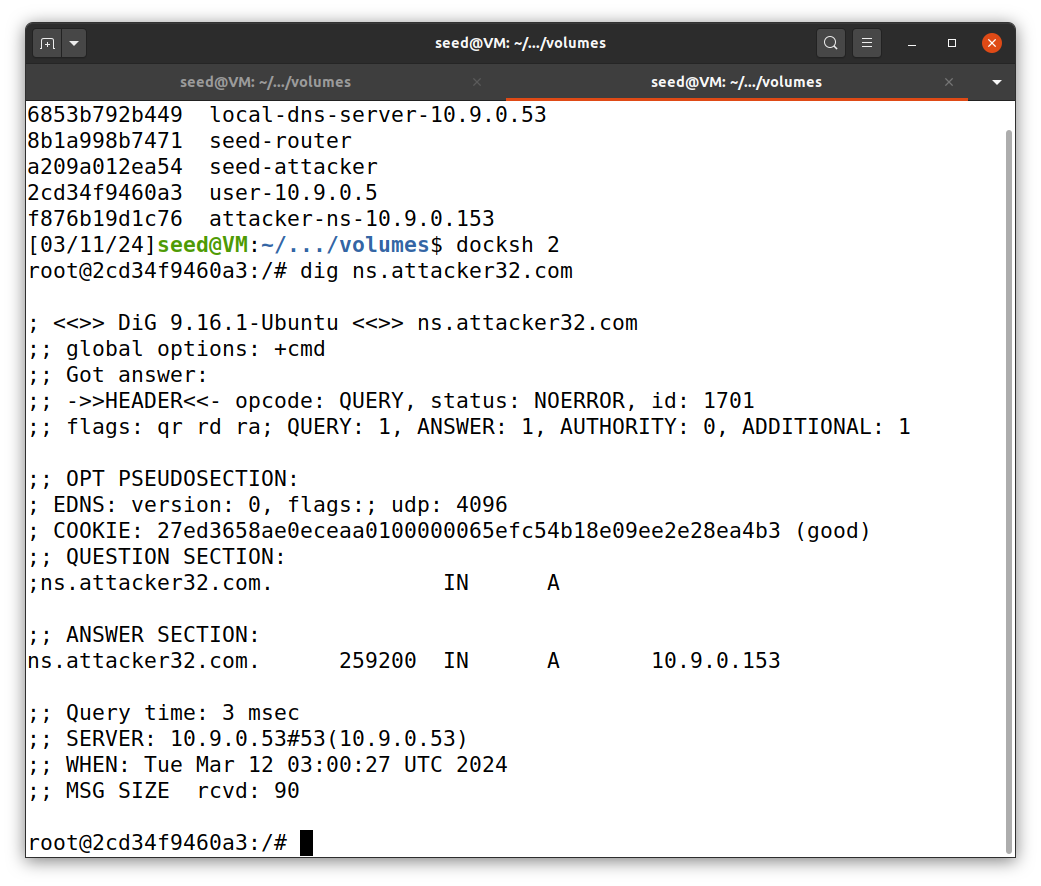
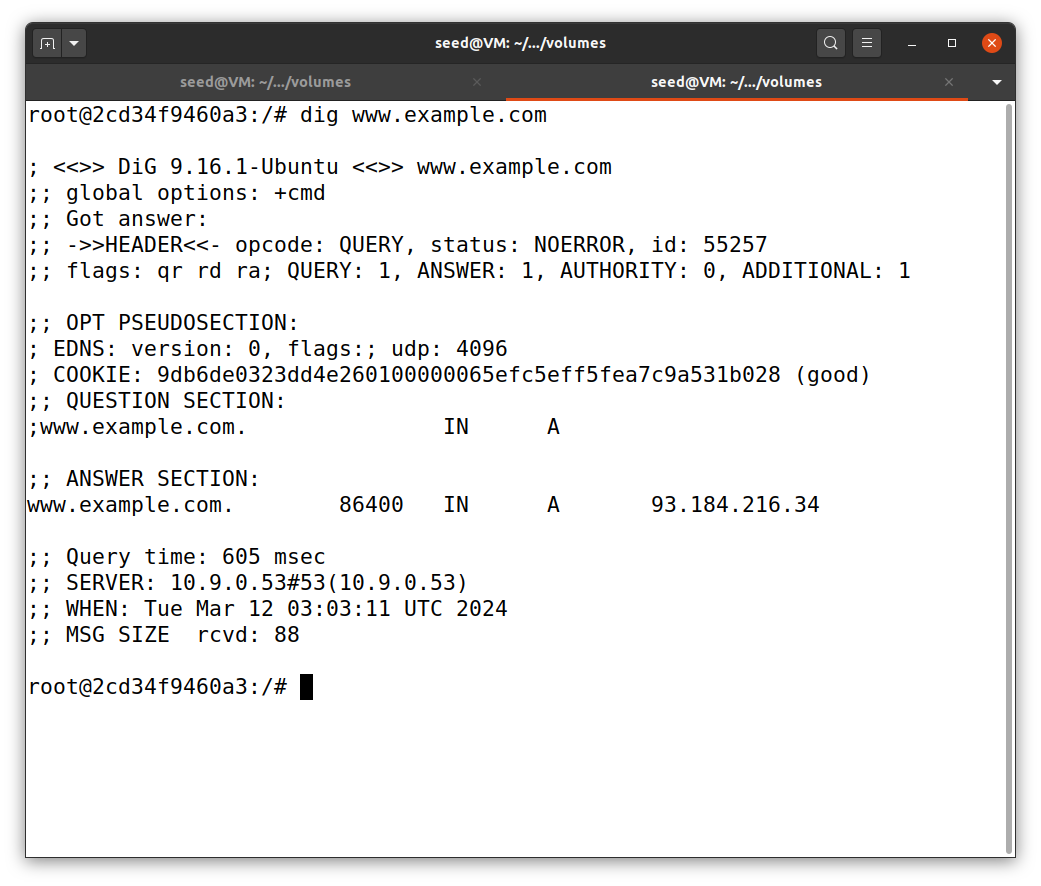
**Lab 7 DNS Attack**

**Testing the DNS Lab Setup**

So first we check the lab setup for this assignment and check few commands to see if everything is working correctly. So first we checked if the docker is up using the dockps , which is true.

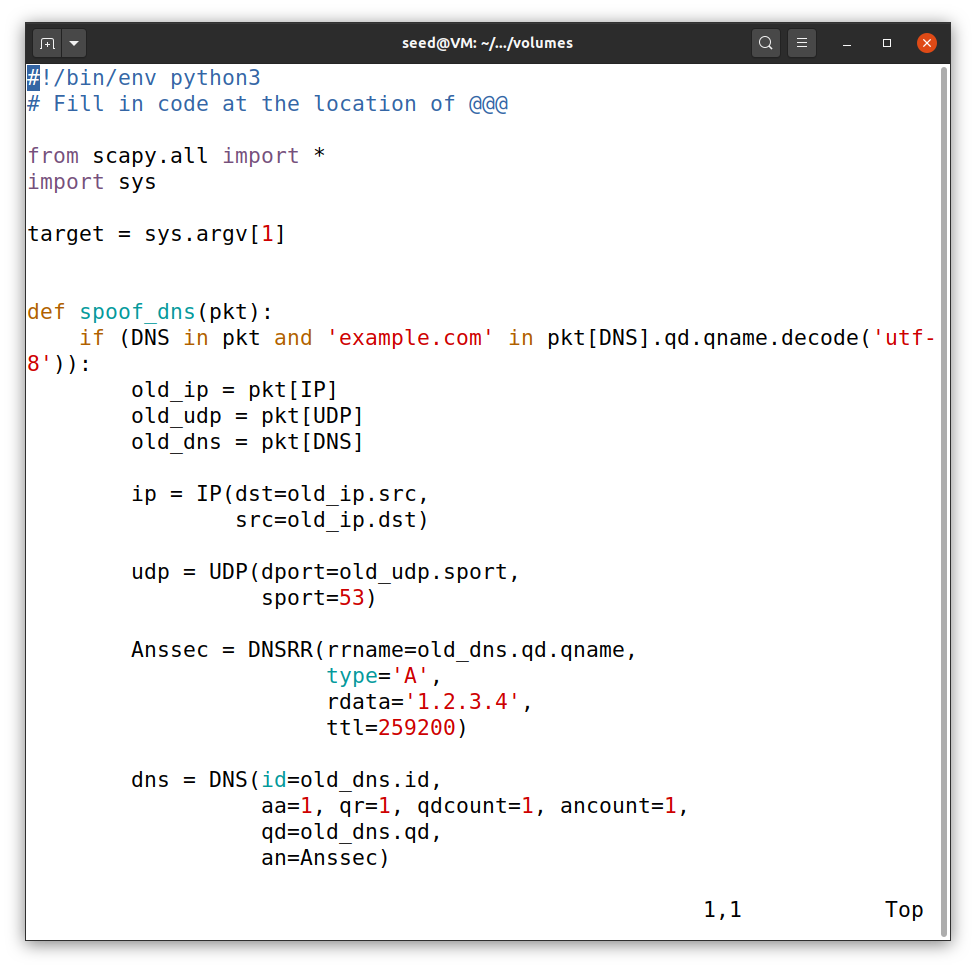
Firstly, I ran the dig command to get the IP address of ns.attacker32.com. This works so the lab setup is done correctly.

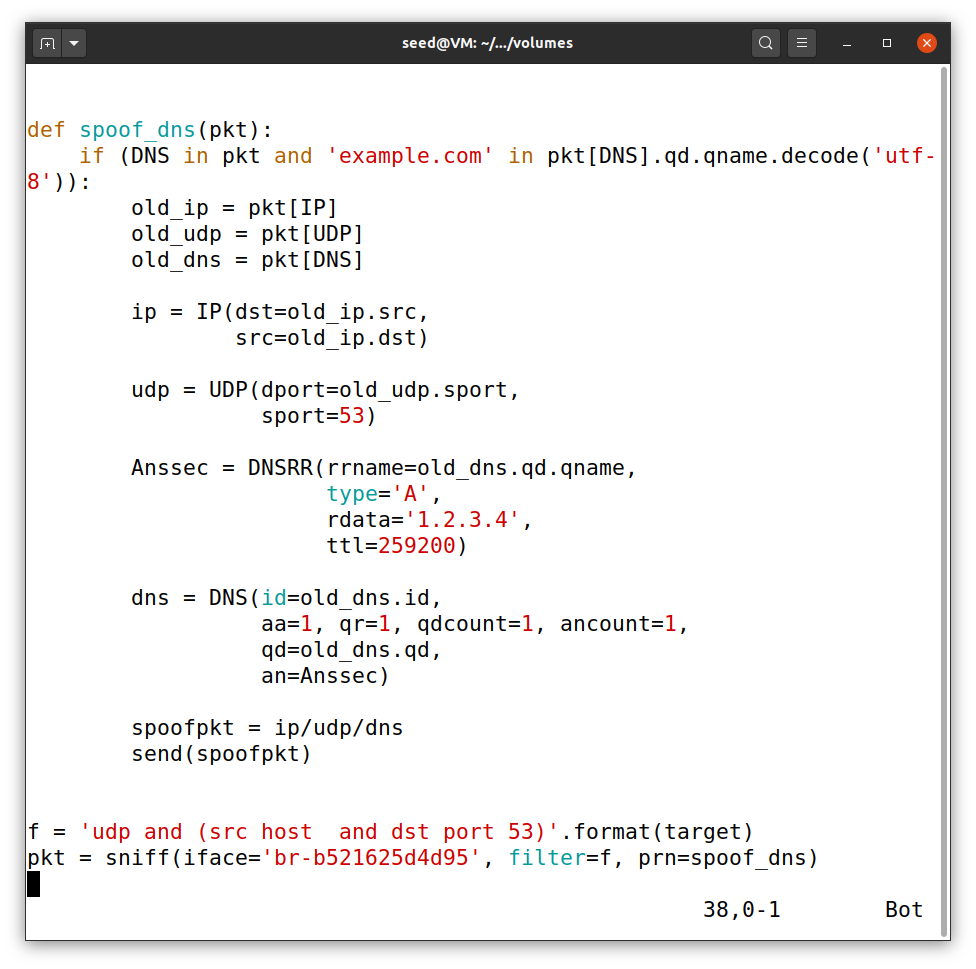
Now we run dig command to get IP address of [www.example.com](http://www.example.com/) by sending the query to local DNS server. So as you can seen after running the command we successfully get the result from the DNS server.

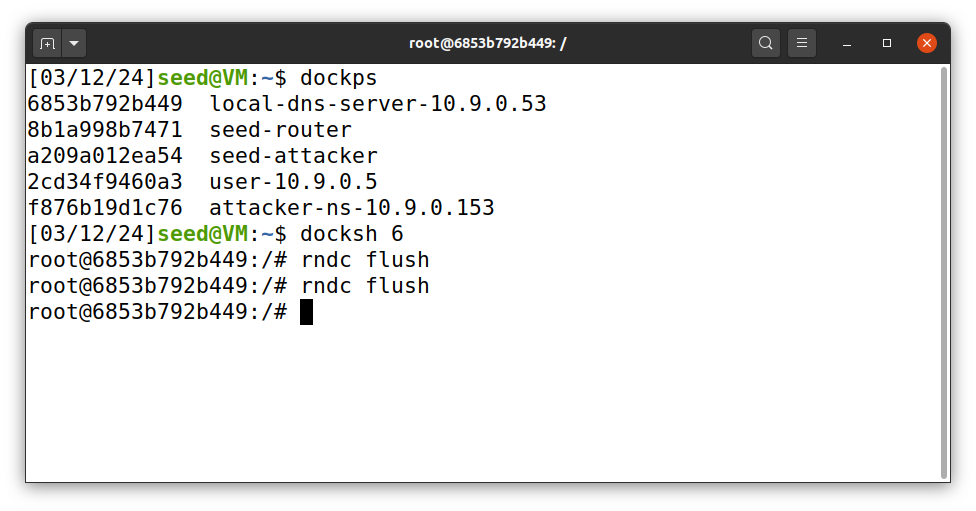
Lastly, we run the dig command on ns.attacker32.com . This works too, so we can see the IP result is given from attacker’s name server instead of local DNS server.

Thus we have successfully done the lap setup.

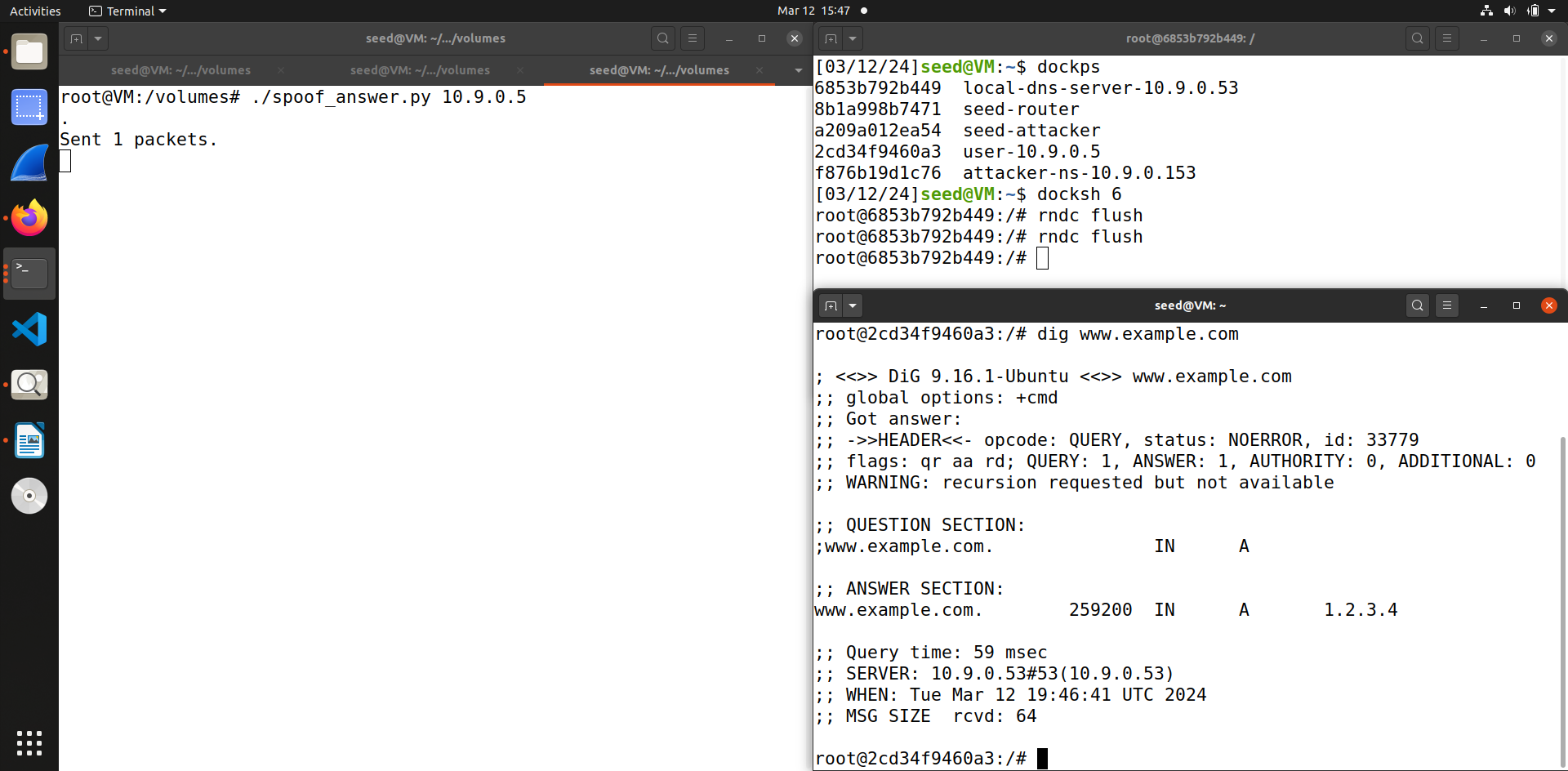
**Task 1 : Directly Spoofing Response to User**



So this is the code we will use to spoof the DNS response to the user.

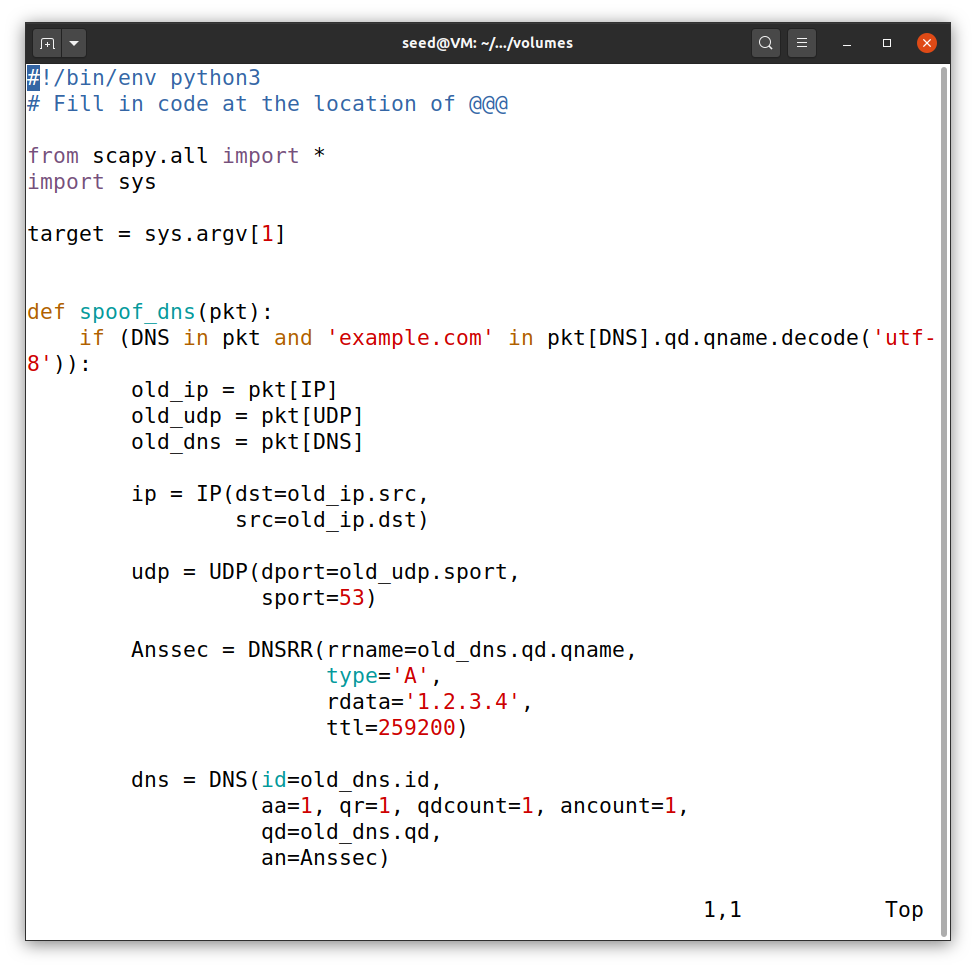
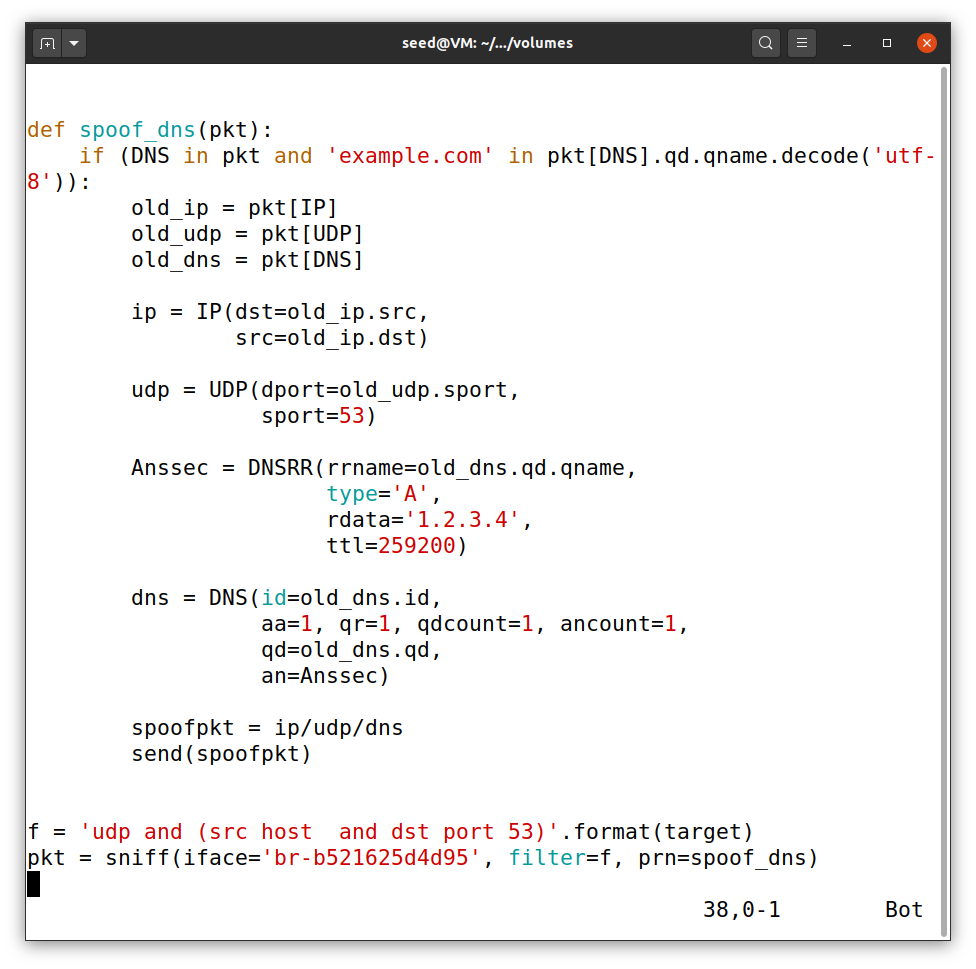


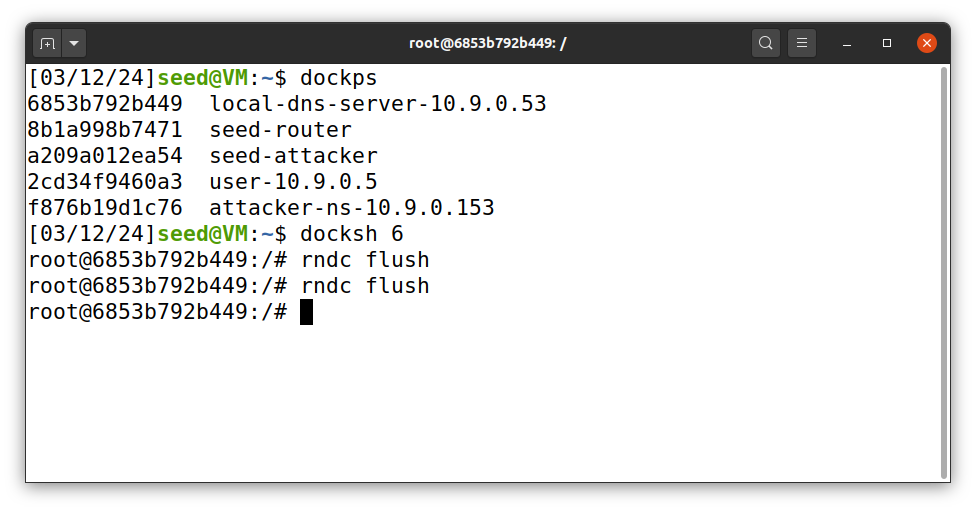
So the first step is to run ‘rundc flush’ command on the local dns server so that all the cache is clear and then we can run the dig command.

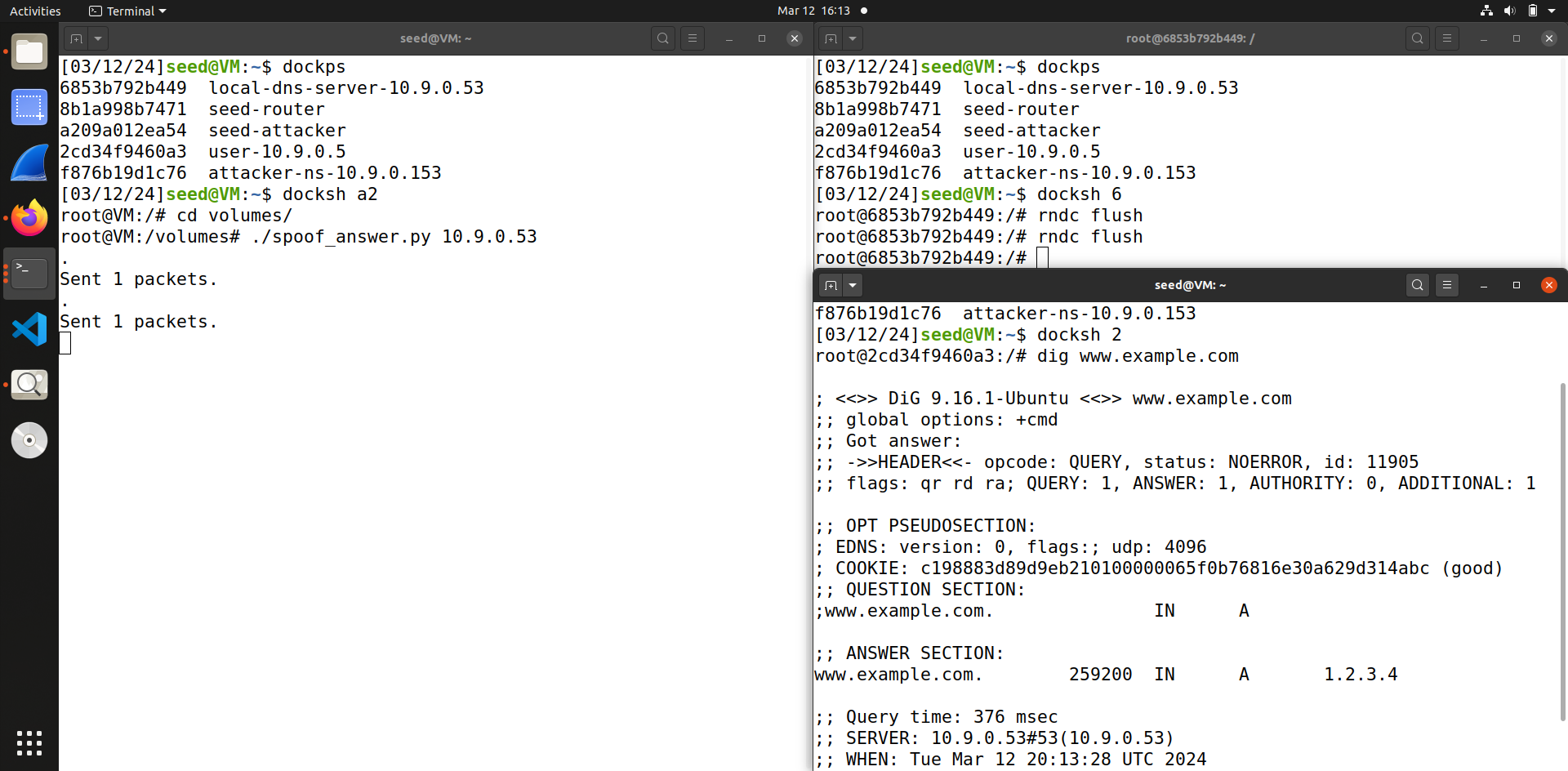
Then we run the program on on the seed attacker, and then run dig command on the victim’s container. As you can see from the screenshot the IP address for [www.example.com](http://www.example.com/) is spoofed and is showing it as 1.2.3.4.

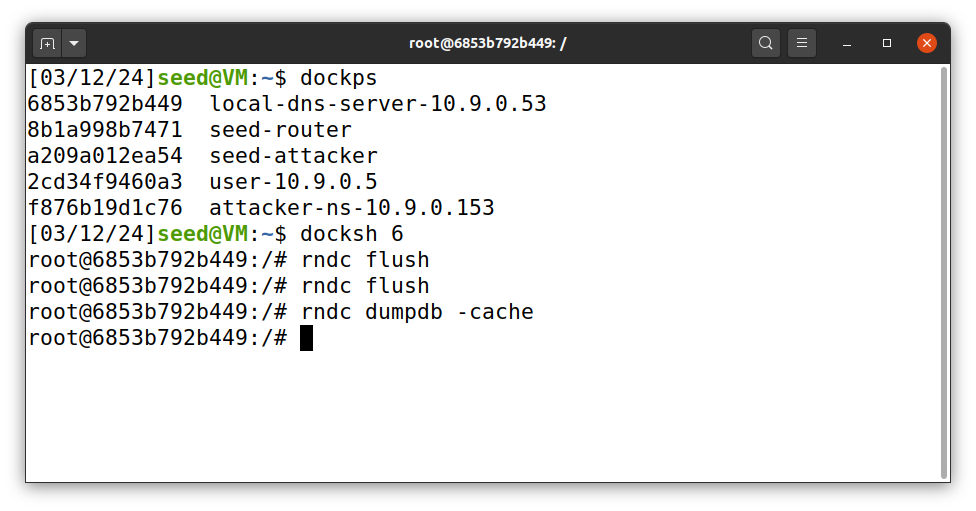
Thus we have successfully spoofed the DNS reply to the user.

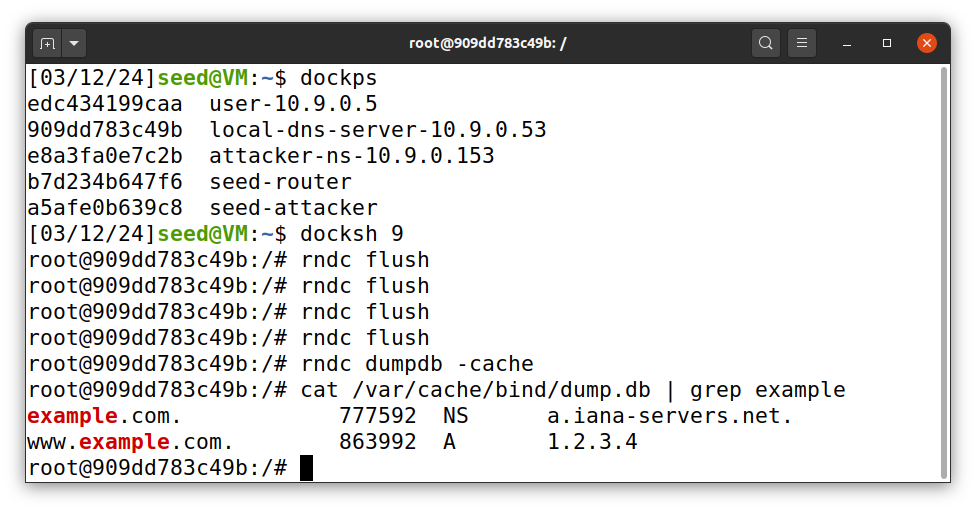
**Task 2: DNS Cache Poisoning Attack – Spoofing Answers**

For this task we will use the same code that we used in the previous task.

Now once again we clear the cache by running ‘rndc flush’ command before we run the dig command on user container.

So we do the same steps as we did in the previous assignment we, run the code on the seed attacker and then run the dig command on the user container, so as you can see the attack was successfully as the answer section shows spoofed reply.

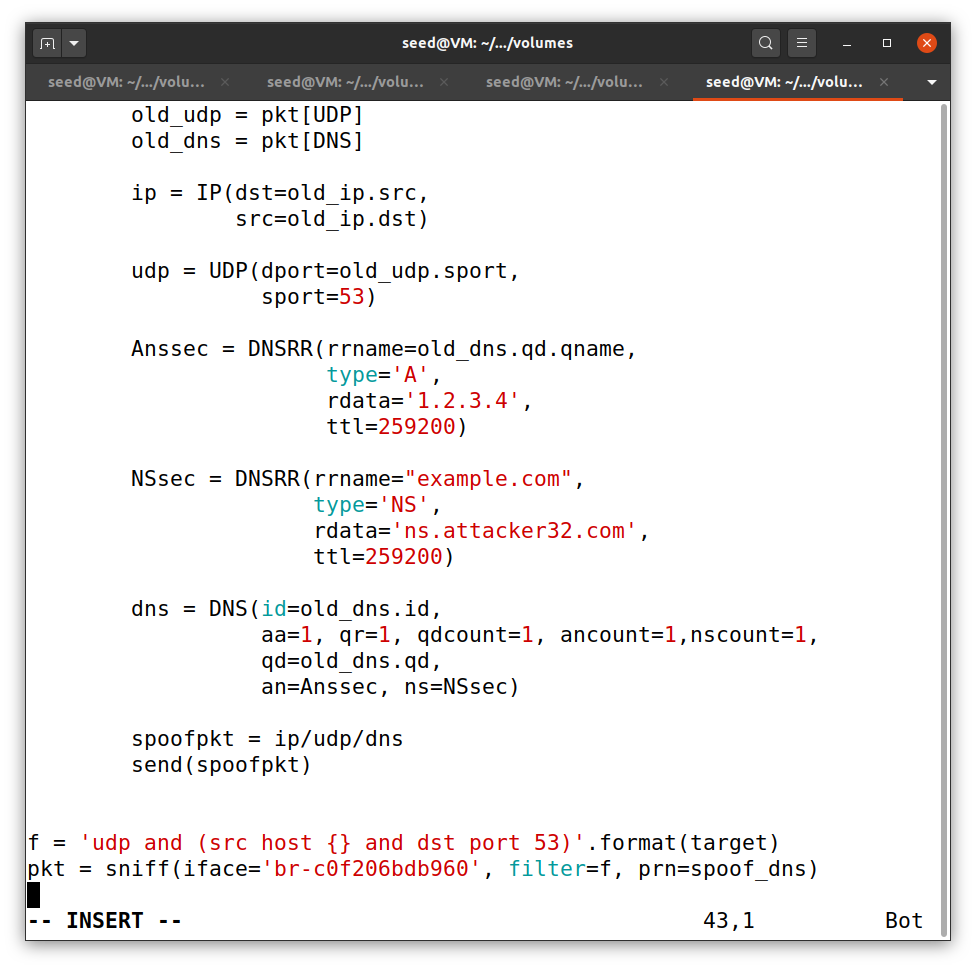
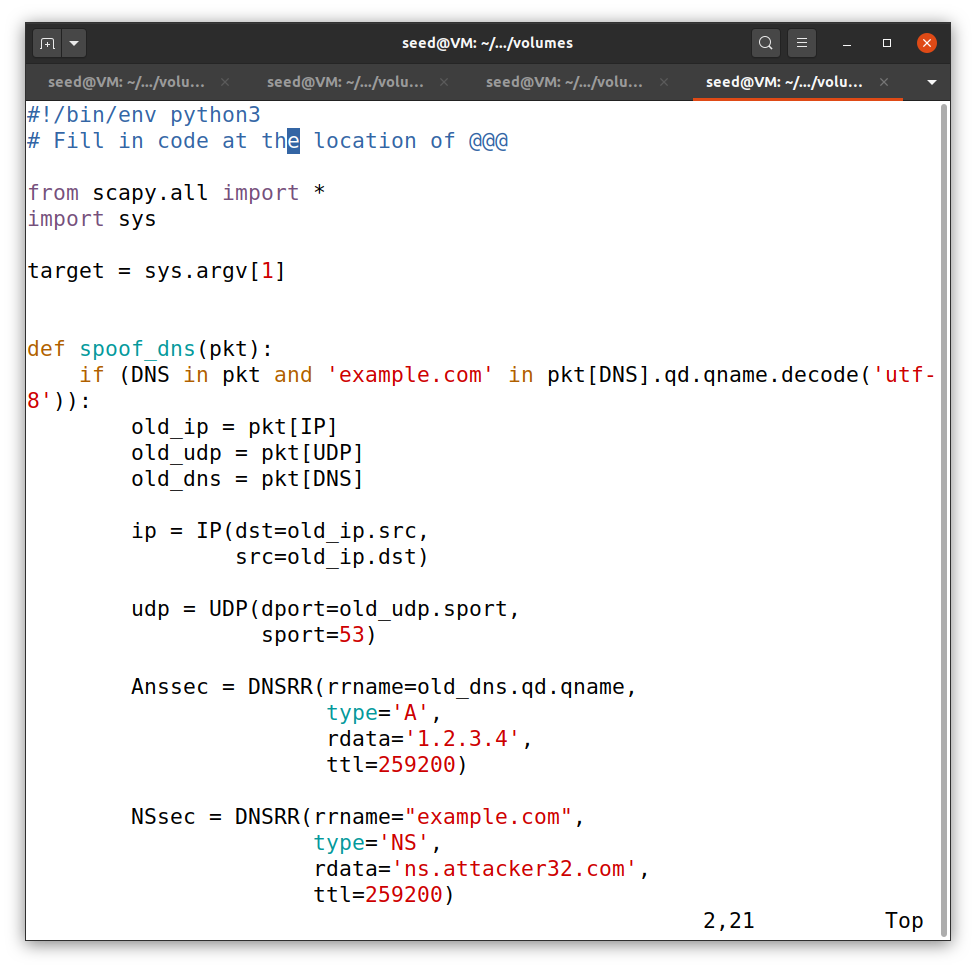
Now we run the dump cache command on the local DNS server to check if it was poisoned

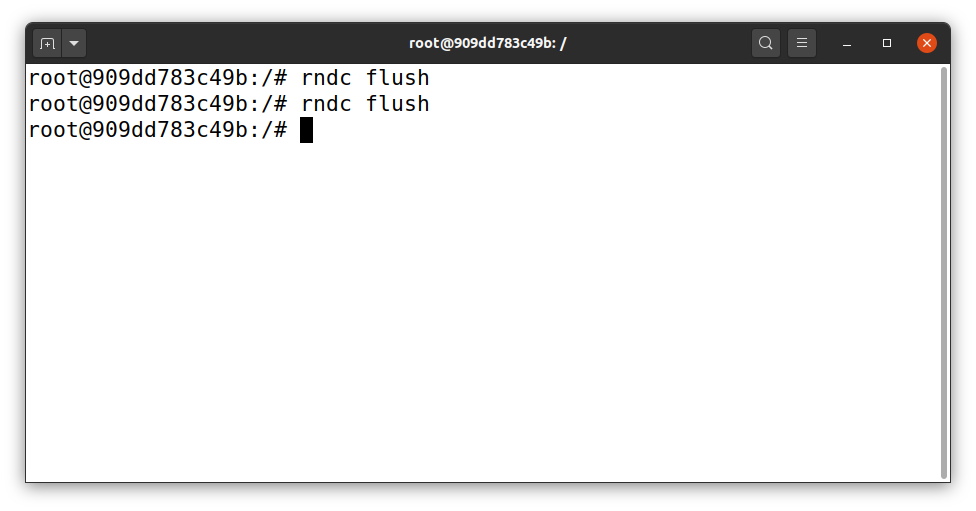


Now I use the grep command to filter out the entries of example, using the cat command as without the grep command the dump is too large to see.

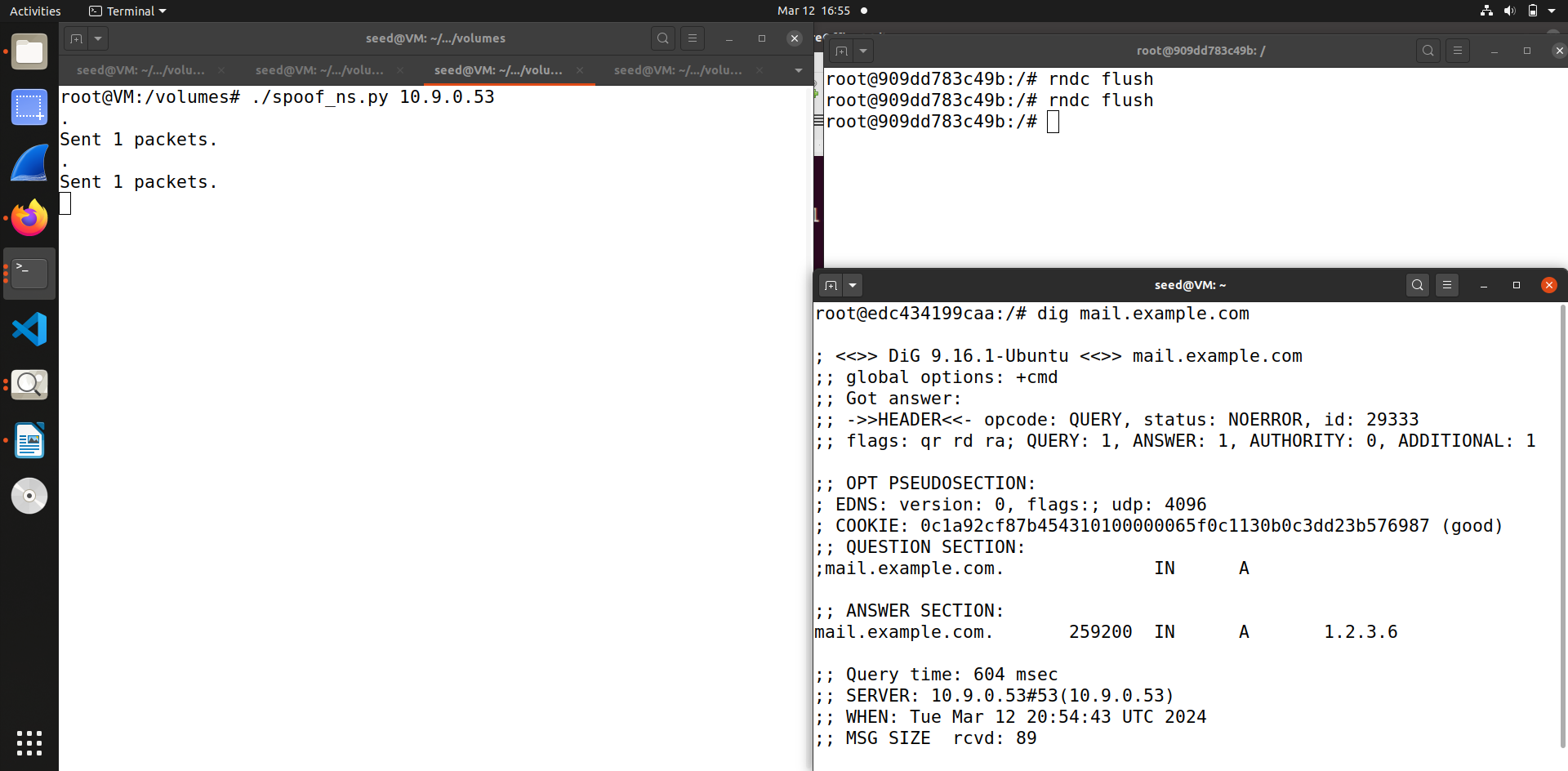
Thus we can see that we have successfully poisoned the local DNS server cache.

**Task 3: Spoofing NS Records**

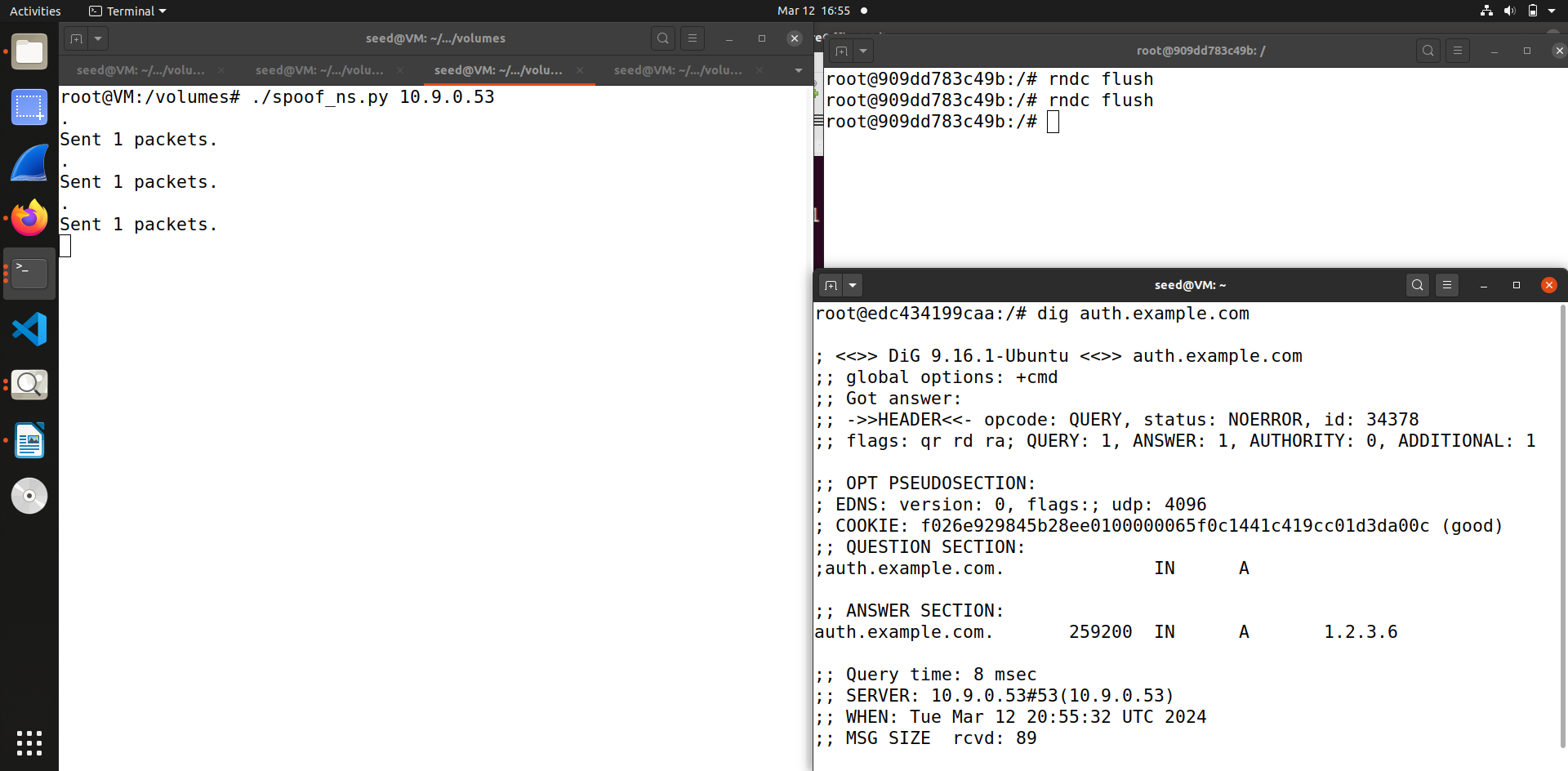
So this is the code we will use for this task to poison the DNS cache and modify the NS record too. So when this entry is cached by local DNS server, ns.attacker32.com will be used as nameserver .

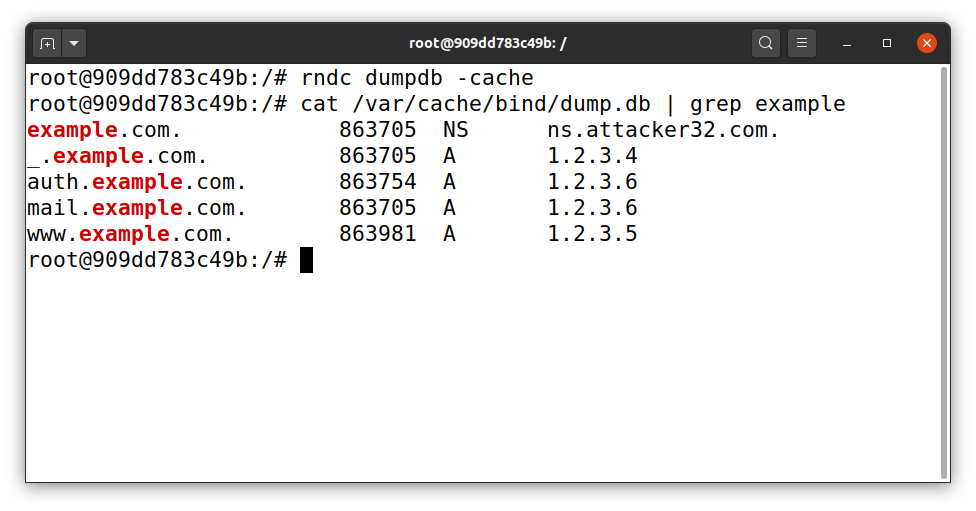


Now once again firstly we clear the cache with rndc flush command.



Now we run the program on the seed attacker and then execute the dig command on the user container with any hostnames in the example.com, so as you can see from the screenshot I changed the host name - the first I used is mail.example.com which also has the spoofed reply.

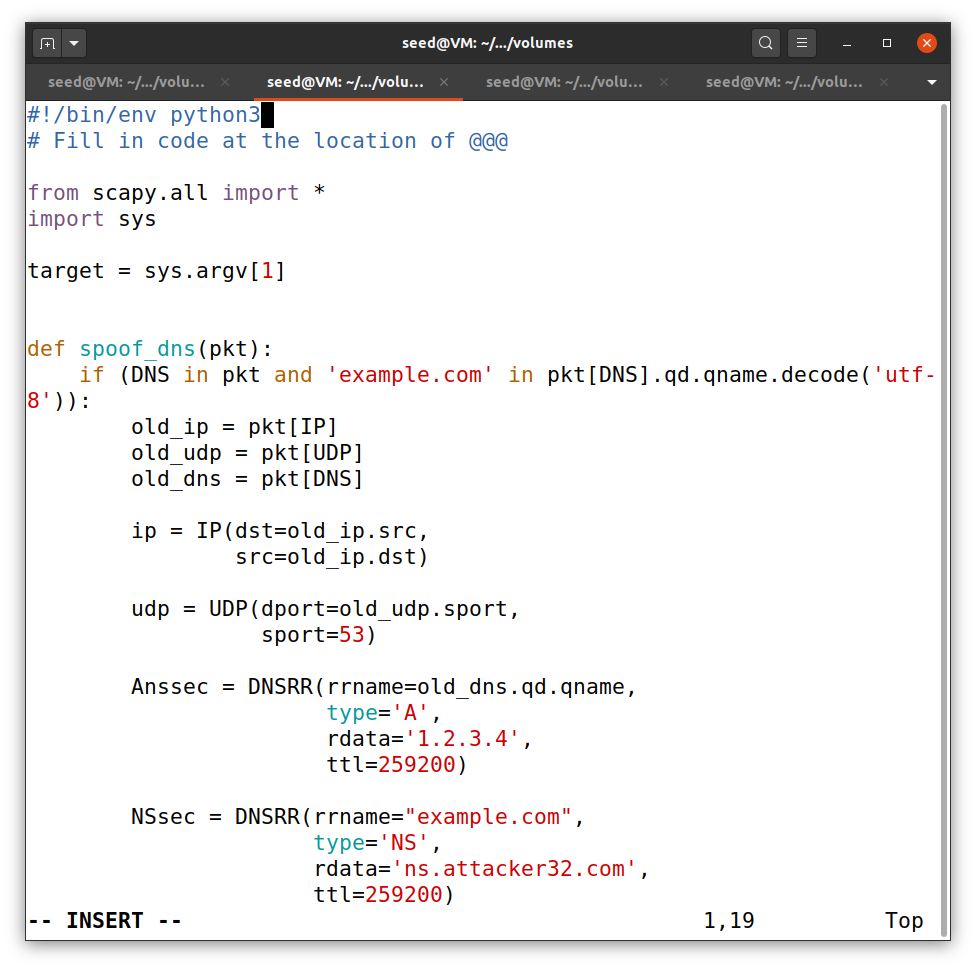
The second hostname I used is auth.example.com which also contains the spoofed reply.

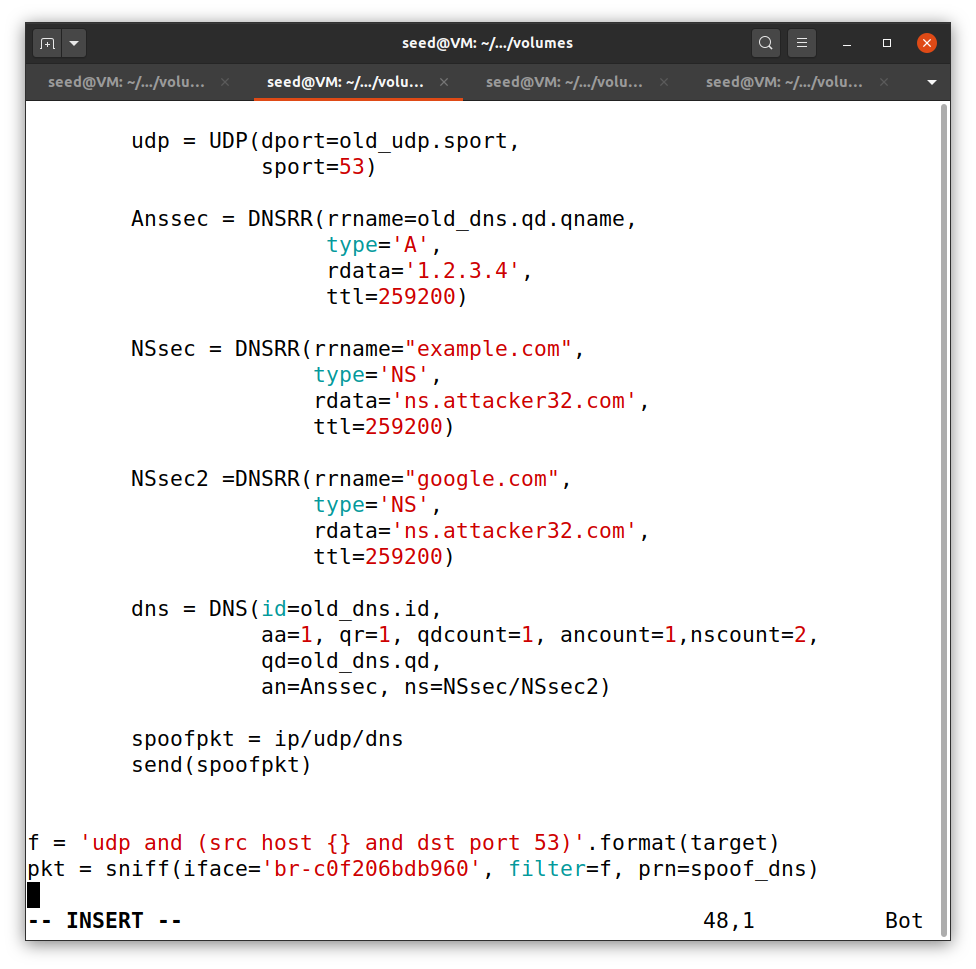


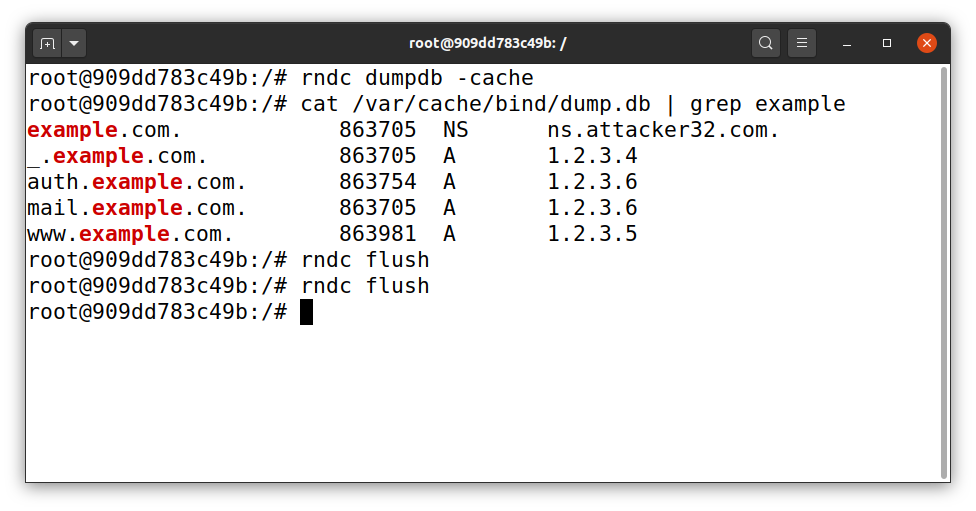
Now when we run the dump cache command then use the grep command to filter example.com

Thus as you can see from the screenshot, the local DNS servercache shows attacker’s spoofed replies, so the attack was successful and we have poisoned the cache along with modifying the Name server.

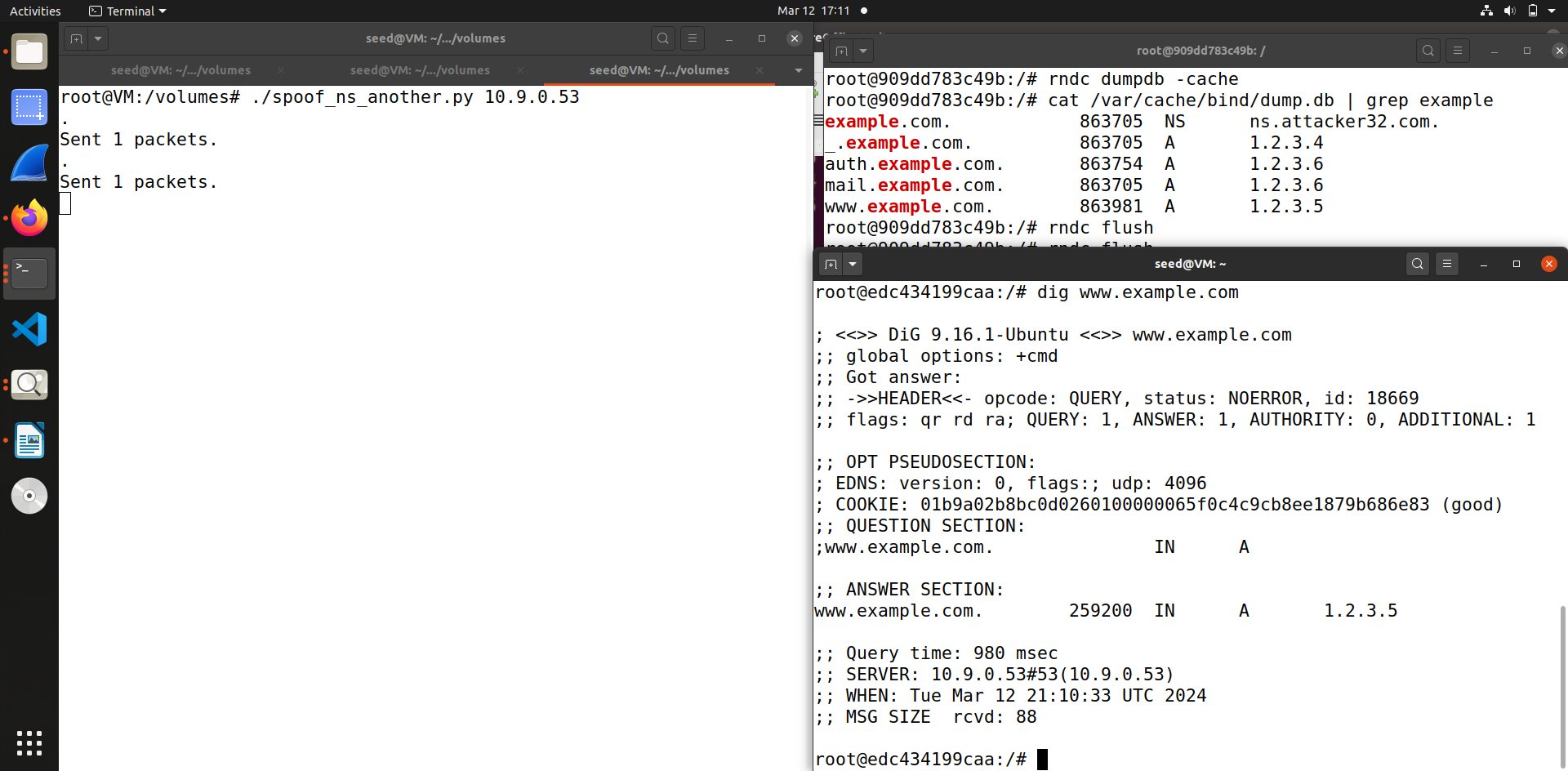
**Task 4: Spoofing NS Records for Another Domain**



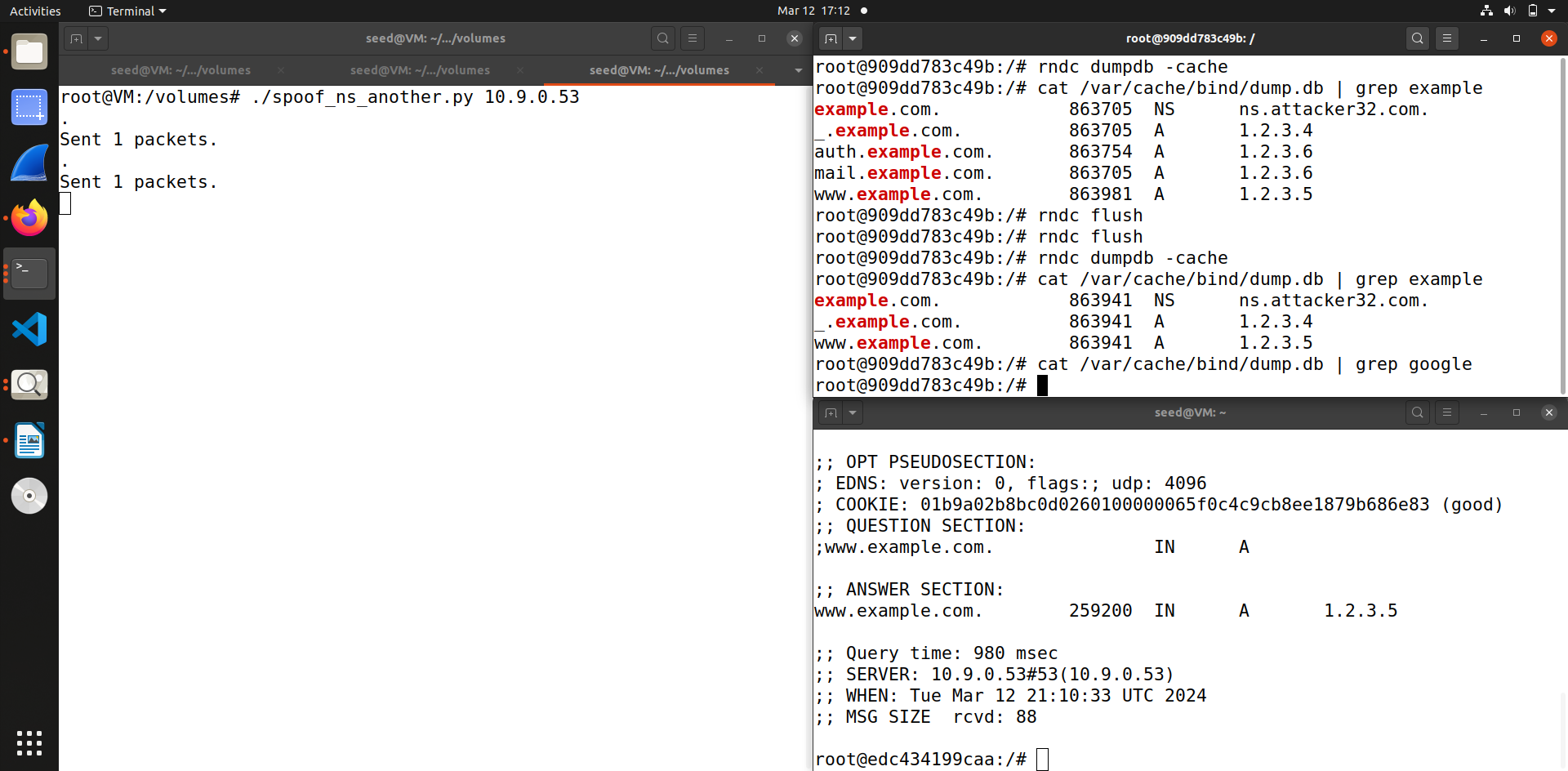
This is the code we will use in this task to poison the cache and add an additional NS record for google.com along with example.com.



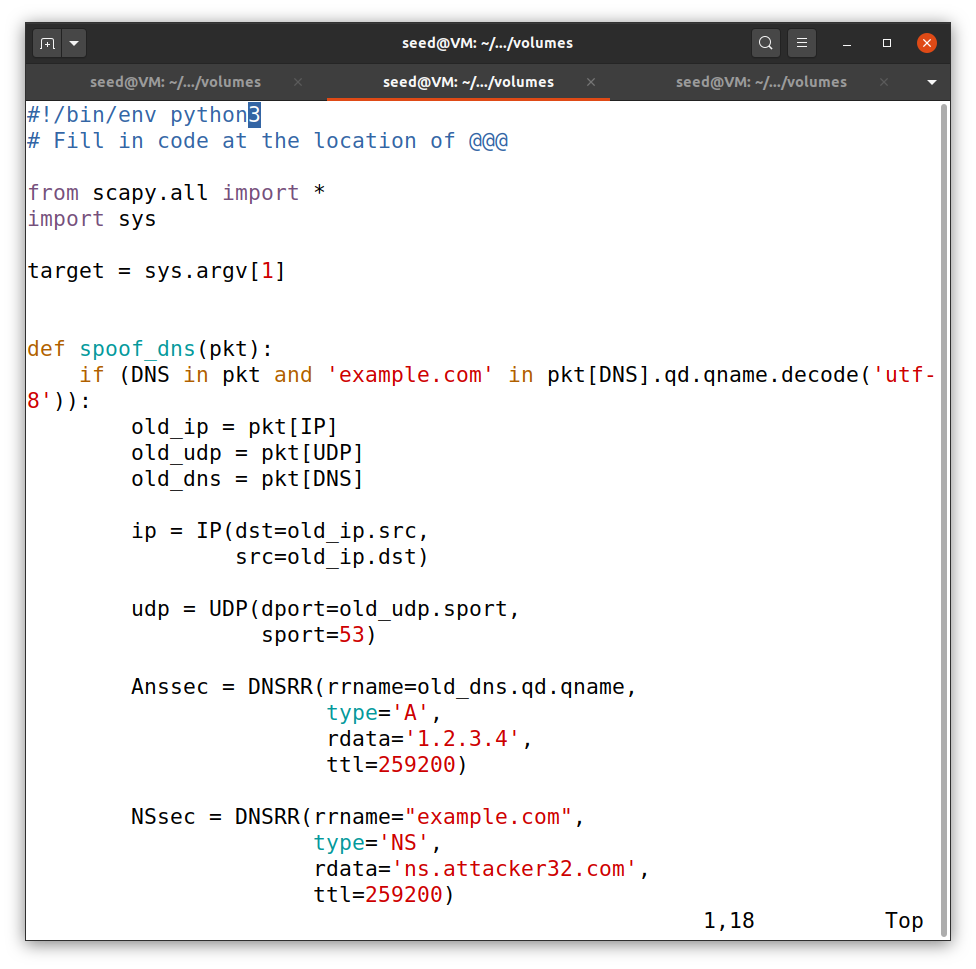
Like all tasks , first we execute the rndc flush command to flush out the cache and then run the dig command on user container.

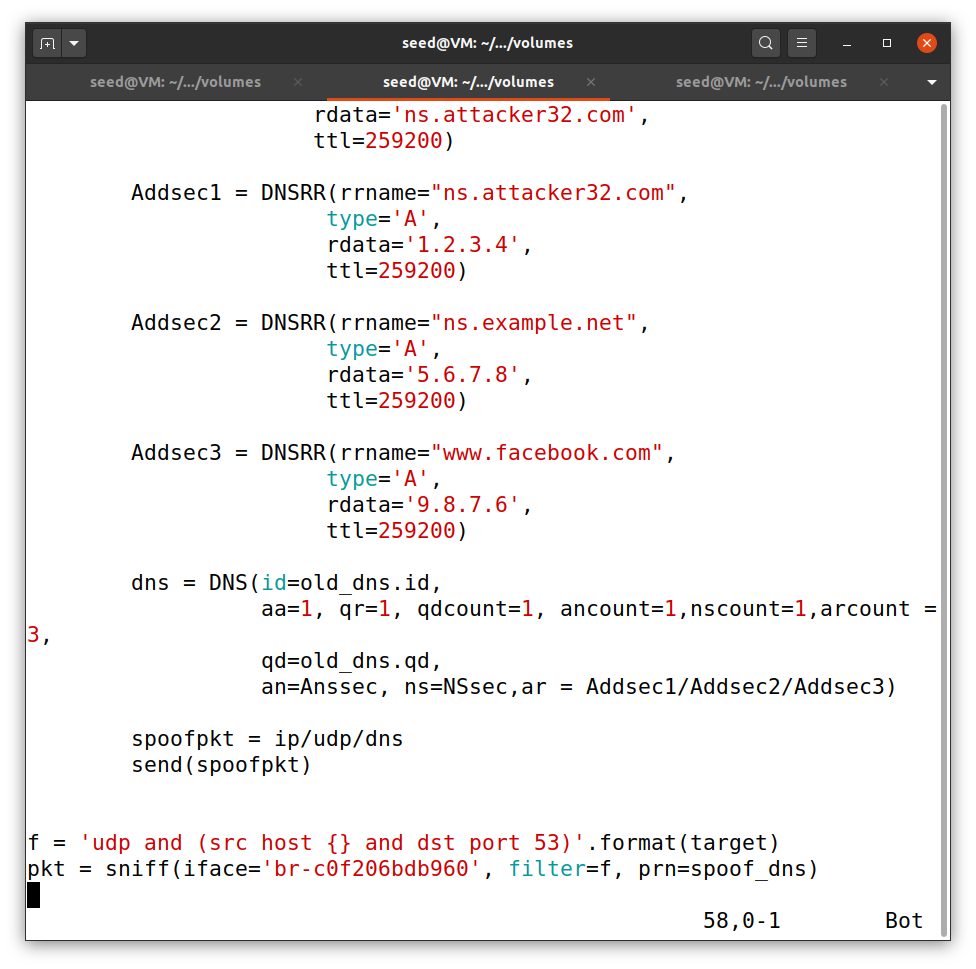


So after that I ran the attack code on the seed attacker and then execute the dig command on user machine with the hostname of [www.example.com](http://www.example.com/) . As you can see from the screenshot the answer section is successfully modified with the spoofed reply.

Then we run the dump cache command and then using grep command filter out the example.com and google.com. So as you can see from the screenshot the cache only shows result of example.com and no result of google.com this is because there is no configuration of this in this labsetup, we only have written the code to modify the record of example.com and not google.com so we get no result when we run the cache command for google.com. But the attack was successful.

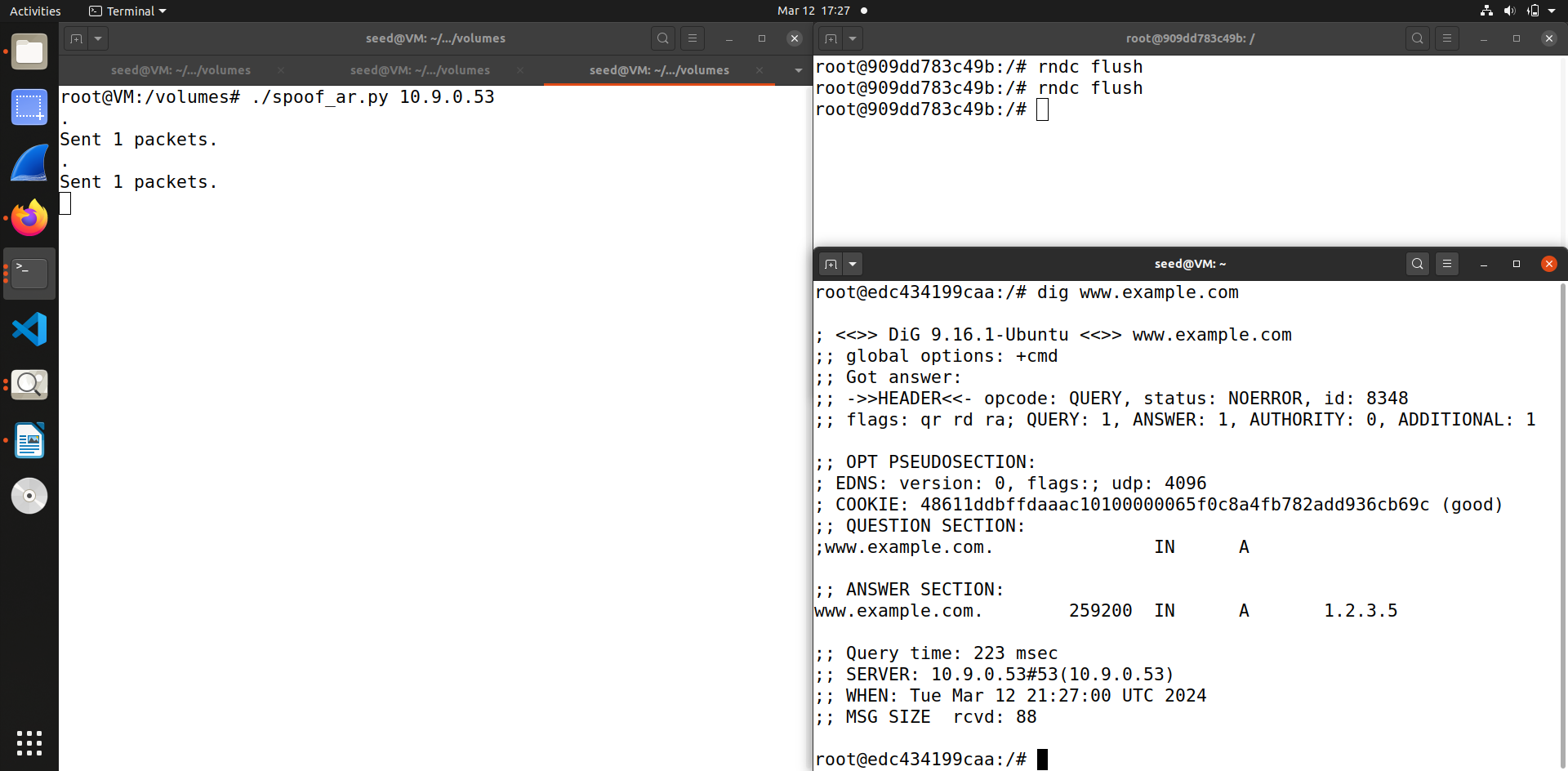
**Task 5: Spoofing Records in the Additional Section**

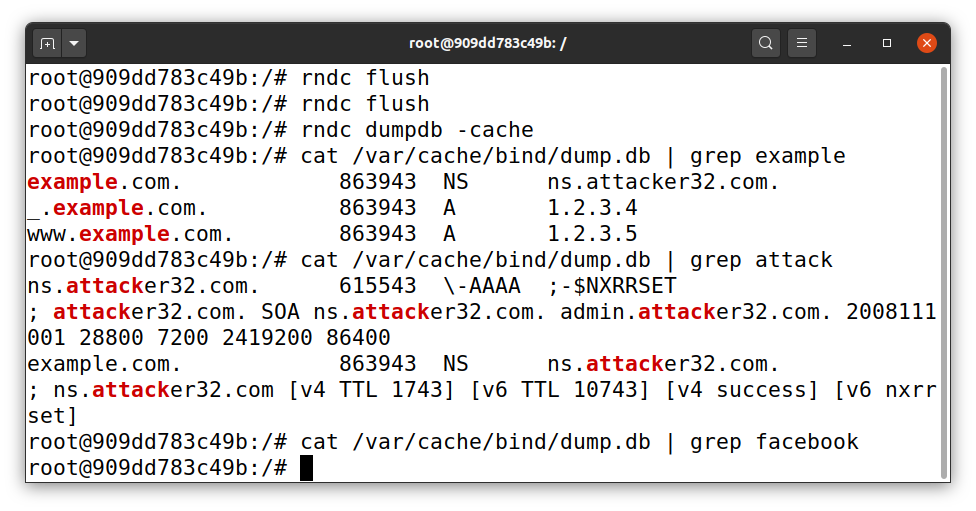


This the code we will use in this task to spoof some entries in the additional section and see whether they will be successfully cached by the target local DNS server. So in this code we have added 3 new records.



As we did on all the task we run the rndc flush command to clear the cache before running the dig command on victim machine.

Now we run the attack code and then execute the dig command on the user machine, which as you can see has the spoofed answer section.

The last step is similar like other task we run the dump cache command and the using grep command filter out the cache.

Thus you can see our attack was successful, and we have the additional section as well, which has been cached by the DNS server. As you can see from the screenshot we captured the entries of example.com , but there is not entry or result for the other additional section entries such as facebook.com or attacker32.com also is not cached as there was no configuration done for those section, only example.com was been configured.