**Crypto And Network Security**

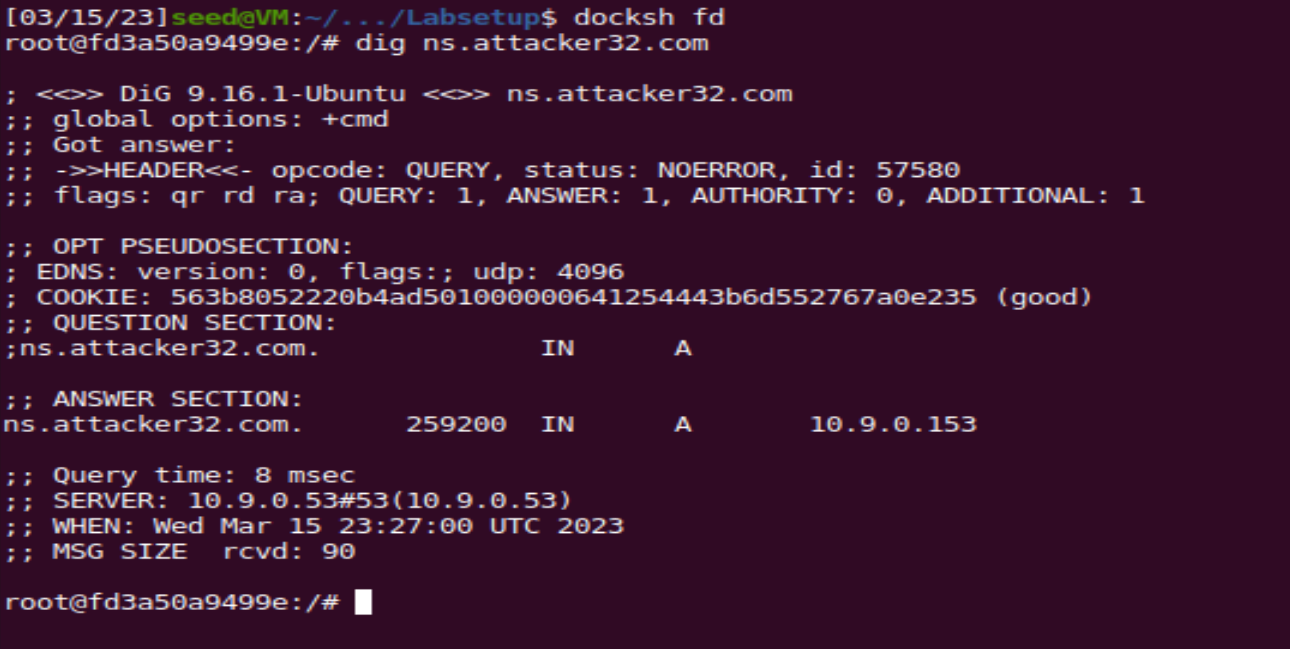
**Lab 7**

**Name: GAURAV SETTY**

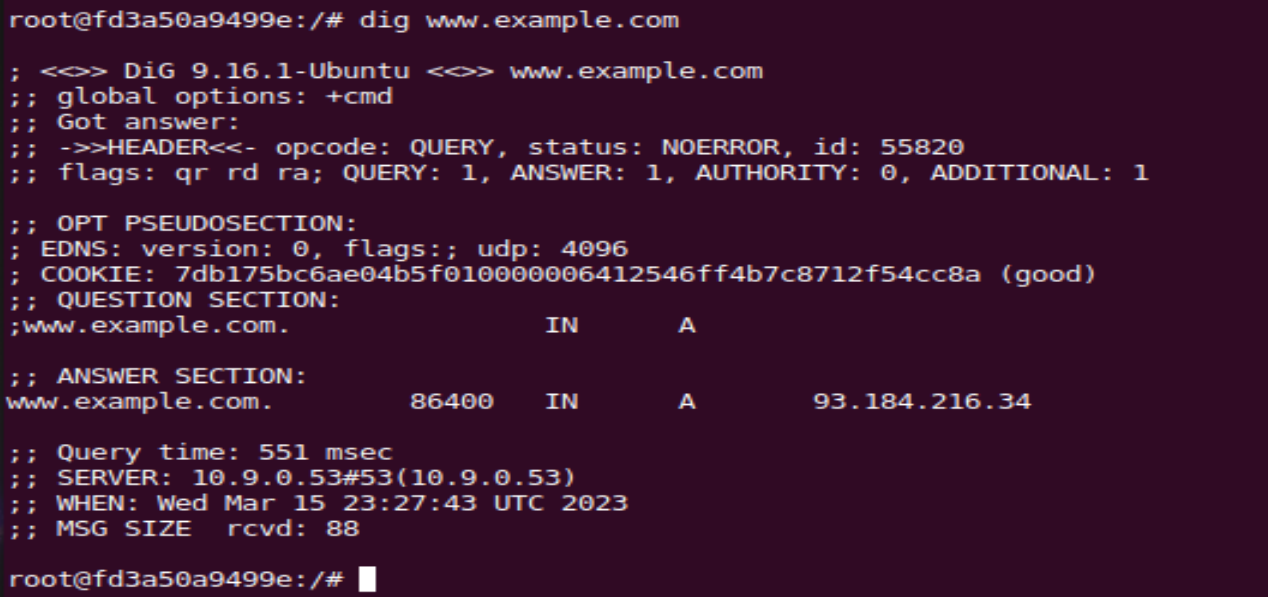
**Email:** [**settgm01@pfw.edu**](mailto:settgm01@pfw.edu)

**Tests:**

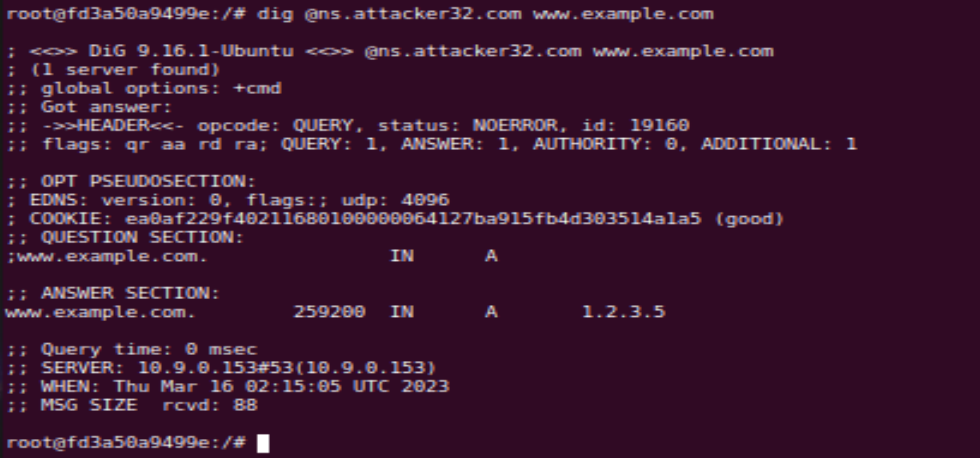
Fetching IP address of ns.attacker32.com (10.9.0.153)



Fetching IP address of [www.example.com(93.184.216.34)](http://www.example.com(93.184.216.34)) from local DNS

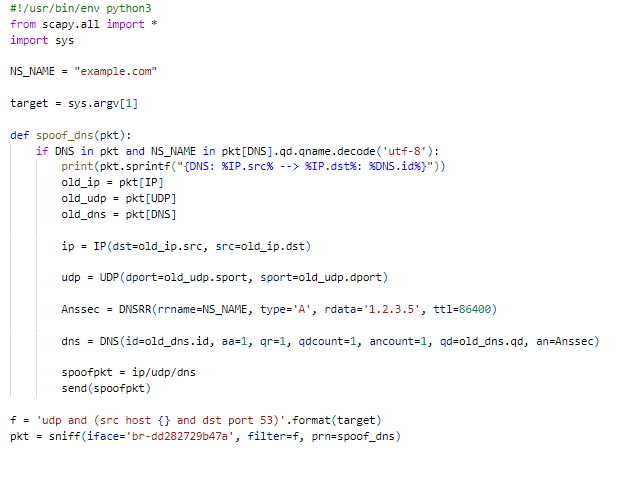


Fetching IP address of [www.example.com(1.2.3.5)](http://www.example.com(1.2.3.5)) from ns.attacker.com

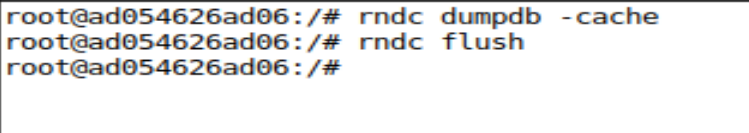


**Task 1: Directly Spoofing Response to User:**

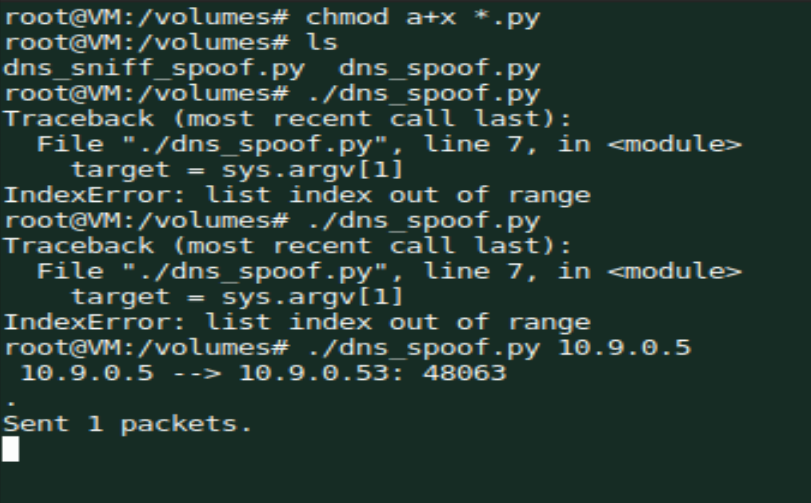
Code:



Flush the cache:



Now, run the program from attacker:

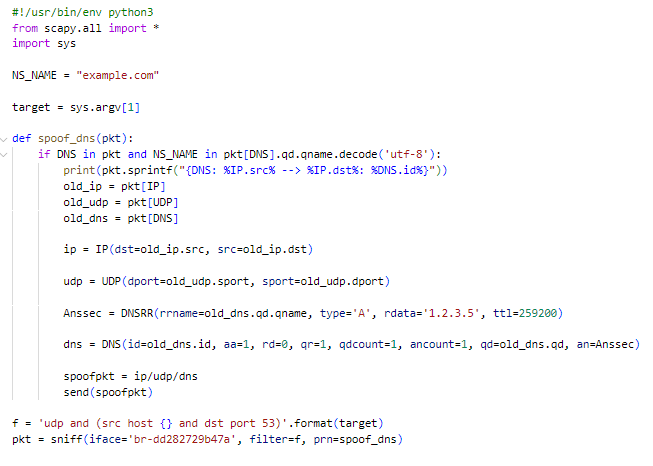


Now, my spoofed DNS response also returns a false IP address when using the simple dig command.

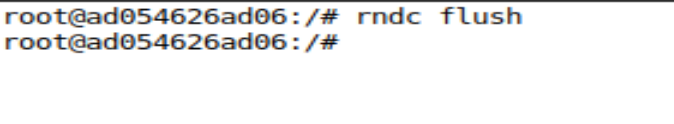


**Task 2: DNS Cache Poisoning Attack – Spoofing Answers**

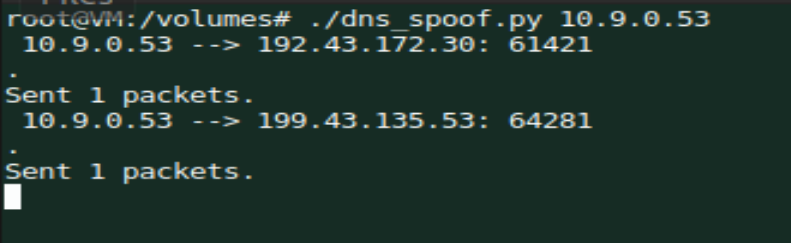
**Code:**



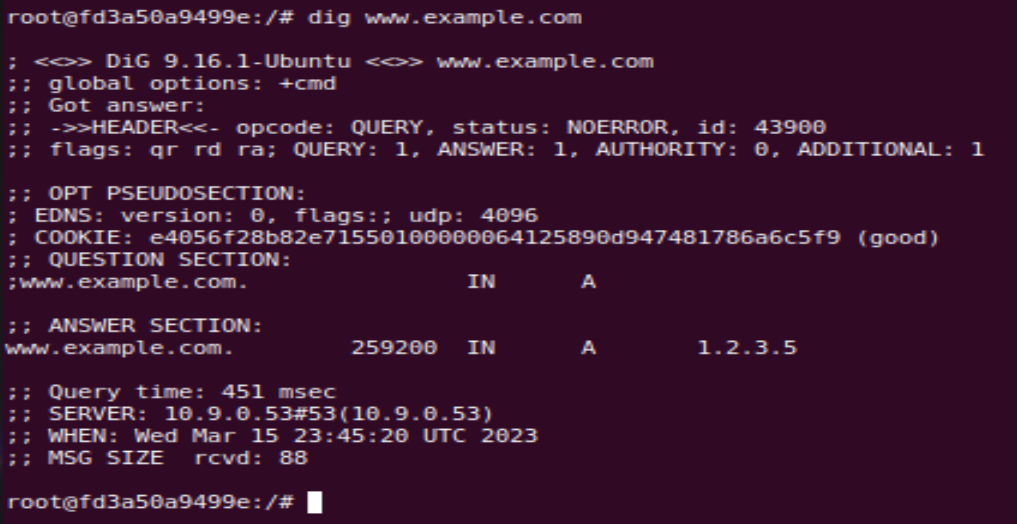
Flush the cache before running the code



Now, run the program from attacker’s machine

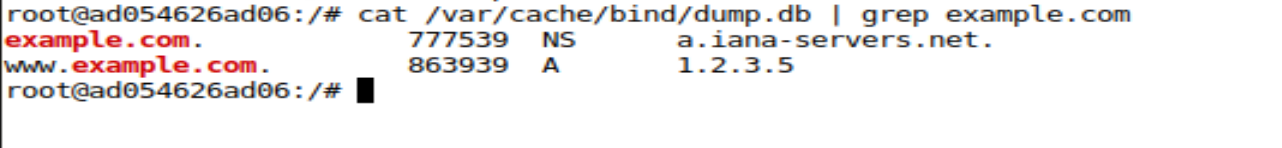


Dig [www.example.com](http://www.example.com) from user.



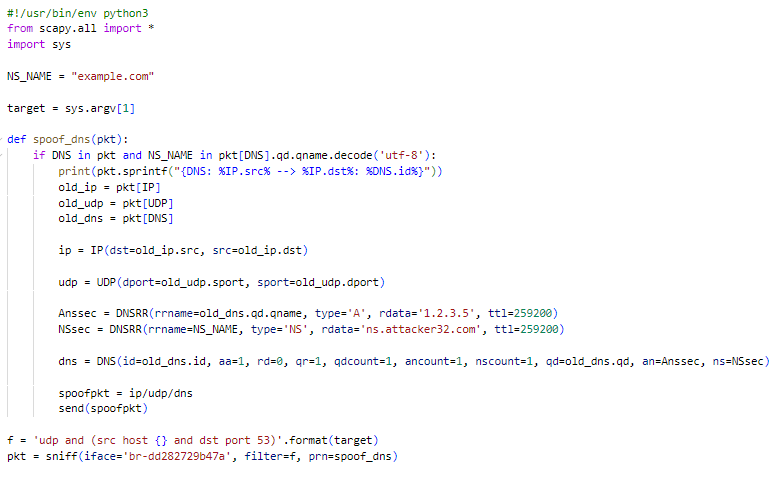
The DNS spoofer was attacking the server (10.9.0.53), and when user 10.9.0.5 sent a DNS request, the server wrongly cached the attacker program's spoof DNS response.

Now, let’s look at the cache.



**Task 3: Spoofing NS Records**

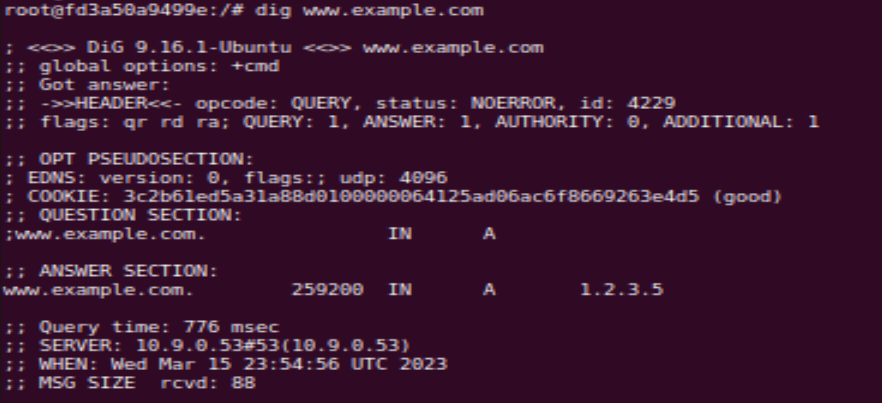
**Code:**



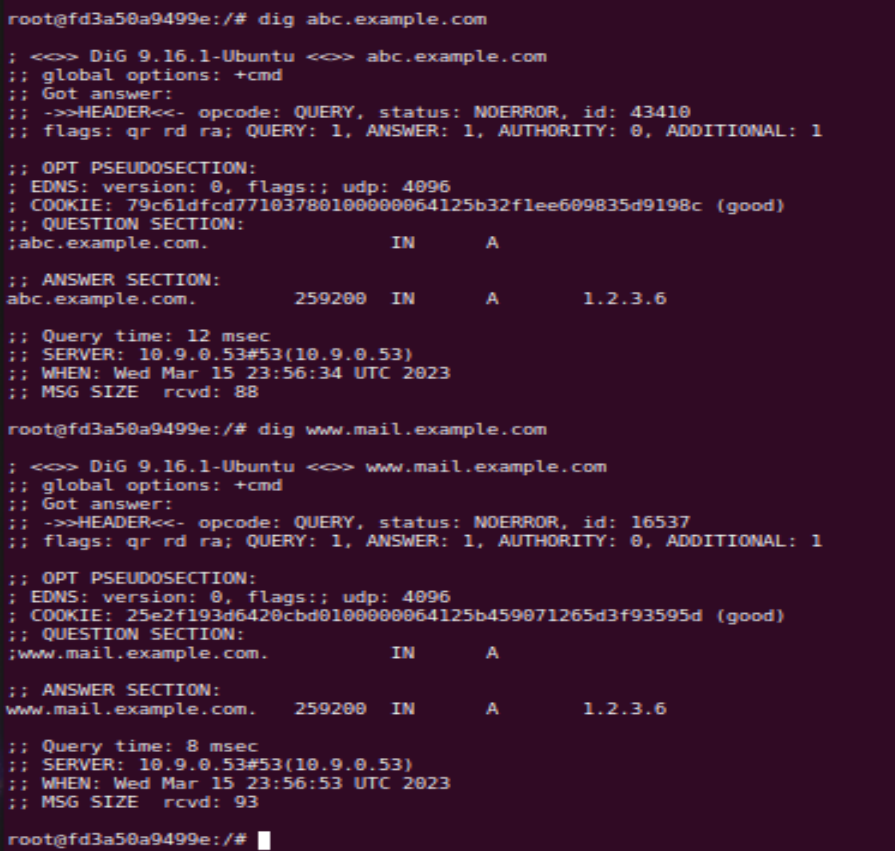
Flush the cache and run the code.



Using dig example to get the IP address:

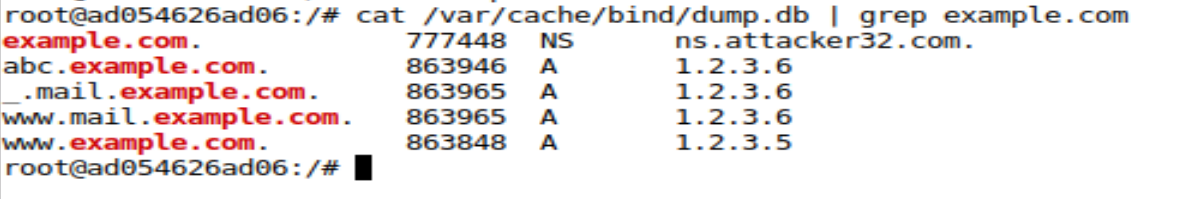


Running dig on abc.example.com.



It gave the spoofed IP!!

Let’s check the cache,

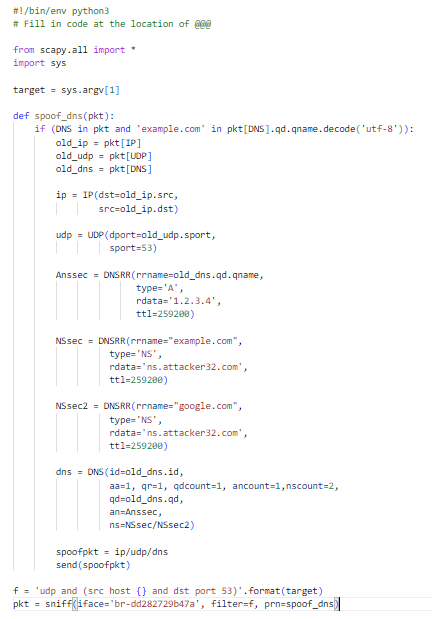


We can see that a new server name has been added (ns.attacker32.com).

The local DNS server, which has a poisoned cache, is responding to the query with false information via the attacker’s name server. That is why the result shows 1.2.3.6, as can be seen.

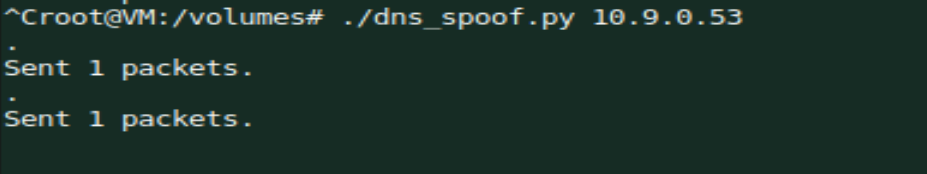
**Task 4: Spoofing NS Records for Another Domain**

**Code:**

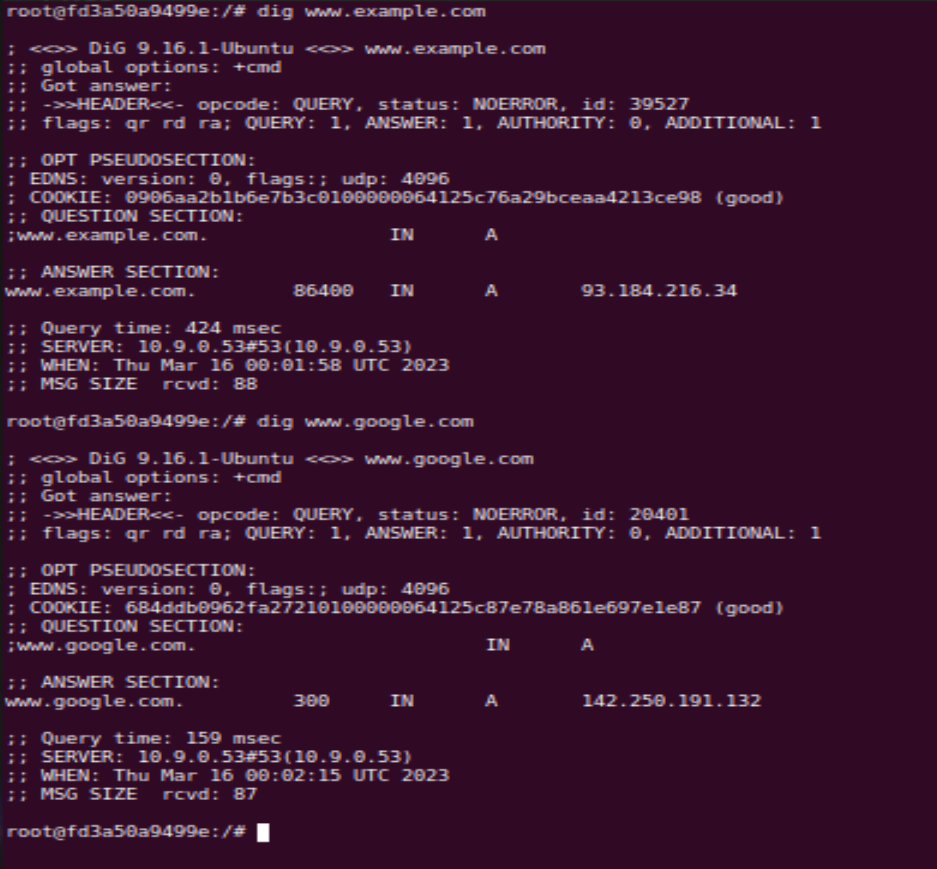


Flush the cache and run the program,

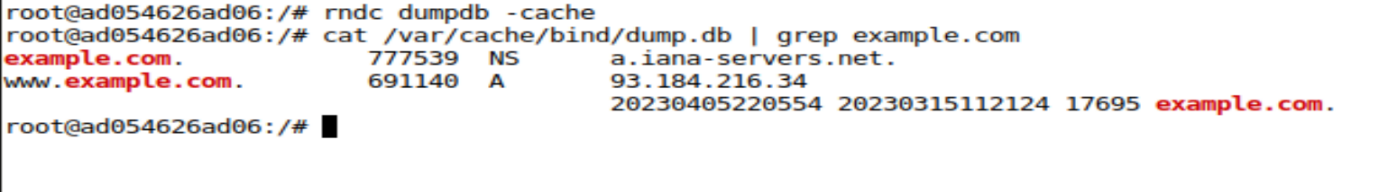


****

Let’s dig [www.example.com](http://www.example.com) and [www.google.com](http://www.google.com) and check what happens.



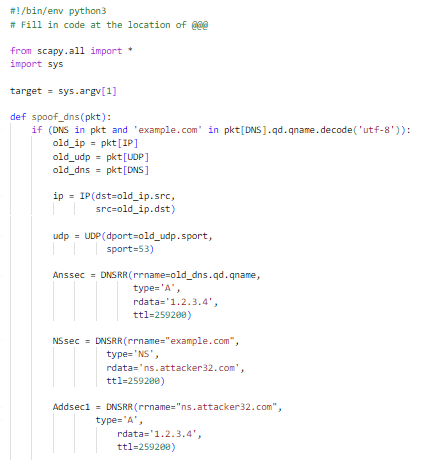
Now, let’s check the cache.



Google.com is not cached, but example.com is. This is because the DNS server used by the attacker is only configured for example.com

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

**Code:**

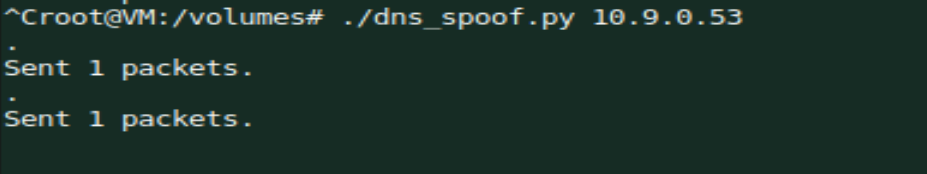




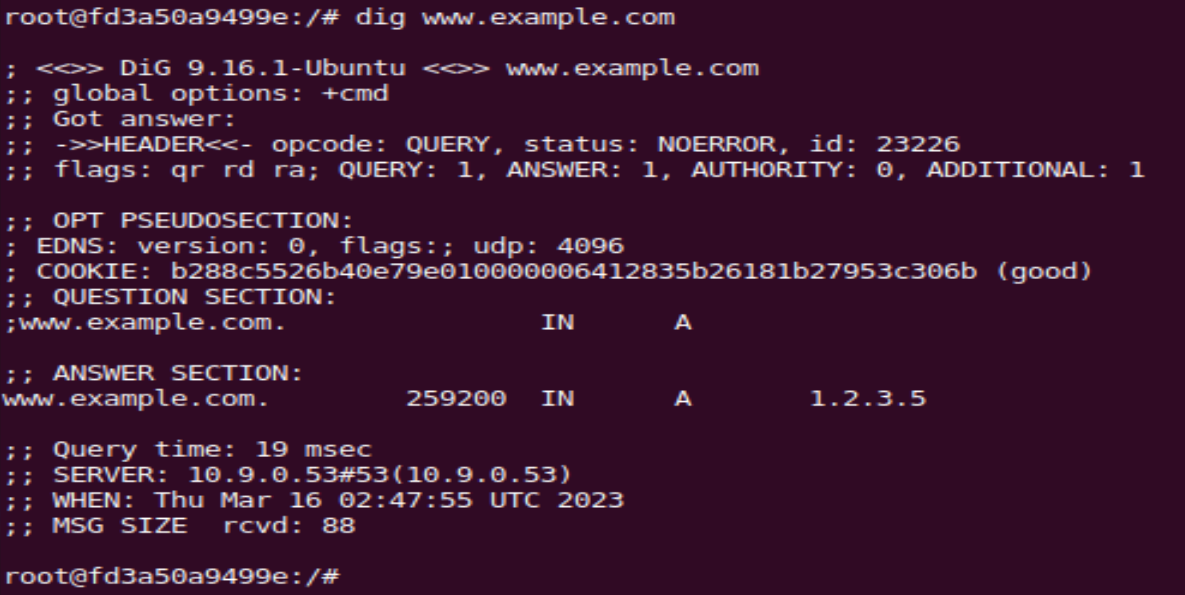
Now, flush the cache



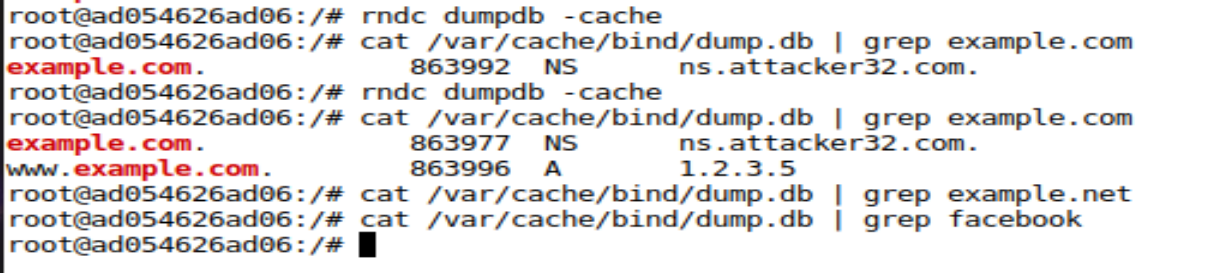
Run the program.



Let’s dig [www.example.com](http://www.example.com)



Checking the cache,



We can see that there are no cache entries for facebook or example.net, this implies that attack has failed as there aren’t any configurations for domains other than example.com