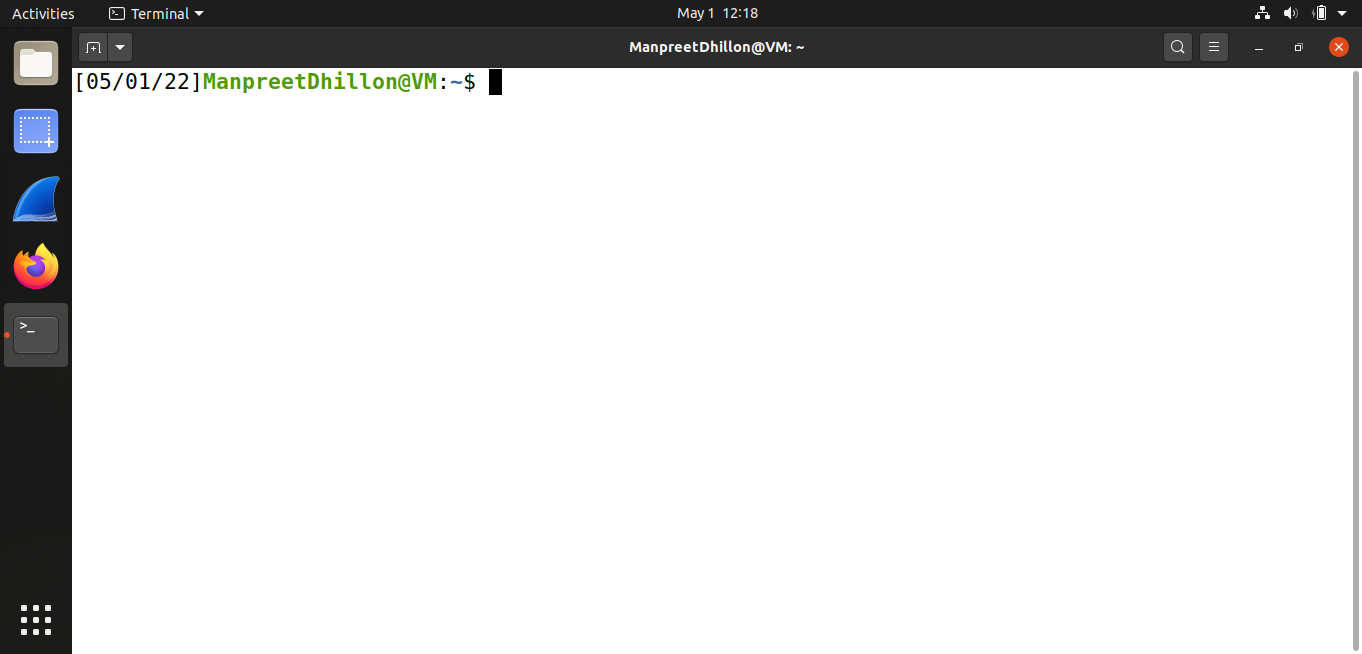
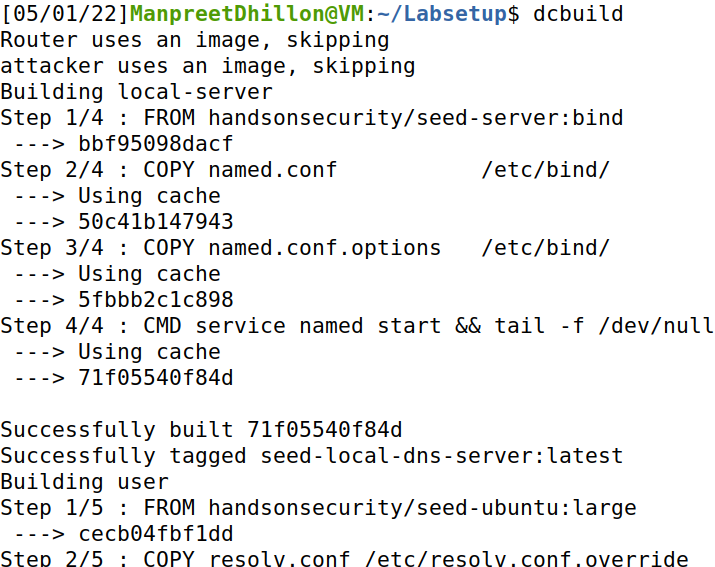
**DNS Local Lab Report**

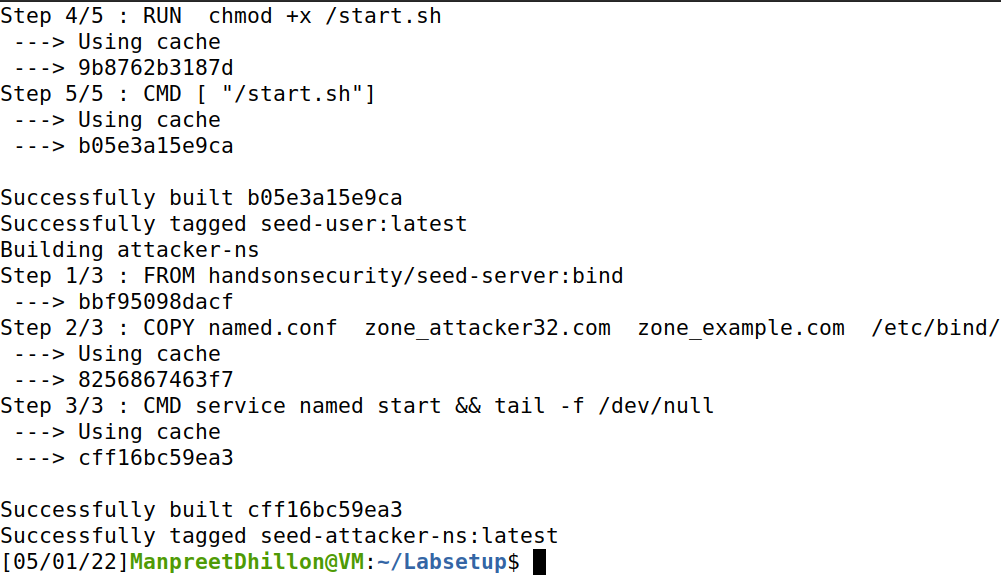
**Lab Setup**

I downloaded the pre-built Ubuntu VM from SEED labs website and set it up in the virtual machine.

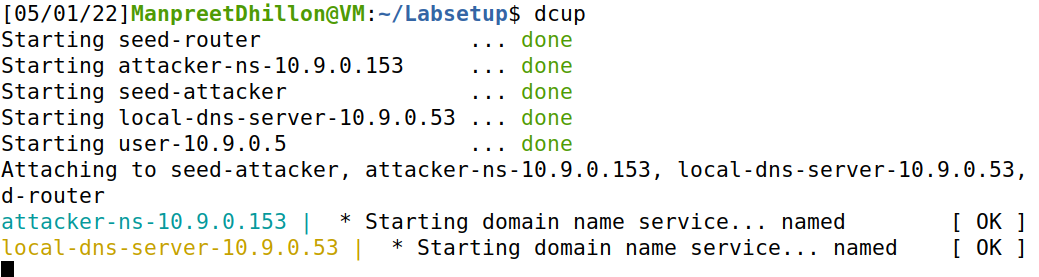


I copied the provided zip file to the VM and extracted it then built the lab using the “dcbuild” command listed on the instructions.

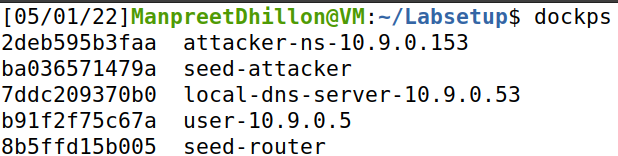




I started the docker machine using “dcup”

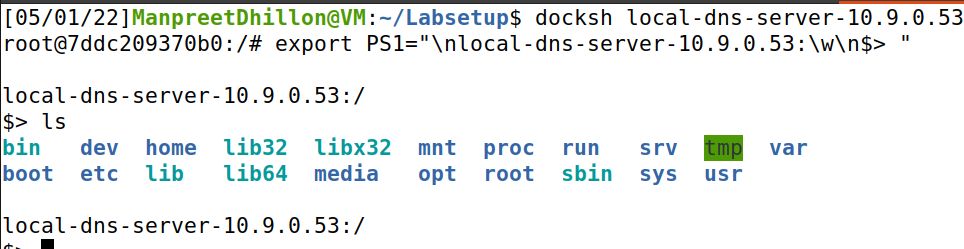


Checking the machines.

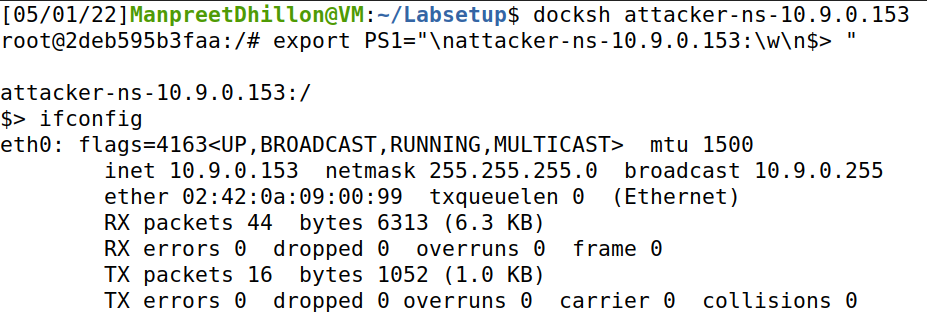


I then spawned the shells of each docker machine and edited PS1 in order to easily differentiate each instance.

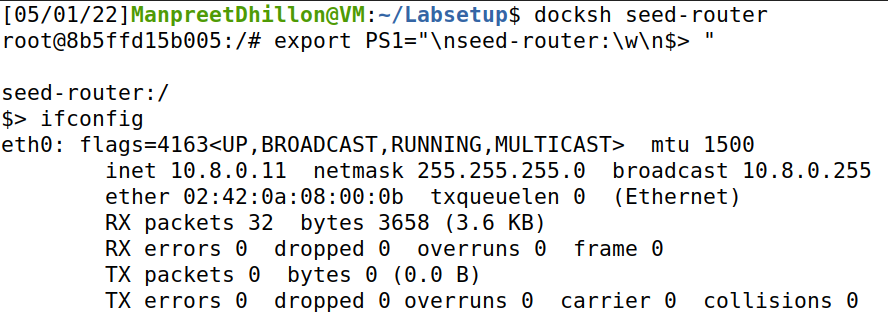
Local DNS



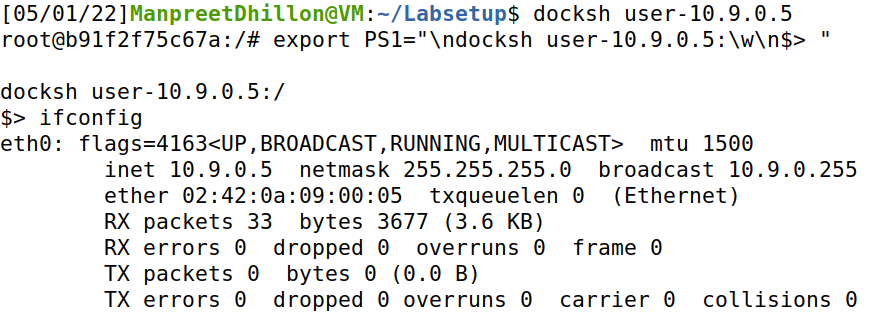
Attacker DNS



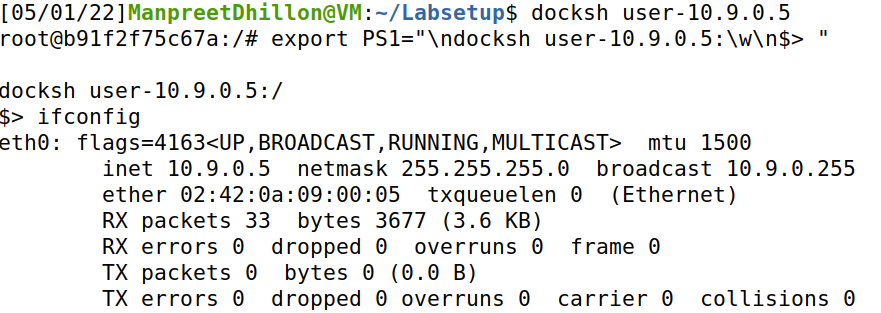
SEED Router



User machine



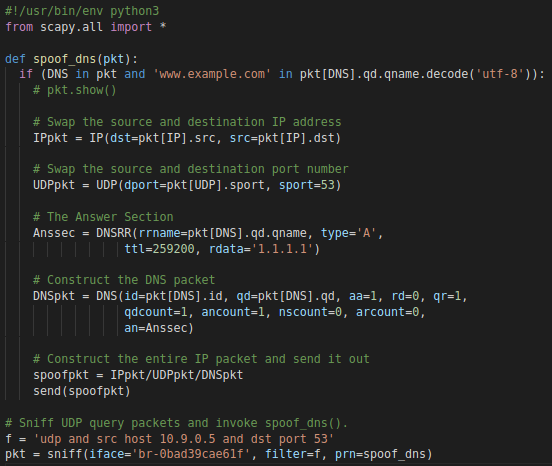
Attacker machine



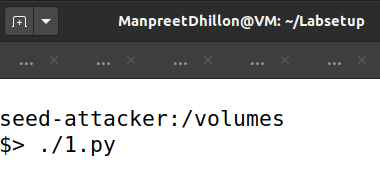
All the machines were up and running as expected.

**Task 1: Directly Spoofing Response to User**

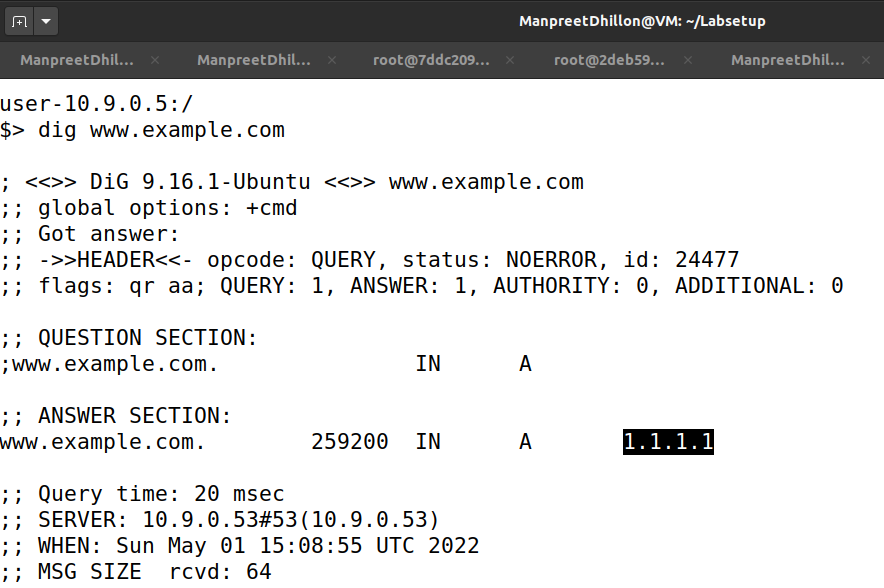
To directly spoof the user DNS response, the script had to listen to DNS query from the user machine hence I set the filter to listen for “[www.example.com](http://www.example.com)” DNS request from 10.9.0.5, user machine, for connections going to port 53, DNS default port and respond with “1.1.1.1” as . Here is the script:



I cleared the DNS cache on the local DNS server then ran the script on the attacker machine.



Using the dig command to get DNS info on the user machine.

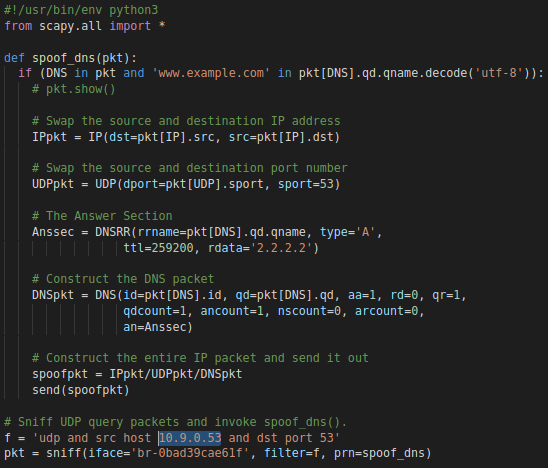


The script successfully spoofed the user machine.

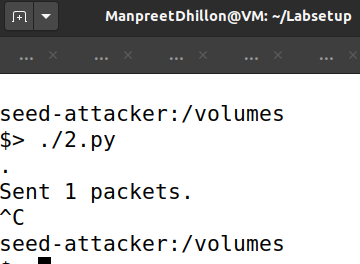
**Task 2: DNS Cache Poisoning Attack - Spoofing Answers**

To spoof the DNS, I modified the script to listen to and respond to the DNS server at the filter instead but the other parameter remained constant except the ID address returned which I set to 2.2.2.2.2.

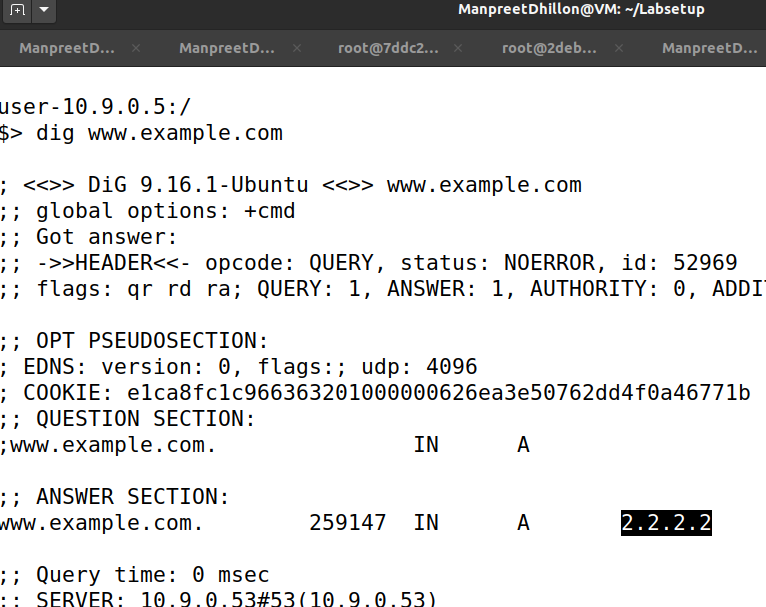
The script



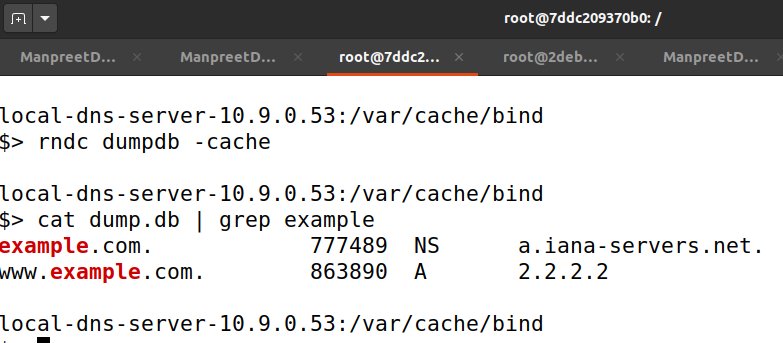
I ran the script on the attacker machine.



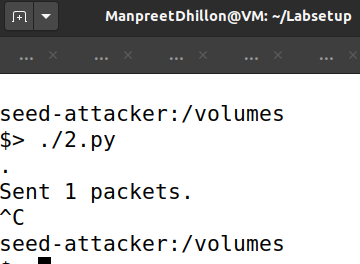
Cleared the DNS cache and ran the dig command on the user machine.



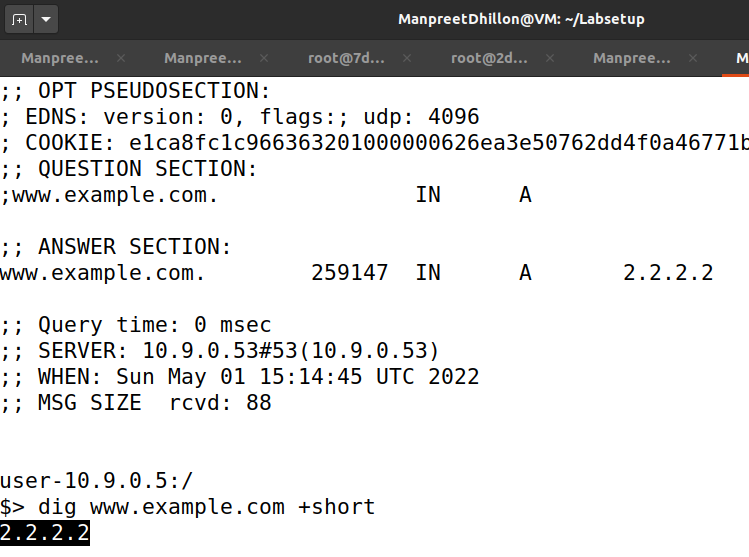
Checking the DNS cache.



The cache poisoning worked, so I stopped the script on the attacker machine.



Then ran the dig command again but this time I filtered the output to only return the IP address using the command like this “dig www.example.com +short”

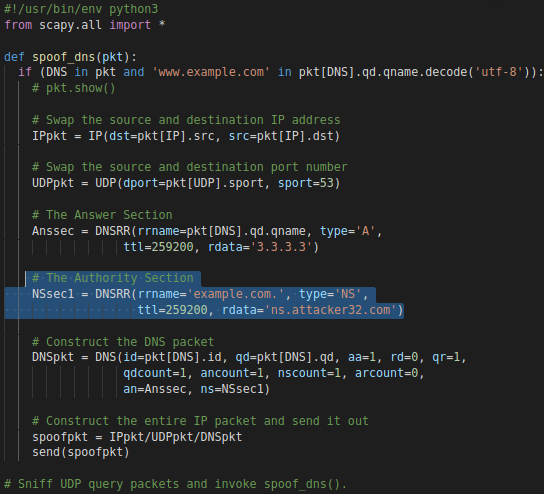


As you can see below, the DNS cache is still poisoned.

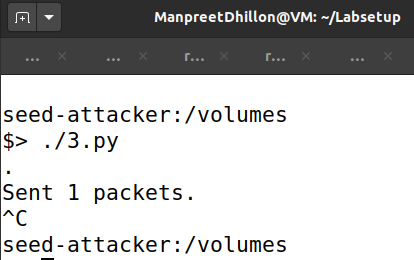
**Task 3: Spoofing NS Records**

In order to spoof the NS records, I modified the script by adding the authority section and adjusting the packet parameters like setting the IP address to be returned to 3.3.3.3 and NS to “ns.attacker32.com”.

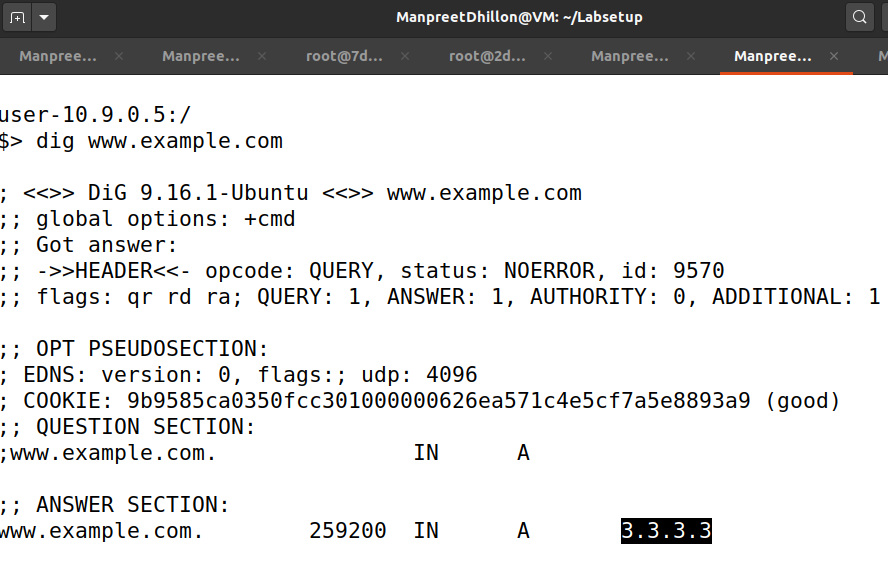
The script.



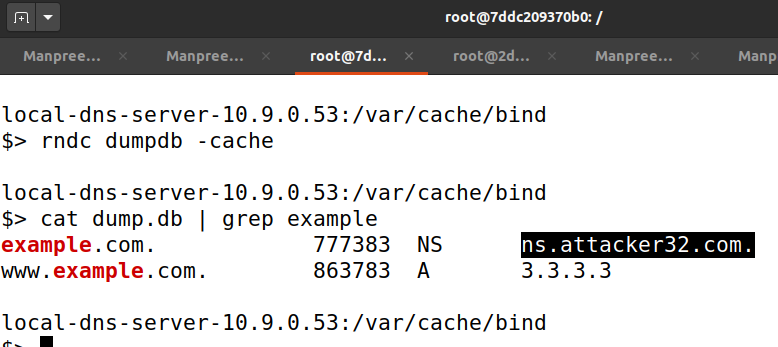
Running the script on the attacker machine.



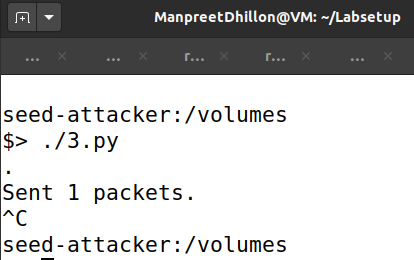
Trigger DNS request with dig command on user machine.



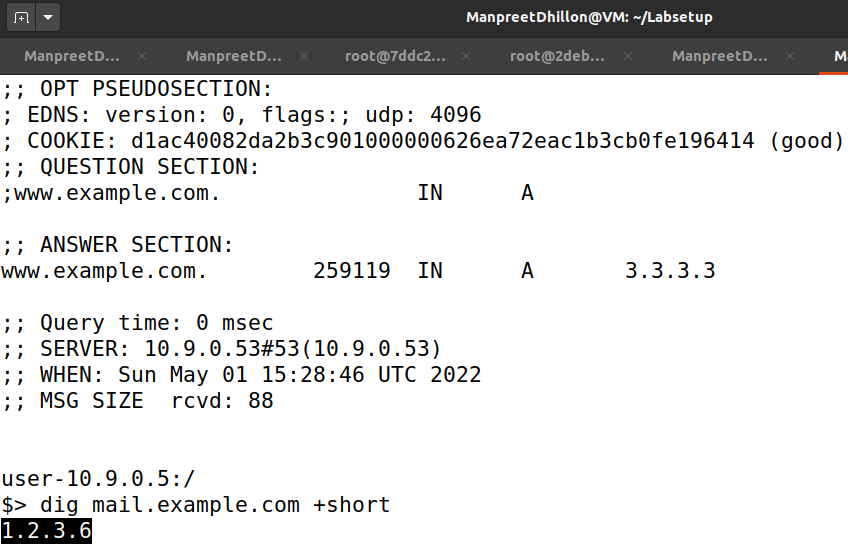
The DNS cache was successfully poisoned and so I checked the nameserver on the cache on the Local DNS machine.



The nameserver was successfully spoofed. So I stopped the script on the attacker machine.



I used the dig command to get “mail.example.com” on the user machine.

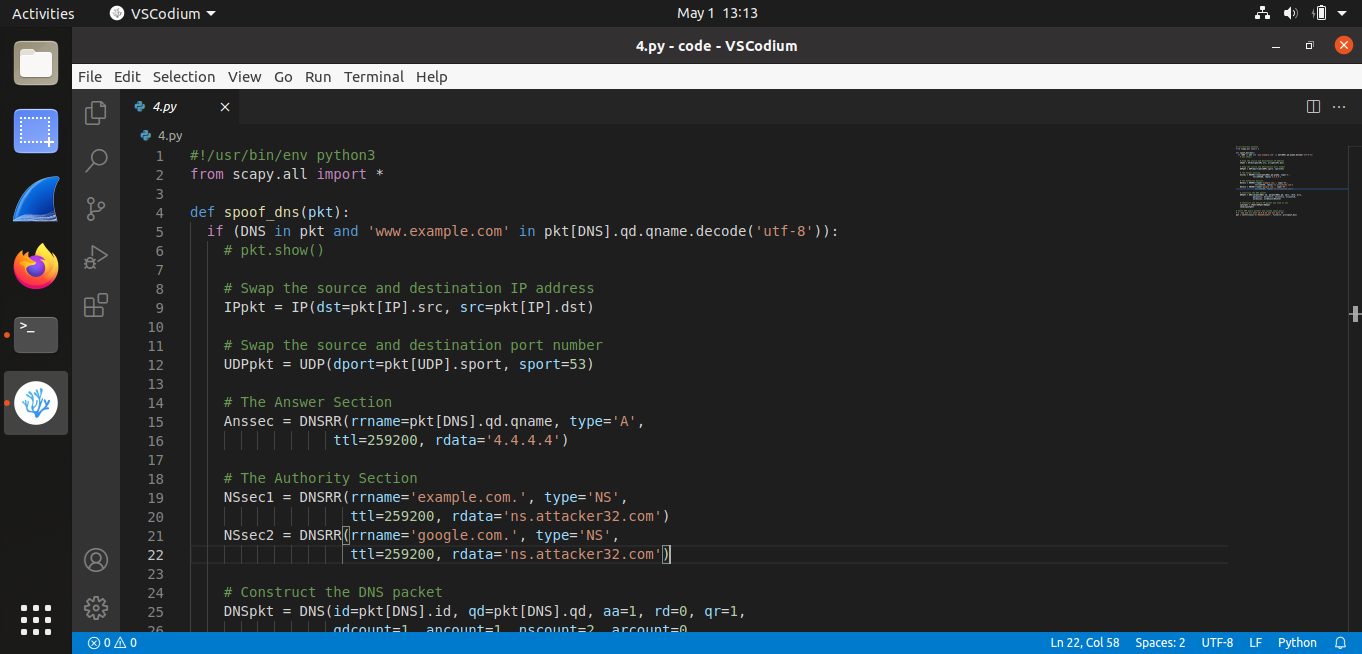


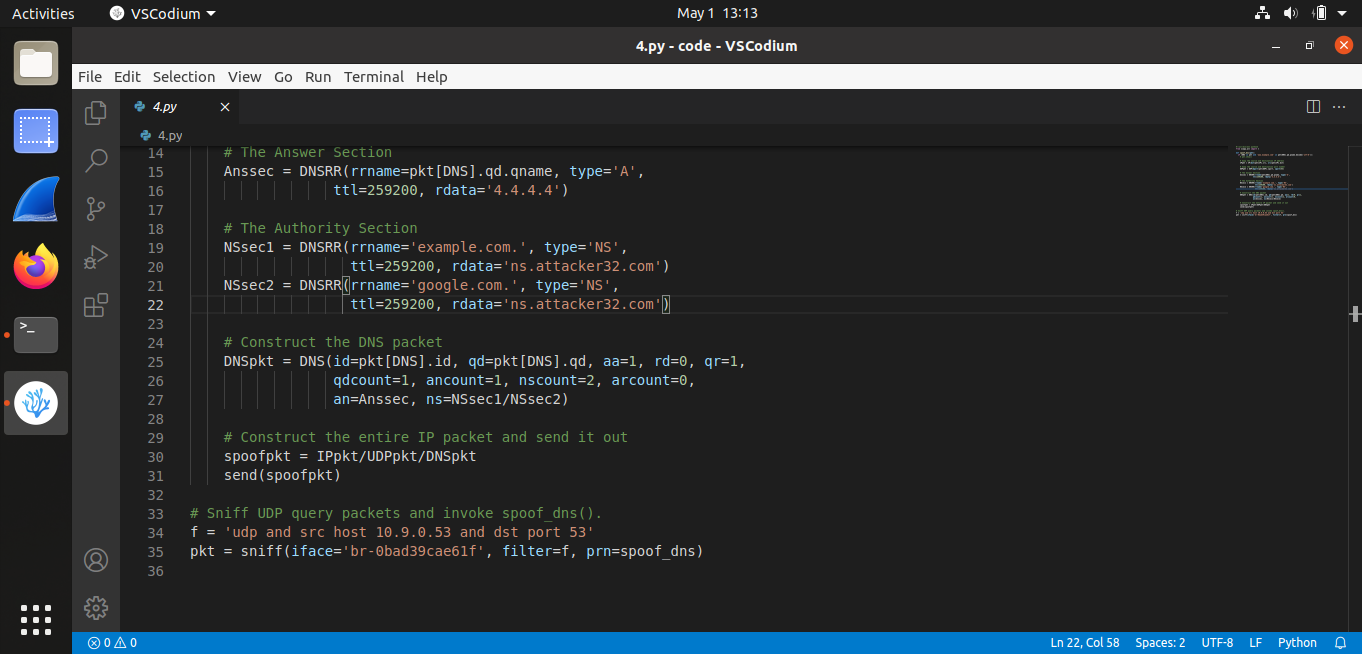
The attack was a success.

**Task 4: Spoofing NS Records for Another Domain**

In order to achieve this, I added another nameserver in the authority section and modified the answer section to return 4.4.4.4 as the IP address.

The script

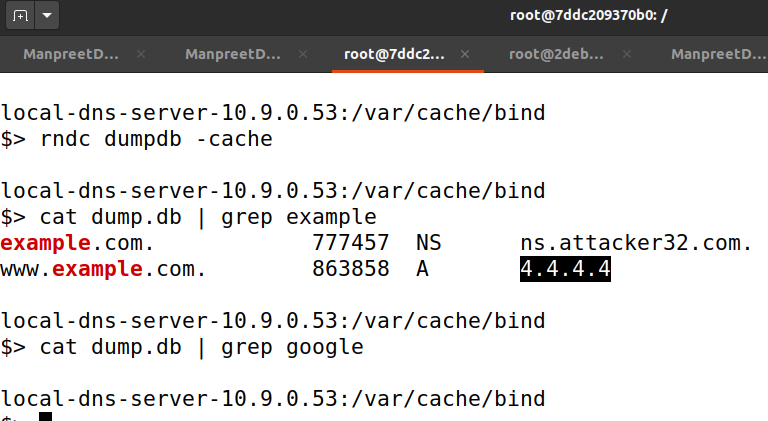




I cleared the DNS cache and ran a dig command on the user machine to trigger a DNS query.



The script successfully poisoned the DNS cache. I checked the DNS cache but fiund out that only example nameserver was spoofed.

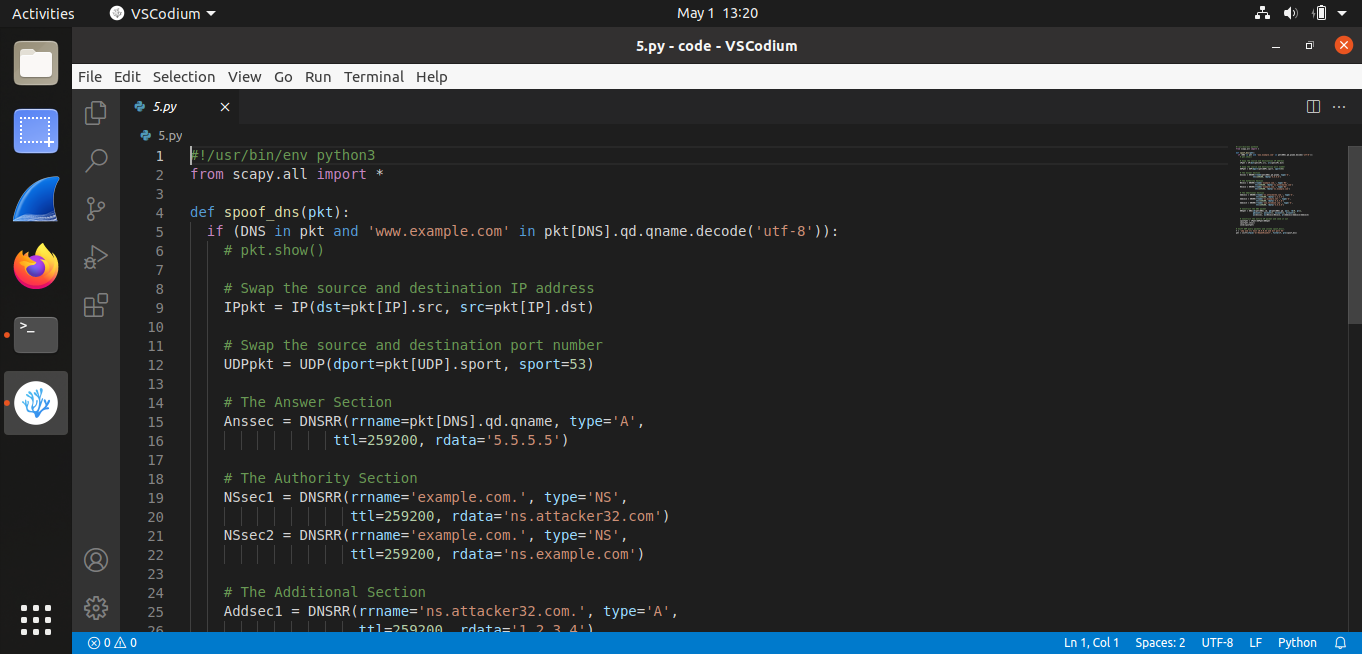


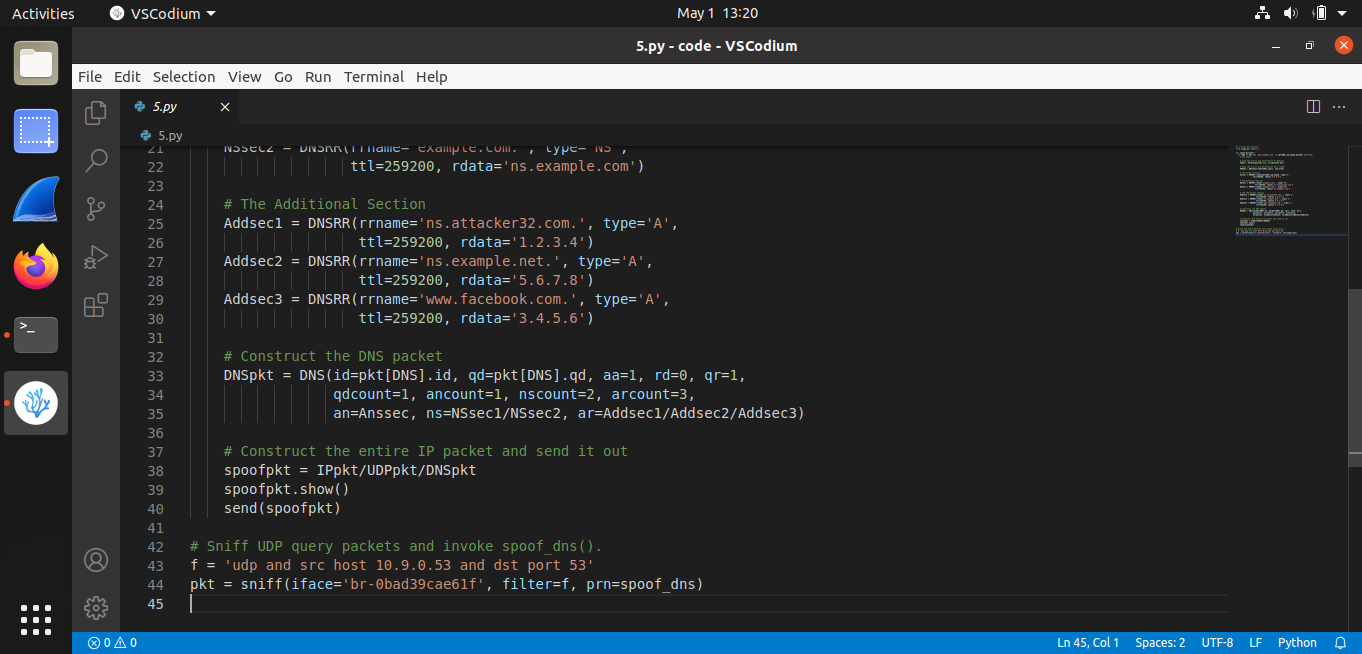
The script did not work as intended and any effort to make it work did not yield a different result.

**Task 5: Spoofing Records in the Additional Section**

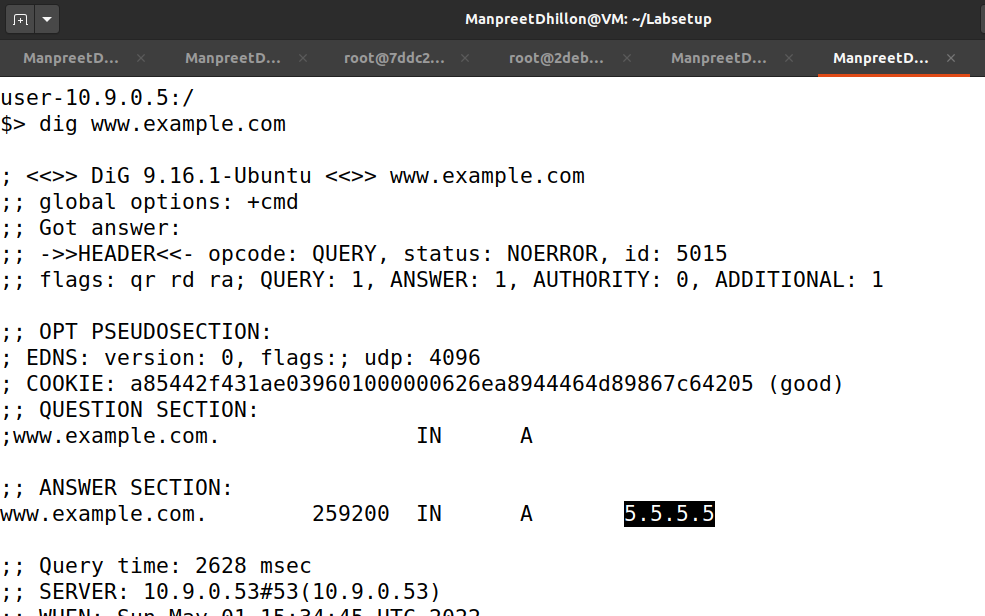
To spoof the additional section, I modified the script above to add an additional section according to the specified instructions. IP address to be returned is 5.5.5.5.

The script.

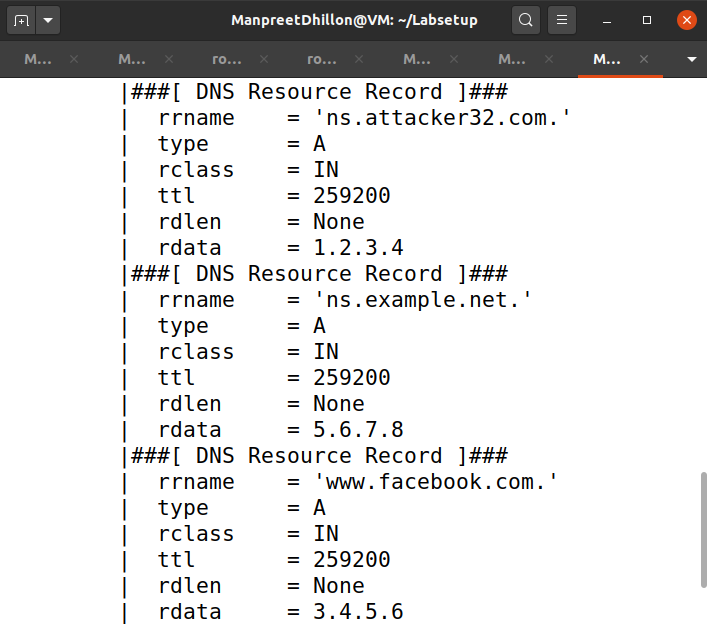




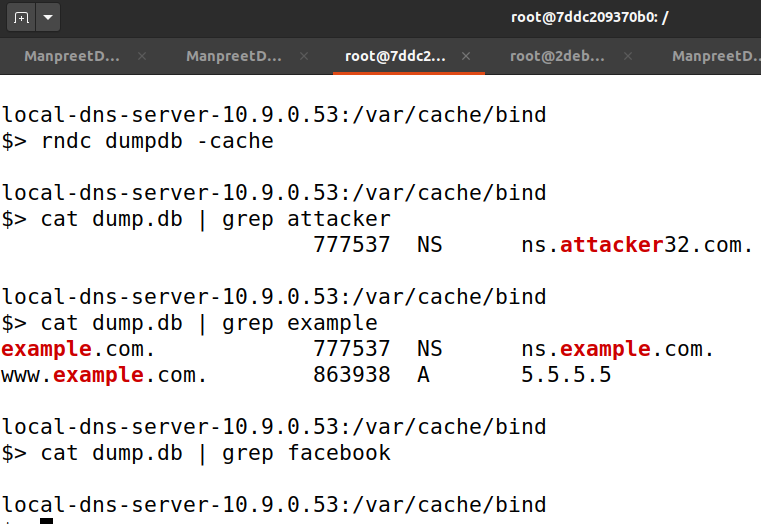
Running the script on the attacker machine.



The DNS cache was poisoned. I checked the packet spoofed by script, and saw that it successfully sent the additional sections.



I checked the DNS cache for an additional section.



facebook.com was notadded because the attacker DNS did not recognize it.

**Conclusion**

Exploring this lab was very fascinating. I enjoyed working on every challenge and learning how DNS queries can be manipulated locally. I have to say my understanding of DNS went from an internet phonebook to a major security hazard if not properly configured.