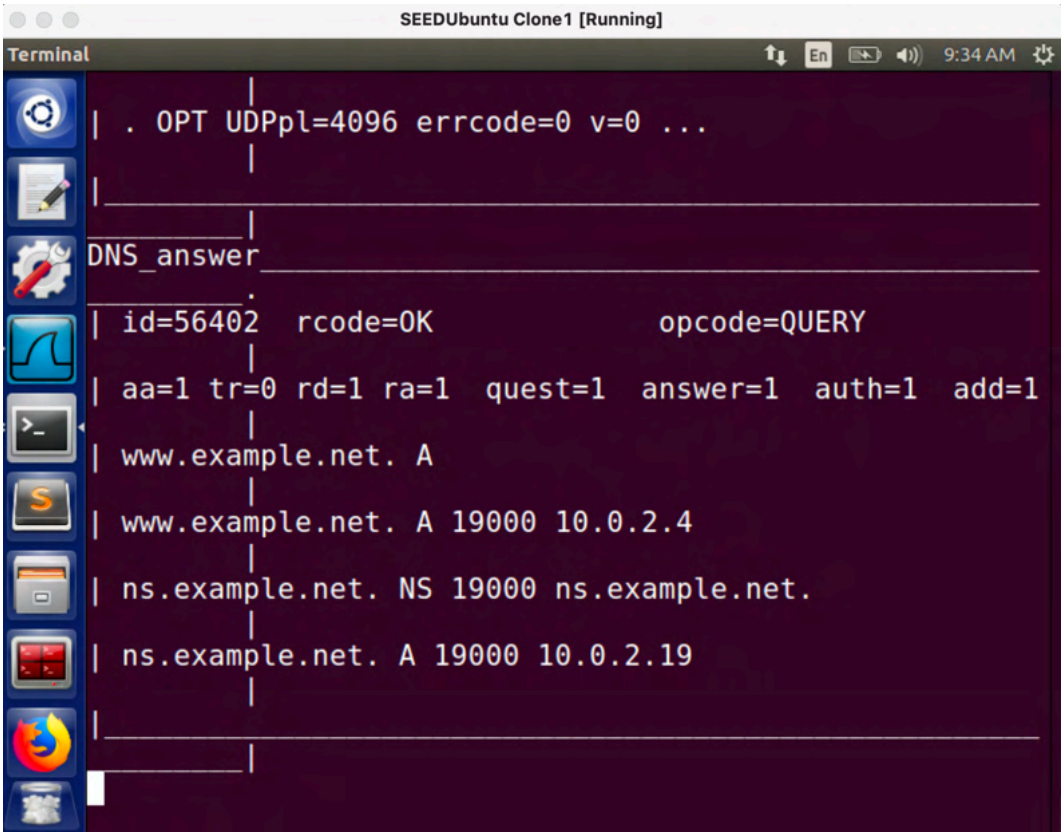
Before attack:



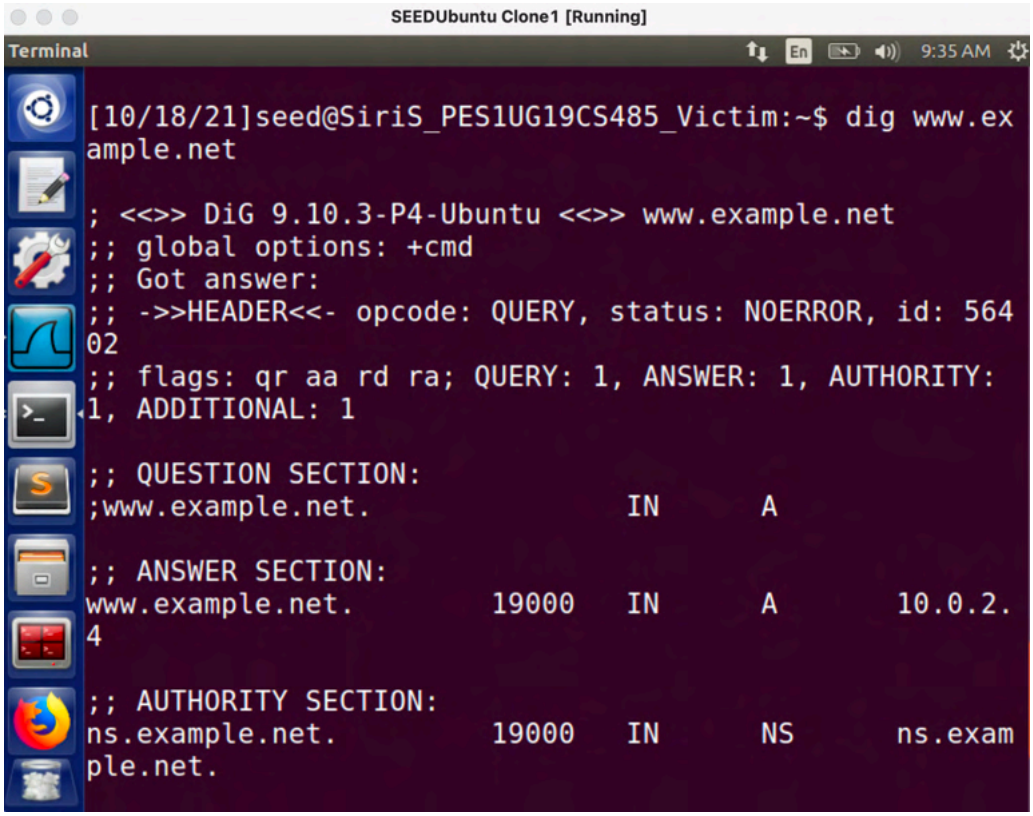


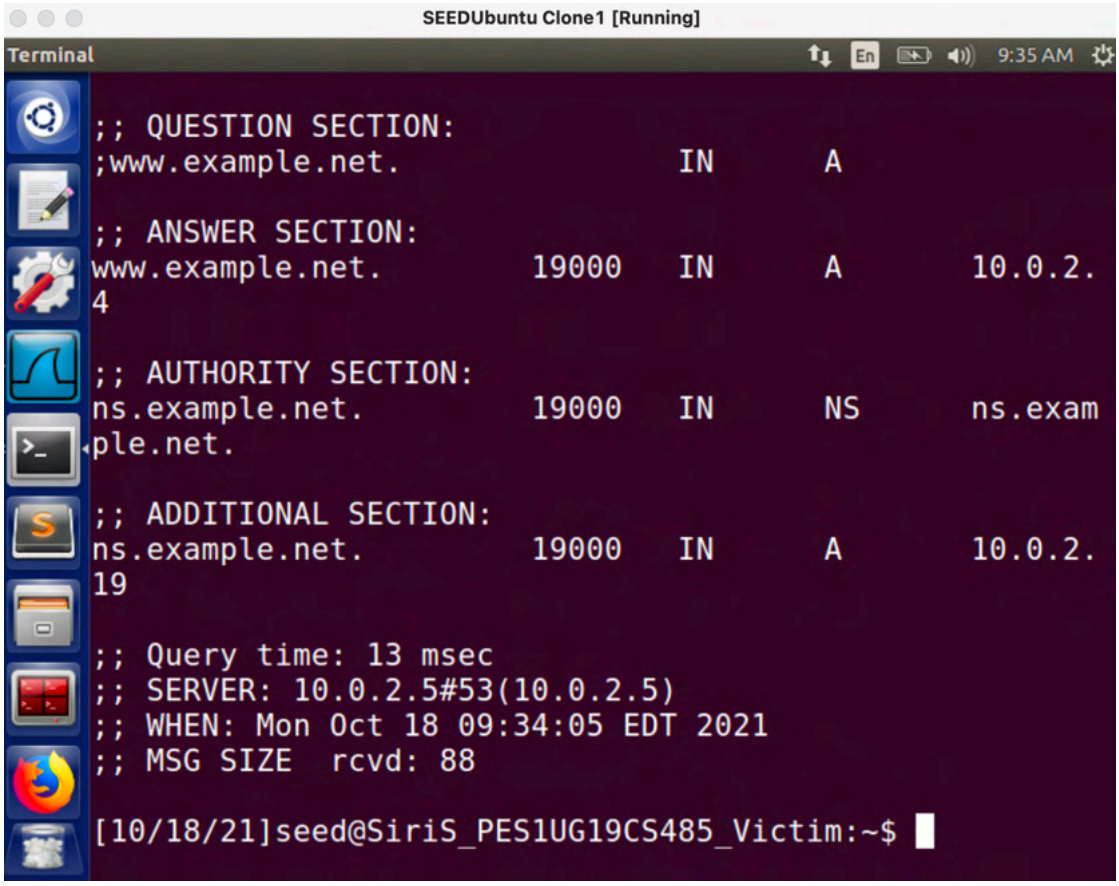
Attack:
Creating a fake DNS response, and sending it back to the victim, before the real DNS server does. Netwox tool 105 provide a utility to conduct such sniffing and responding. We can make up any arbitrary DNS answer in the reply packets. We can use the “filter” field to specify what kind of packets to sniff. For example, by using"src host 10.0.2.19”, we can limit the scope of our sniffing to packets only from host 10.0.2.19.

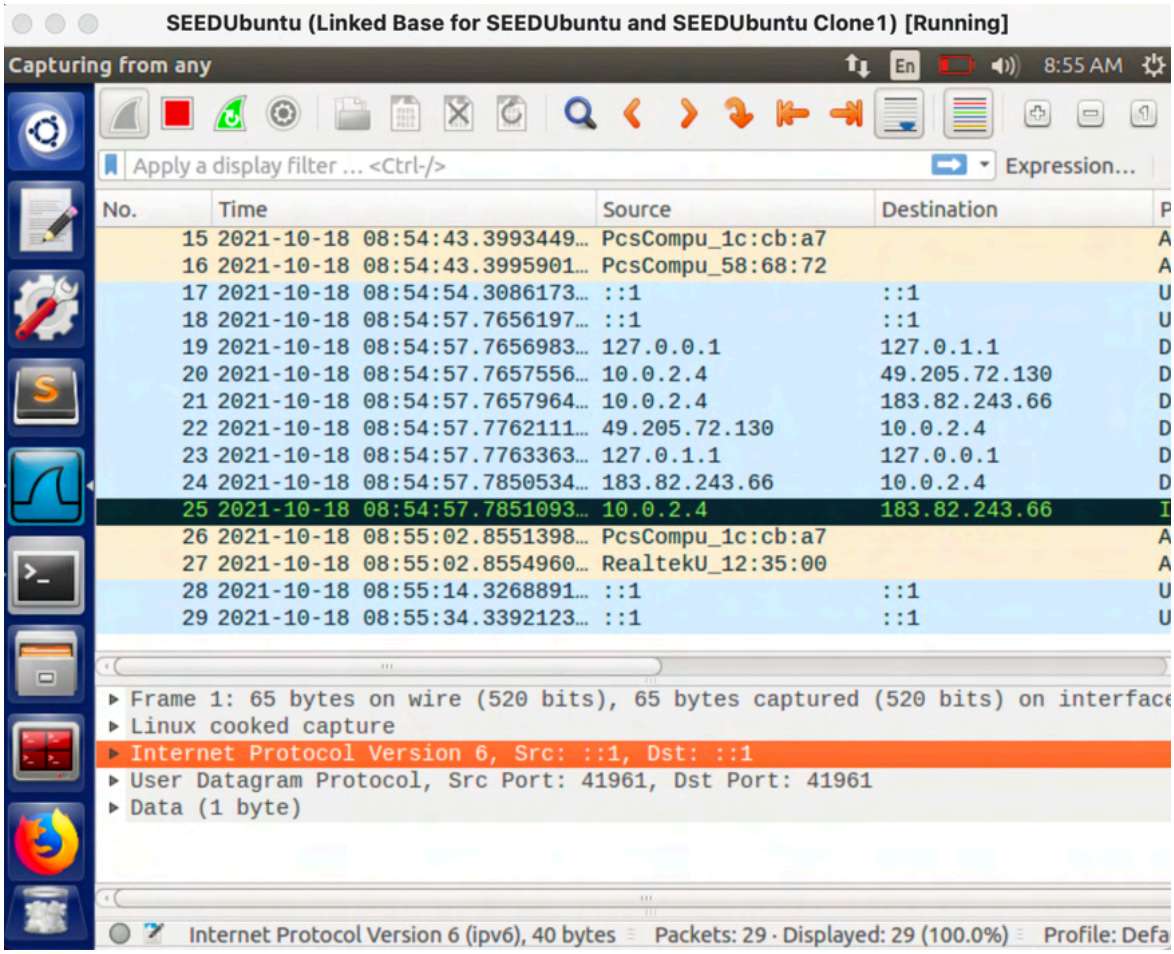




While the attack is running, we run dig command on behalf of the user. This command triggers the user machine to send out a DNS query to the local DNS server, which will eventually send out a DNS query to the authoritative nameserver of the example.net domain (if the cache does not contain the answer).



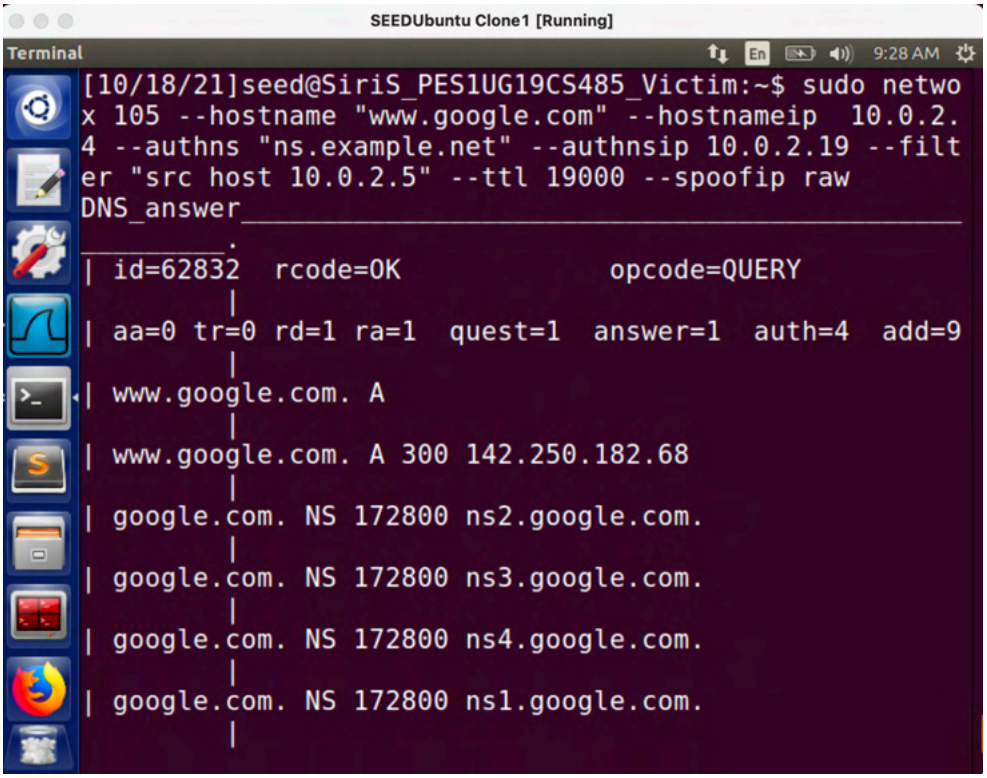


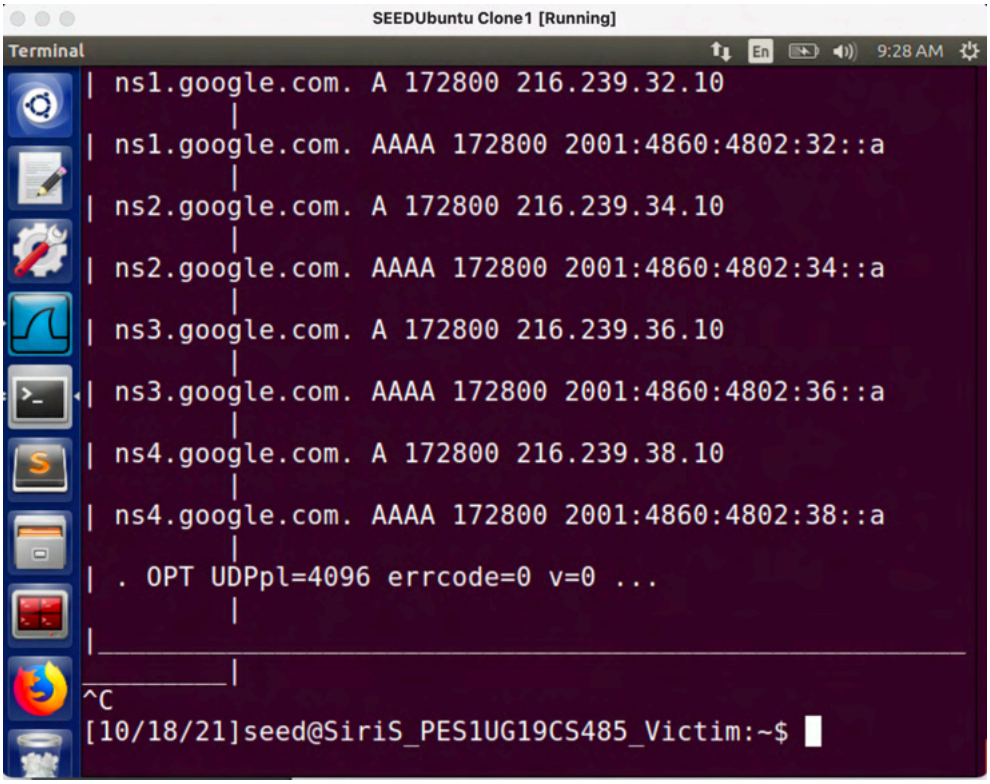


Wireshark screenshot

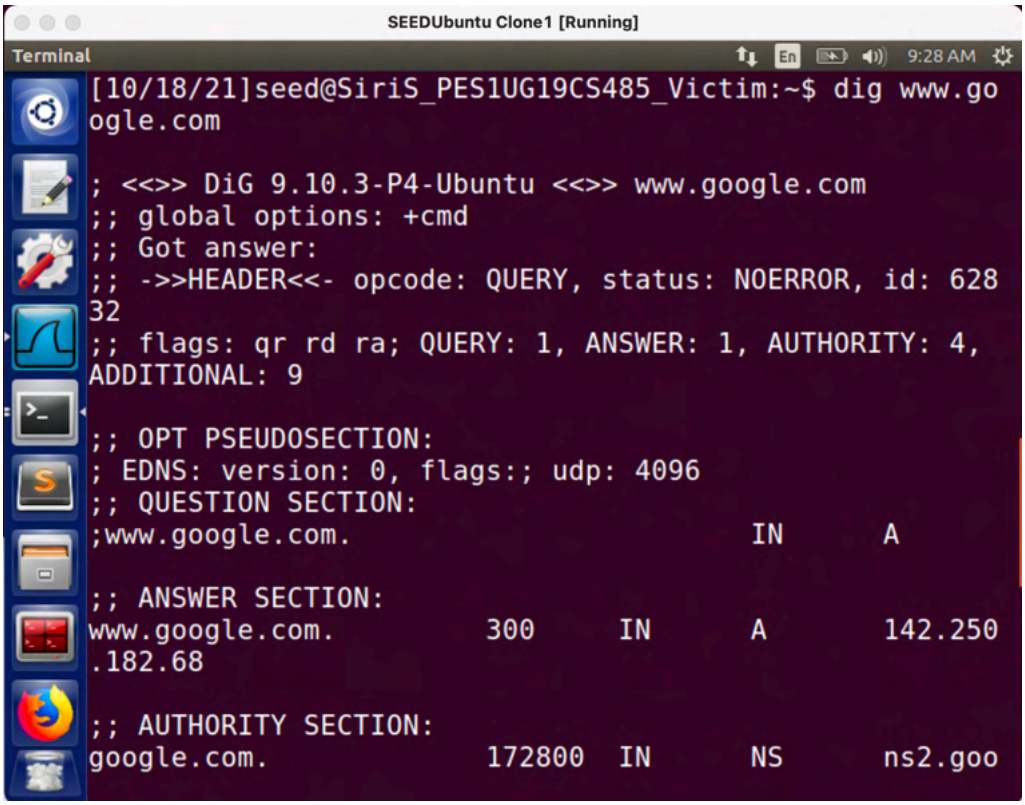
<u>Task 6:</u> DNS Cache Poisoning Attack

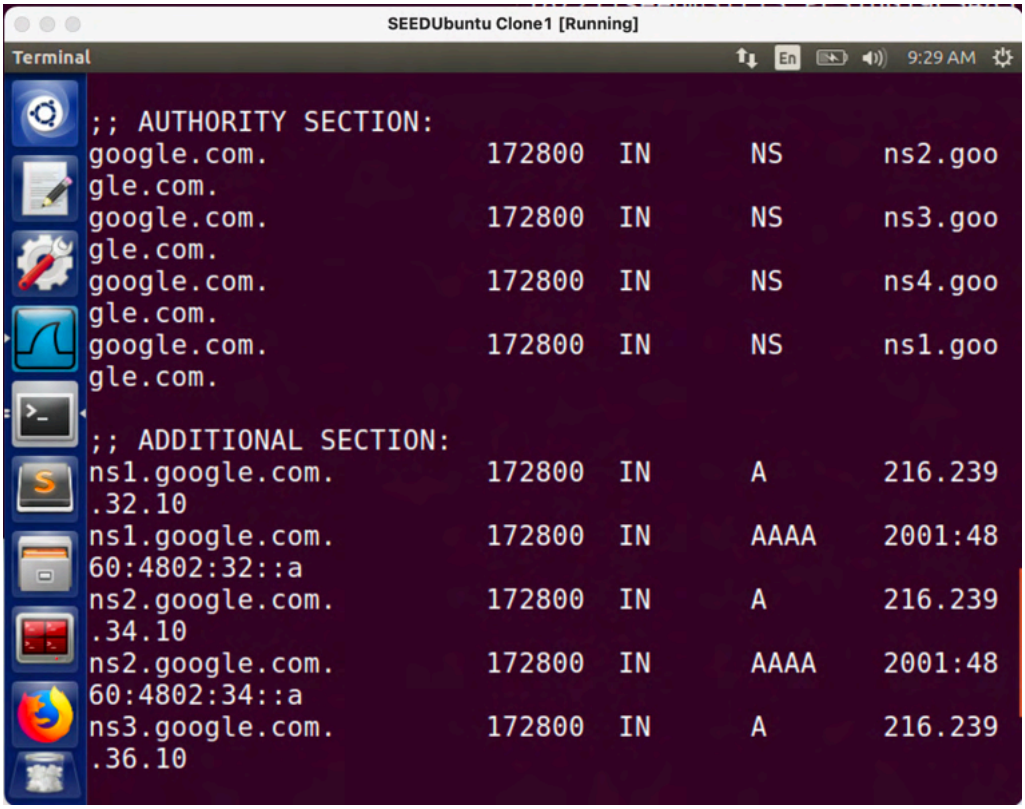
In this attack, we target the DNS queries from the local DNS server (10.0.2.11). In our forged reply, we map www.google.com to 10.0.2.9 and the authoritative server as 10.0.2.19. The netwox tool sniffs the DNS query packet from the DNS server "10.0.2.11" (filter) and sends the forged DNS response packets to the DNS server.

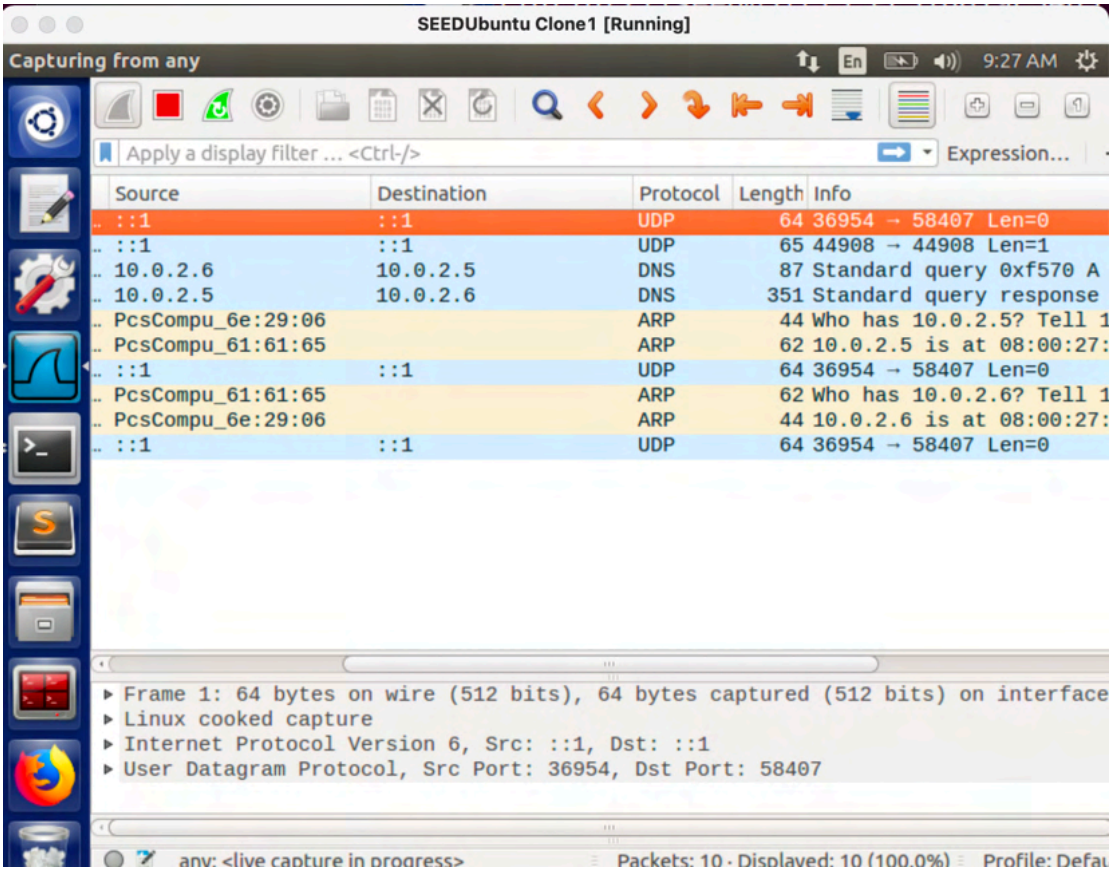




When the attack program is running, we run “dig www.google.com” on behalf of the user. We see that the victim machine gets a forged reply from the DNS server with the Answer section, Authority section and Additional section.



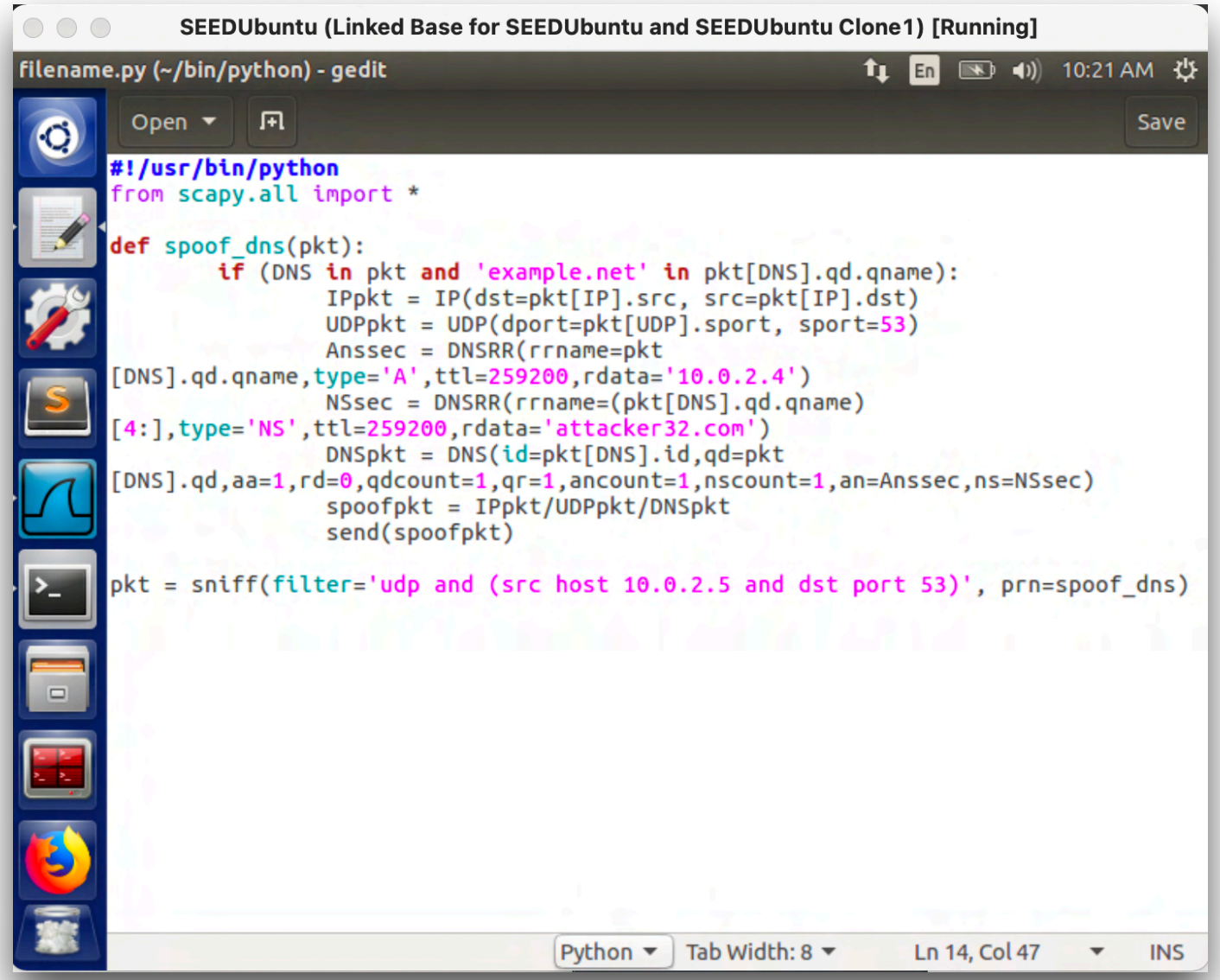




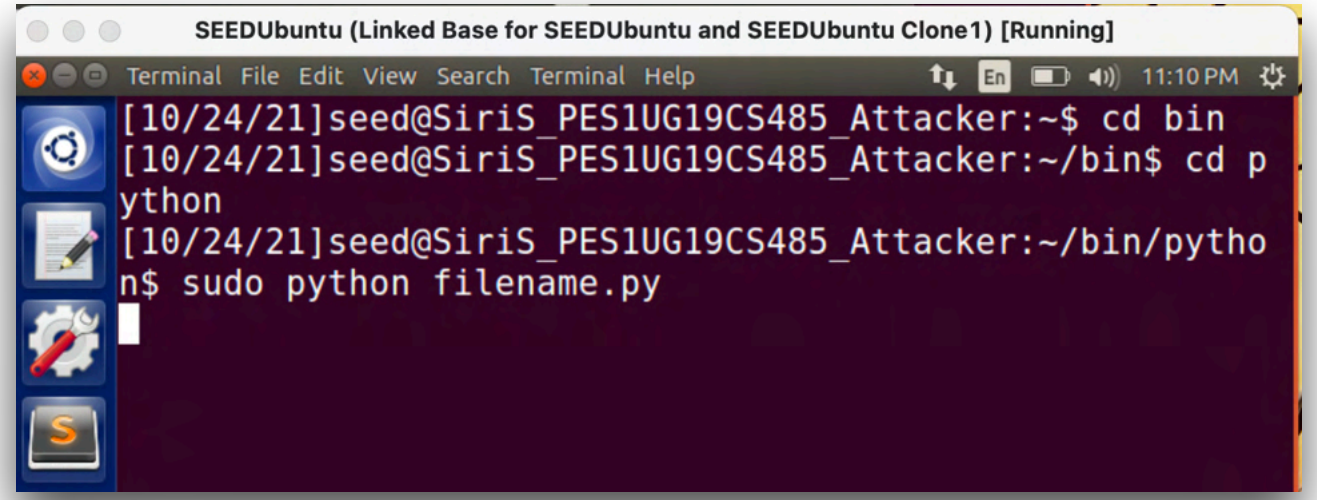
Wireshark screenshot

<u>Task 7:</u> DNS Cache Poisoning: Targeting the Authority Section

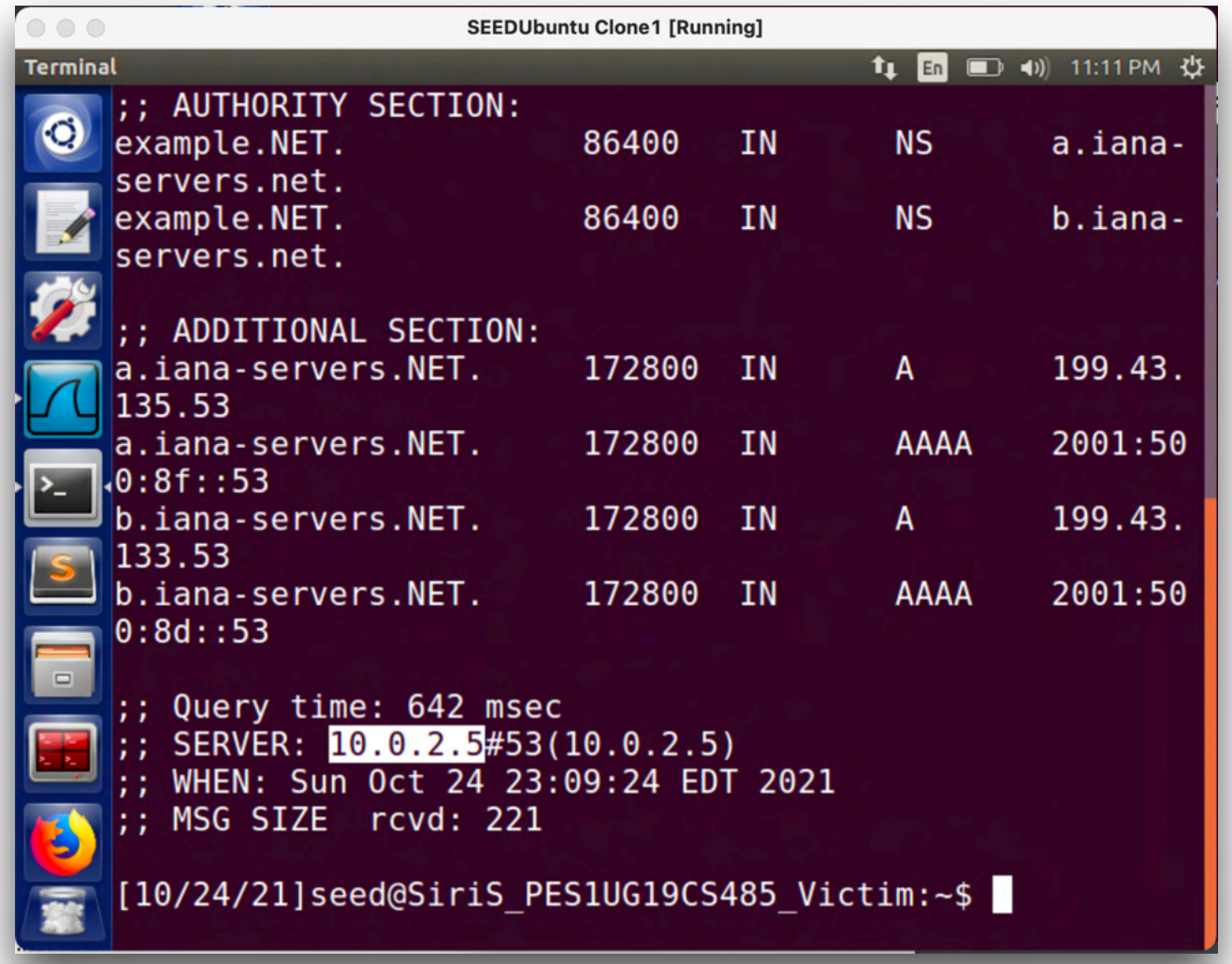
We add “ns.attacker32.com” to the Authority section. When this entry is cached by the local DNS server, ns.attacker32.com will be used as the nameserver for future queries of any hostname in the example.net domain.



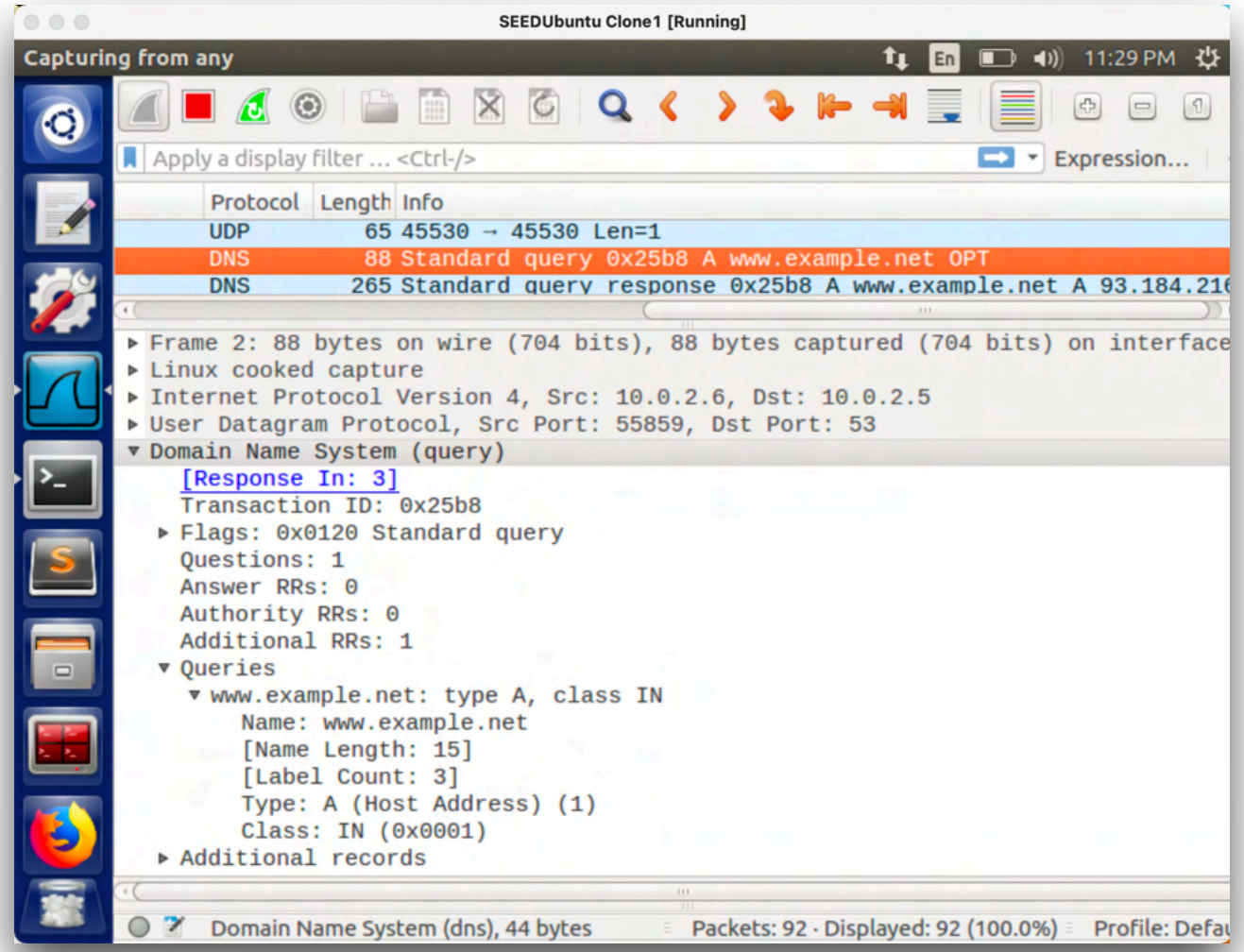
Upon running the code in the attacker:



We can check the cache dump which caches the authority name server and answer.



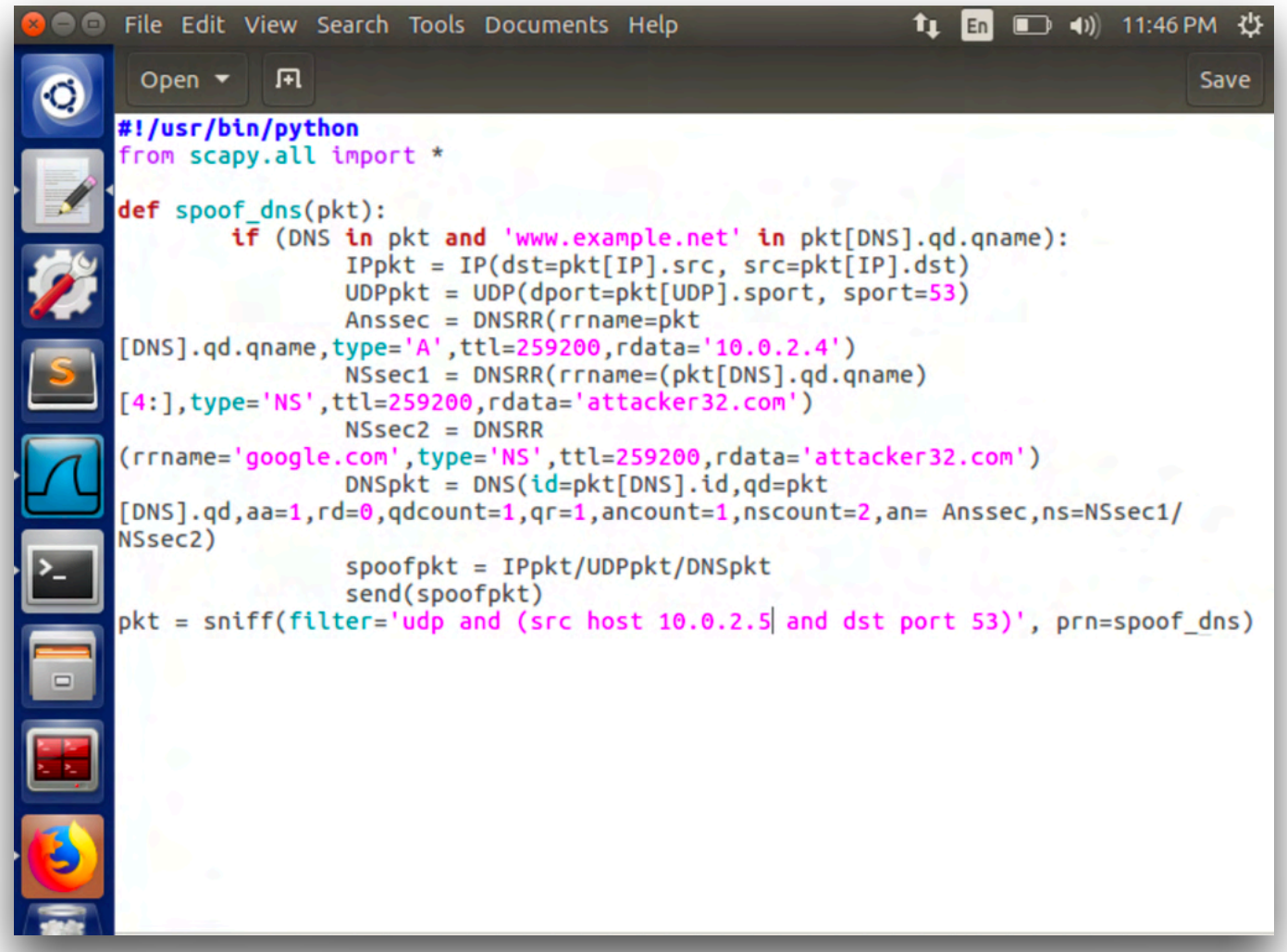
Wireshark screenshot:



Using Wireshark, we can see that a DNS query request is sent to our DNS server (10.0.2.5). 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.

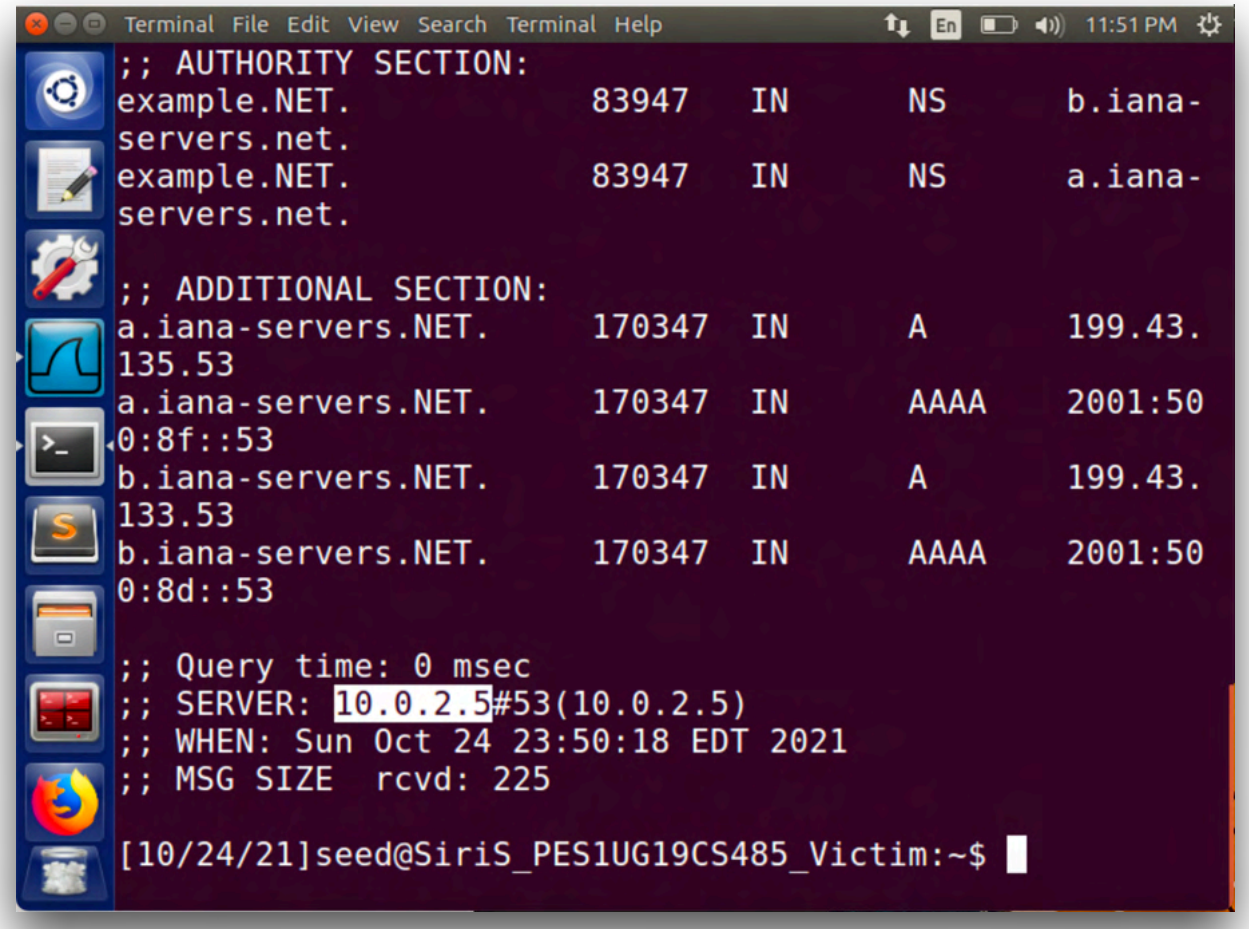
<u>Task 8:</u> Targeting Another Domain

Inorder to to extend the impact of the cache poisoning with a nameserver for the domain to other domains, we run this code:

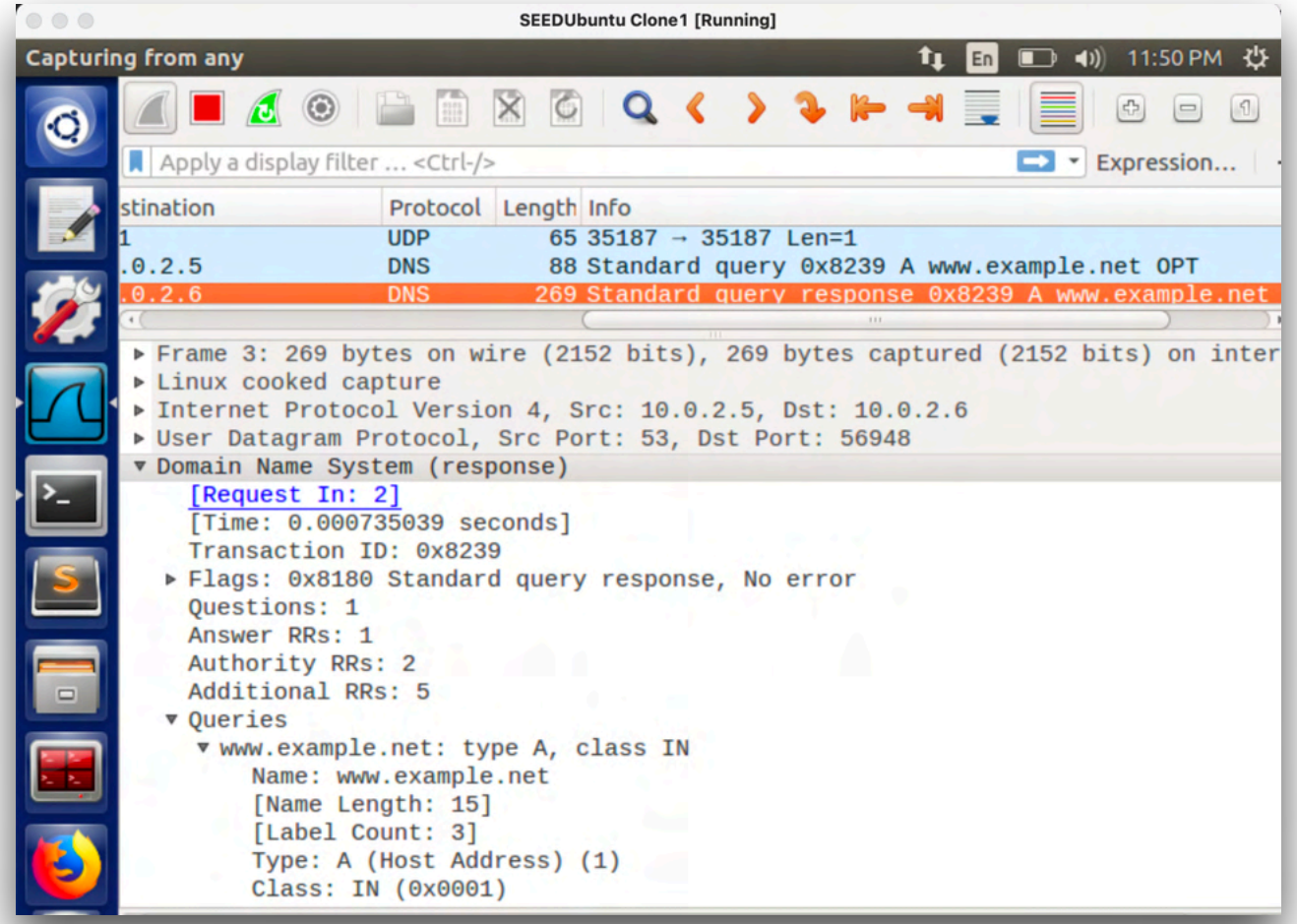


The following code sends DNS response with two authoritative entries similarly as we did for the previous task. We create a Name Server Resource Record for “google.com” with “attacker32.com” as the domain.

After running the code, we run “dig www.example.net” on behalf of the user. 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 for “example.net”, but not for “google.com”.

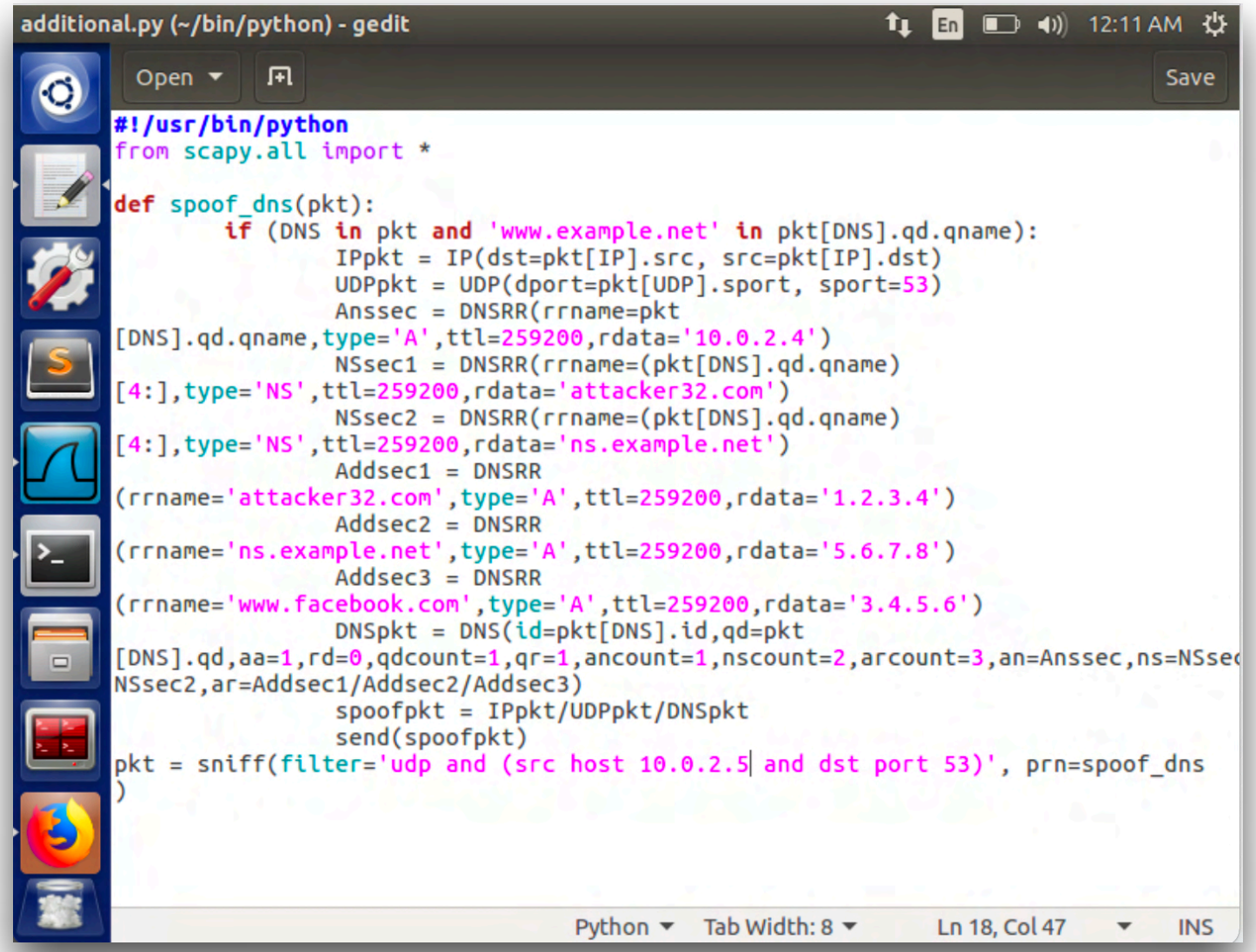


Wireshark Screenshot:

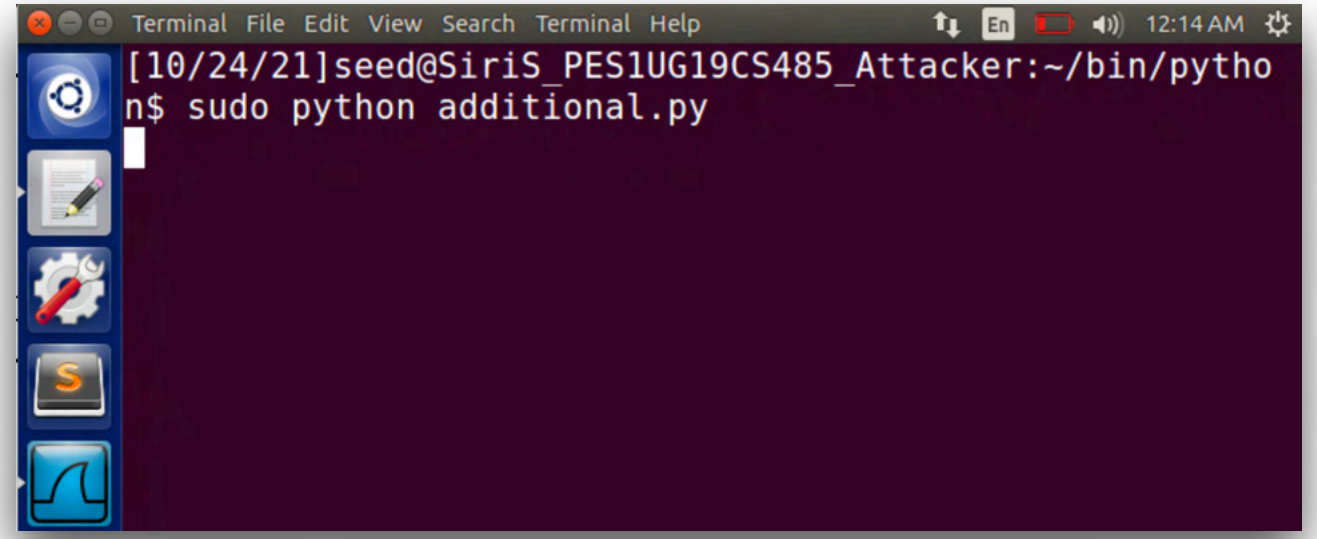


Task 9: Targeting the Additional Section

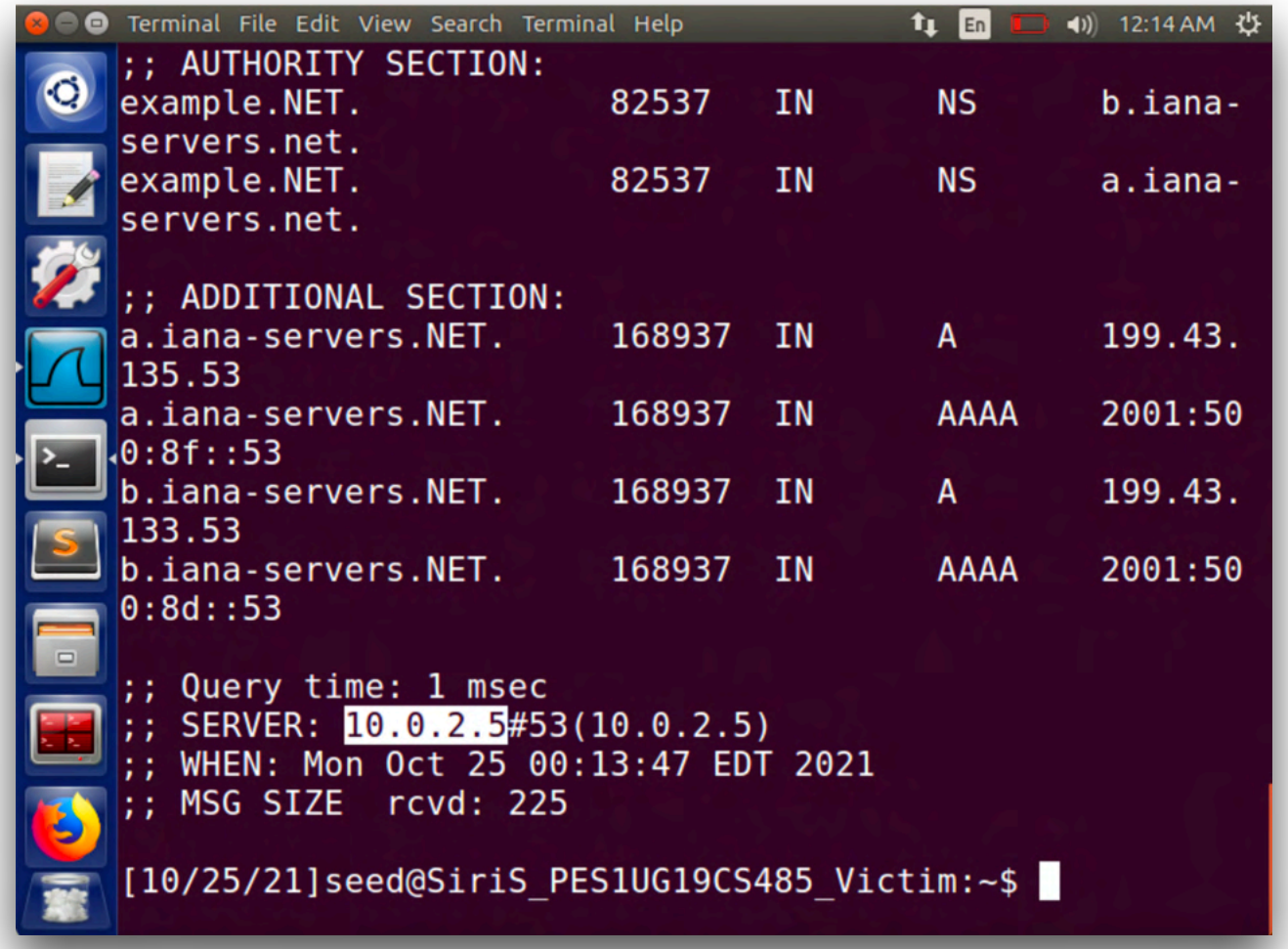
The below code shows that we create Additional sections with the Resource Record name (domains/nameservers) and Resource data (IP address). The resource record names are “attacker32.com”, “ns.example.com”, “www.facebook.com”.



We run the code on the attacker:



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 a forged reply from the DNS server with the 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.



Wireshark screenshot: