# Local DNS Attack Lab

**Submitted by: Mahika Gupta**

**SRN: PES1UG20CS243**

**Date: 09/10/2022**

# Lab Environment Setup

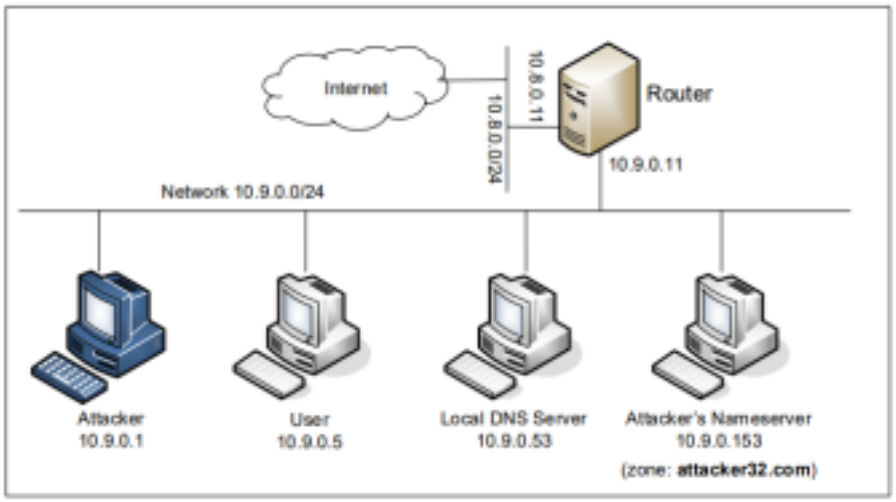



Figure 1 : Lab Environment setup

```
PES1UG20CS243:Mahika~/.../Labsetup$>dockps
724547e0a713  seed-attacker
b87483dc98ce  user-10.9.0.5
2e0ad912a404  seed-router
cb9c6ea4495b  attacker-ns-10.9.0.153
d1c8bdfc1a84  local-dns-server-10.9.0.53
PES1UG20CS243:Mahika~/.../Labsetup$>
```

# 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**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32049
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 47315e8e93dc4a530100000063410ddb8172889667f793da (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 08 05:42:51 UTC 2022
;; MSG SIZE  rcvd: 90

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**We can see that the setup is working.**

**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**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26809
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b0d4ddf39a04f8d80100000063410e2f8dc6a21c693b3e50 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1672 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 08 05:44:15 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**On using dig command for example.com, the query is directed to the local DNS server, which returns the IP address of example.com in the Answer section.**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig @ns.attacker32.com www.example.com

; <<>> 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: 49759
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7673d00702d11e580100000063410e66f6b39a116500a0d5 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sat Oct 08 05:45:10 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**On using dig command directed at ns.attacker32.com nameserver, we get the response with the spoofed IP address set by the attacker for example.com in the Answer section.**

# 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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>rndc flush
```

**This clears the local DNS cache to avoid response from local DNS being received before the attacker's spoofed response.**

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**:

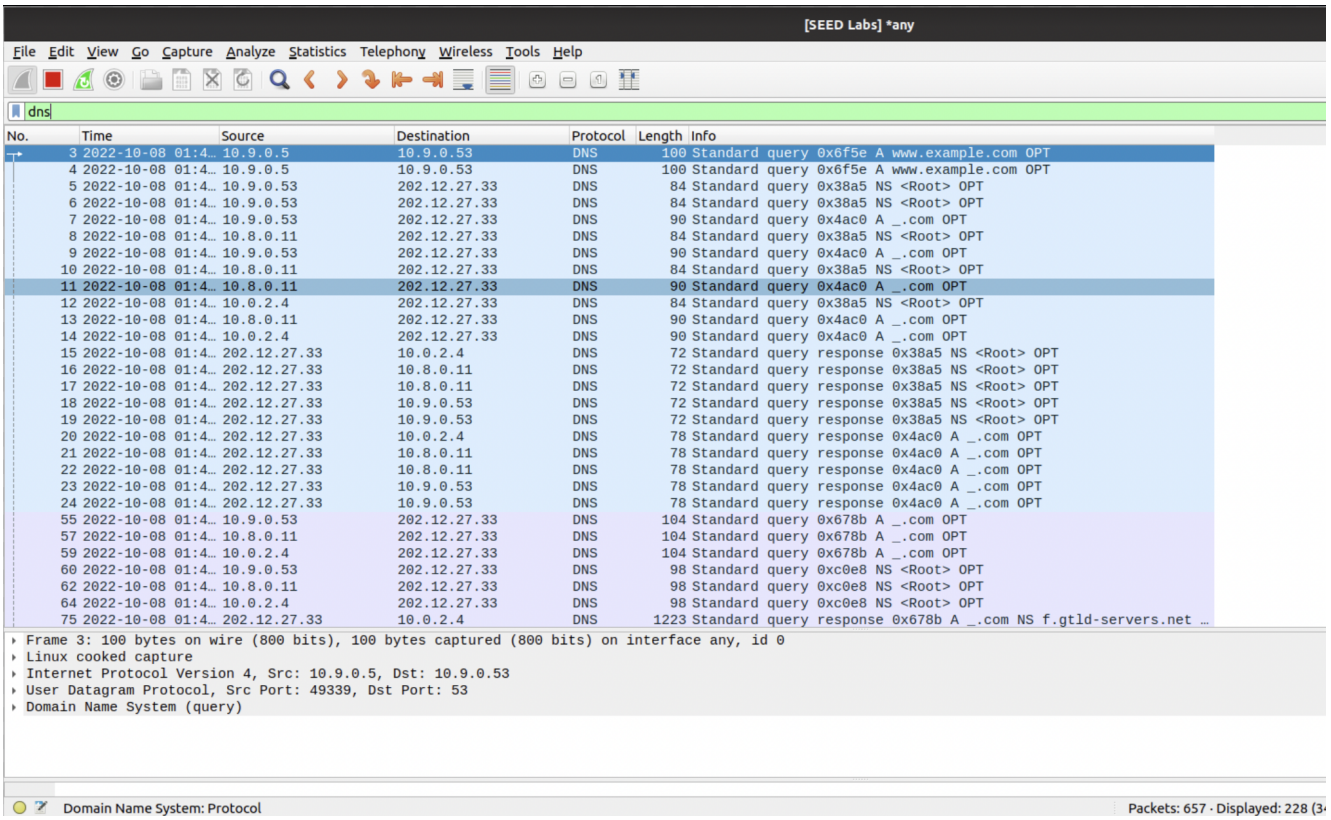
**# dig www.example.com**

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 28510
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 176abd6db04d2bd30100000063410f2471ff08c4a4ba6bb8 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; Query time: 1720 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 08 05:48:20 UTC 2022
;; MSG SIZE  rcvd: 72

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**On using dig command for example.com, we can see that there is only a question section consisting of the query, and no answer section. In the additional section we can see that the query was directed to the local dns server at IP address 10.9.0.53.**




**We can see on wireshark that DNs query is sent to the local dns server. The local DNS in turn sends query to Root DNS for the address of .com domain. After receiving a response from Root DNS the local DNS then queries the Namerservers,and so on**

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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>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**

```
Terminal
Terminal  Terminal  Terminal  Terminal  root@
seed-attacker:PES1UG20CS243:Mahika/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        = 43001
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xbe54
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 36502
        dport     = domain
        len       = 64
        chksum    = 0x149d
###[ DNS ]###
           id        = 3932
           qr        = 0
```

```
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 4096
            |  extrcode  = 0
            |  version   = 0
            |  z         = 0
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\x83\x9b\x8a@\x93\x95~\xa5'

.
Sent 1 packets.
```

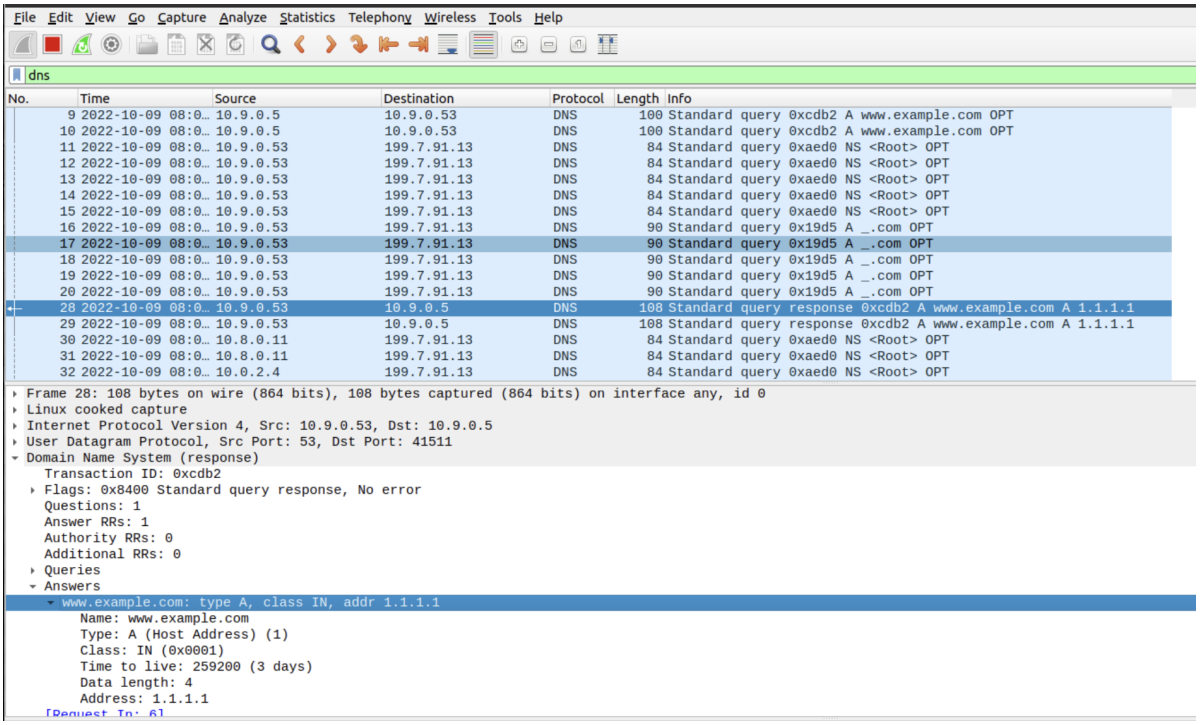
**On running the attack, we can see that a spoofed query response is sent to the victim machine, which displays the IP address of example.com as 1.1.1.1**

**On the victim terminal run the command**:

**# dig www.example.com**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3932
;; 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: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 08 05:52:21 UTC 2022
;; MSG SIZE  rcvd: 64

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**We can see that the Answer section contains the spoofed IP address of example.com, which was sent by the attacker.**




**On wireshark, we observe the DNS query response packet, which the attacker sends, appears to be arriving from the local dns, and returns the IP address of example.com as 1.1.1.1 in the answer section.**

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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>rndc dumpdb -cache
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>cat /var/cache/bind/dump.db | grep example
example.com.            691109  NS      a.iana-servers.net.
                                        20221022214625 20221001223409 1686 example.com.
www.example.com.        691109  A       93.184.216.34
                                        20221022040544 20220930163209 1686 example.com.
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>
```

**Here we observe that the dns cache has stored the actual IP address of example.com which is received by the local dns. The attacker has not meddled with the local dns, and has simply sent a spoofed packet to the victim before the real dns response from local dns was received. Hence we see no change in the dns cache.**

## A potential issue

When we do this lab using containers, sometimes we see a very strange situation. The sniffing and spoofing inside containers is very slow, and our spoofed packets even arrive later than the legitimate one from the Internet, even though we are local. In the past, when we used VMs for this lab, we have never had this issue. We have not figured out the cause of this performance issue yet (if you have any insight on this issue, please let us know).

If you do encounter this strange situation, we can get around it. We intentionally slow down the traffic going to the outside, so the authentic replies will not come that fast. This can be done using the following tc command on the router to add some delay to the outgoing network traffic. The router has two interfaces, eth0 and eth1, make sure to use the one connected to the external network 10.8.0.0/24.

// Delay the network traffic by 100ms
**# tc qdisc add dev eth0 root netem delay 100ms**

// Delete the tc entry
**# tc qdisc del dev eth0 root netem**
// Show all the tc entries
**# tc qdisc show dev eth0**

You can keep the tc entry on for this entire lab, because all the tasks will face a similar situation

```
seed-router:PES1UG20CS243:Mahika/
#>tc qdisc add dev eth0 root netem delay 100ms
seed-router:PES1UG20CS243:Mahika/
#>tc qdisc show dev eth0
qdisc netem 8002: root refcnt 2 limit 1000 delay 100.0ms
seed-router:PES1UG20CS243:Mahika/
#>tc qdisc del dev eth0 root netem
seed-router:PES1UG20CS243:Mahika/
#>tc qdisc show dev eth0
qdisc noqueue 0: root refcnt 2
seed-router:PES1UG20CS243:Mahika/
#>
```

# 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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>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**

```
seed-attacker:PES1UG20CS243:Mahika/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        = 61177
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x3301
     src       = 10.9.0.53
     dst       = 199.43.135.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x58f0
###[ DNS ]###
           id        = 49946
           qr        = 0
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 512
            |  extrcode  = 0
            |  version   = 0
            |  z         = D0
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\x07\x98\xf1?\t\x8d\xd0\xe7'

.
Sent 1 packets.
```

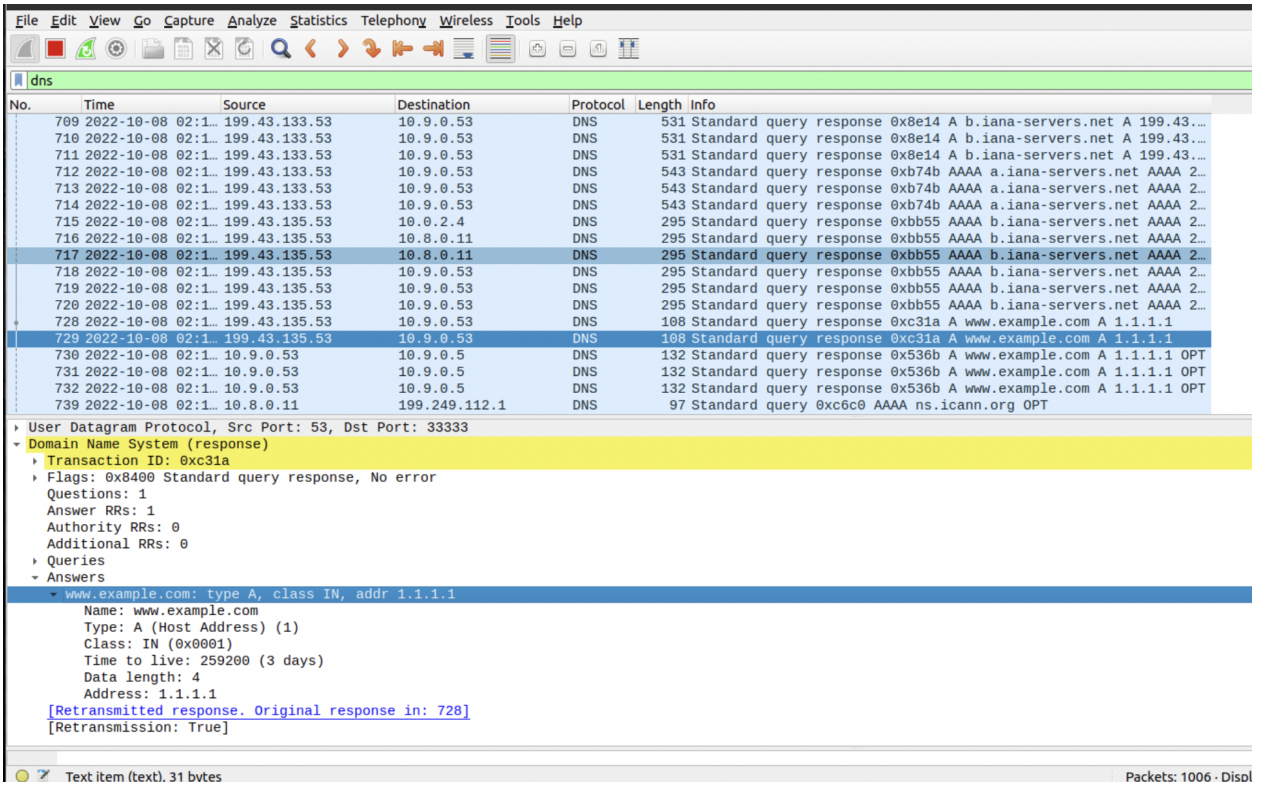
**This spoofed packet sent to the local DNS contains a spoofed IP address for example.com 1.1.1.1. The local DNS receives this query and stores this information in its cache.**

**On the victim terminal run the command**:

**# dig www.example.com**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21355
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 823f03c1d928ac7201000000634115d74ebb7c7212de0851 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 2408 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 08 06:16:55 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**As we can see in the answer section, the spoofed IP address for example.com 1.1.1.1 is displayed.**




**Query Response comes from attacker machine to local dns, and then the local dns sends the same response contents to the victim machine in packet capture 730.**

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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>rndc dumpdb -cache
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>cat /var/cache/bind/dump.db | grep example
example.com.            777125  NS      a.iana-servers.net.
www.example.com.        863526  A       1.1.1.1
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>
```

**Here we can see that the local dns cache stores the spoofed IP address. The local DNS receives the spoofed query from the attacker which contained this fake IP address and stores it in the cache.**

## 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.

**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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>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**

```
seed-attacker:PES1UG20CS243:Mahika/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        = 23895
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xc6a3
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 60370
           qr        = 0
```

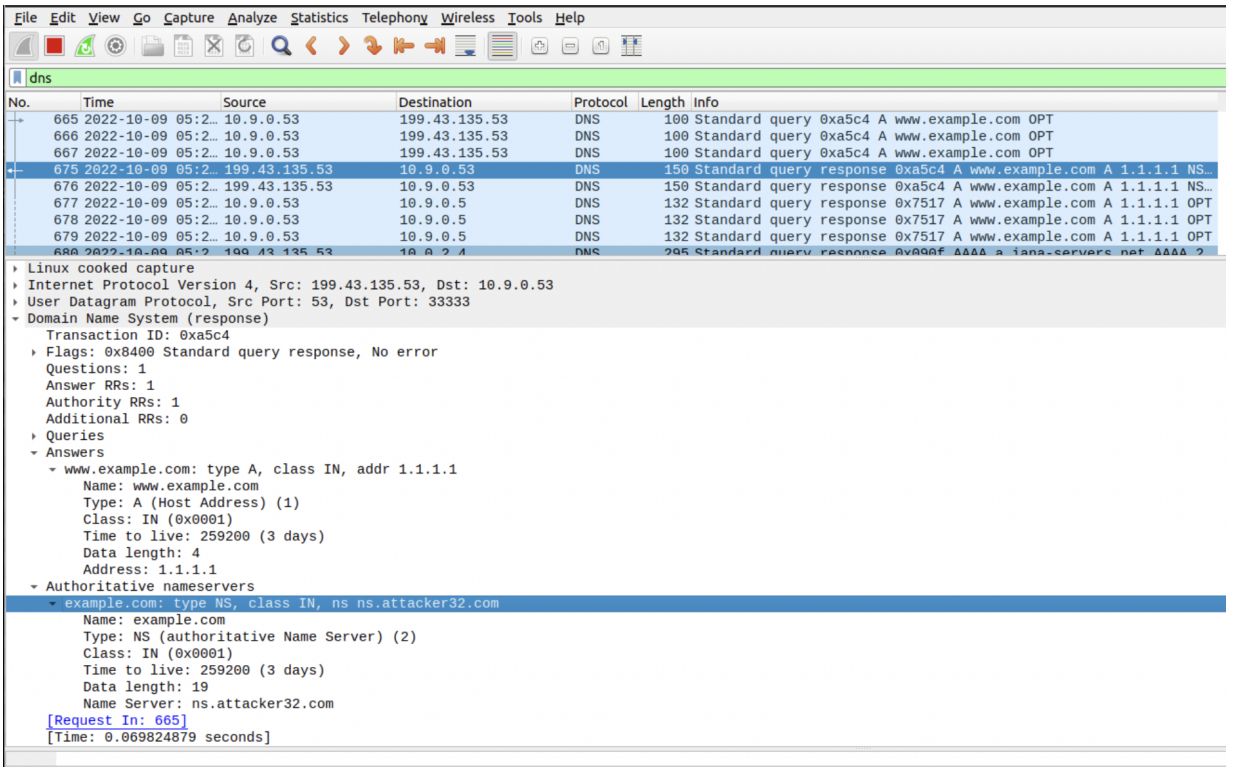
```
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 512
            |  extrcode  = 0
            |  version   = 0
            |  z         = D0
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '2c\xbf\x06\x13\xdc\xee*'

.
Sent 1 packets.
```

**On the victim terminal run the command**:

**# dig www.example.com**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14433
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: abd5436ee122713c01000000634118484dac9ad9df0ae054 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 2940 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Oct 08 06:27:20 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**We can see the Answer section containing spoofed IP address 1.1.1.1 for example.com**




**On wireshark capture of query response we can see that the authoritative section contains the attackers nameserver ns.attacker32.com instead of the actual nameserver of example.com.**

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**:

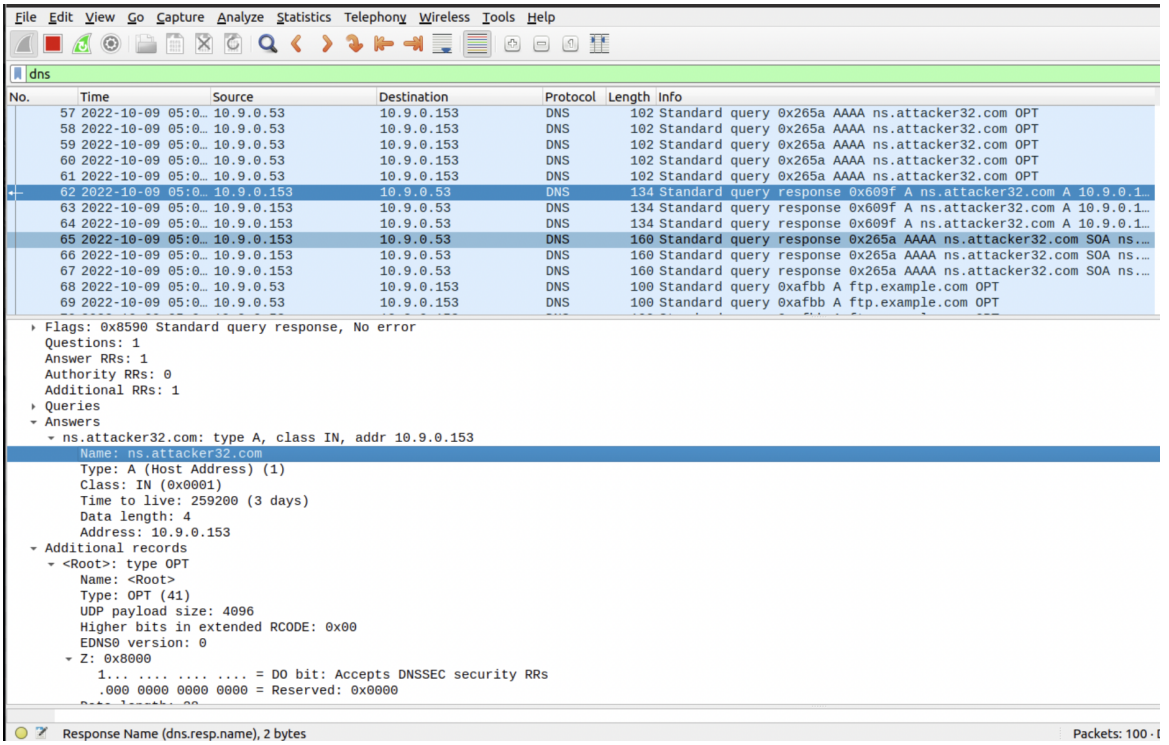
**# dig www.example.com**

**# dig ftp.example.com**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38129
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0184dd1f5c6a31e90100000063428dc7ab2ba5a1769c647b (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        258676  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 09 09:00:55 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28411
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8acdd111f693ea4c0100000063428de82a6d95a76e80d520 (good)
;; QUESTION SECTION:
;ftp.example.com.               IN      A

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

;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 09 09:01:28 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**We can see that ftp.example.com contains a spoofed IP address 1.2.3.6 in the answer section of the dns query.**




**We can see that the dns response packet for ftp.example.com contains the attackers nameserver in the Answer section as well as a spoofed ip address for ftp.example.com**

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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>rndc dumpdb -cache
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>cat /var/cache/bind/dump.db | grep example
example.com.            777319  NS      ns.attacker32.com.
ftp.example.com.        863968  A       1.2.3.6
www.example.com.        863722  A       1.1.1.1
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>
```

**We can see that the spoofed nameserver record of the attackers nameserver is saved on the cache for example.come as well as ftp.example.com**

# 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.

**Also fill in the appropriate interface name in the code for task 4 as done in previous tasks.**

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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>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**

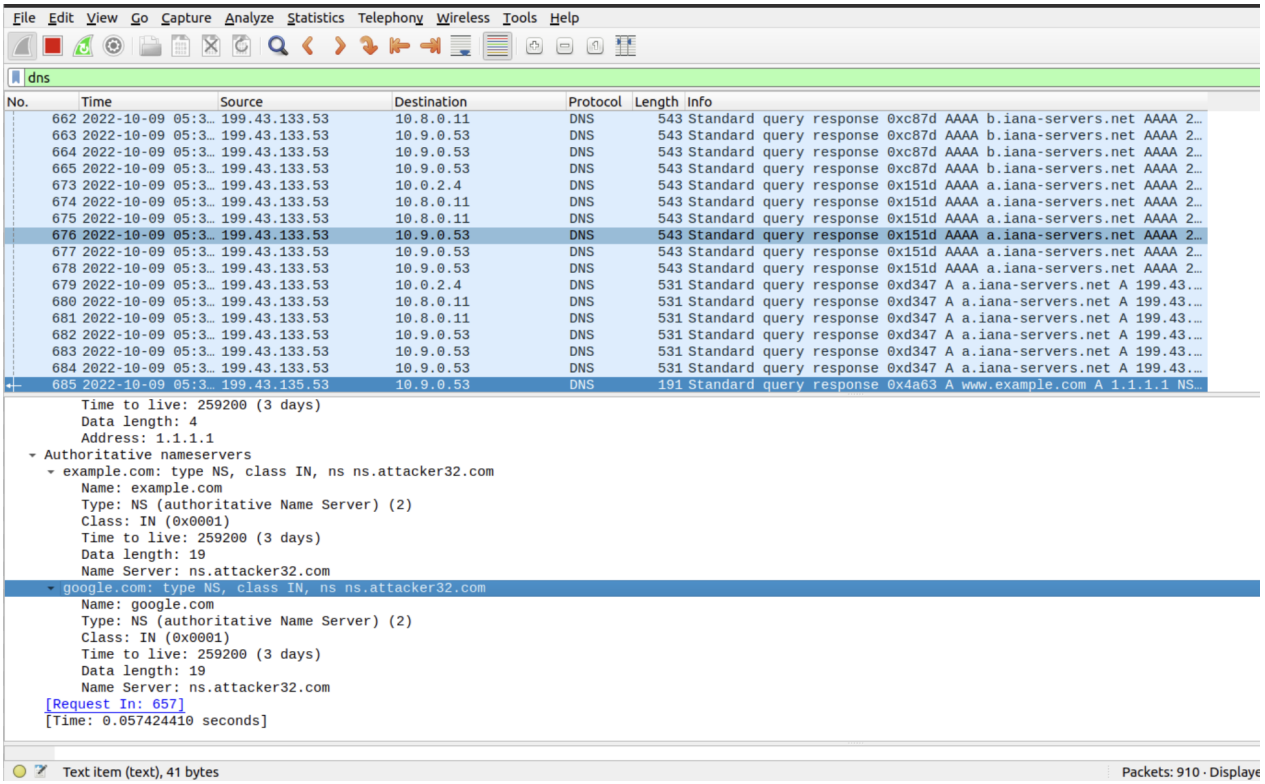
```
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 512
            |  extrcode  = 0
            |  version   = 0
            |  z         = D0
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\x07\x98\xf1?\t\x8d\xd0\xe7'

.
Sent 1 packets.
```

**On the victim terminal run the command**:

**# dig www.example.com**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55282
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 4f086581086bd7e80100000063429502ee3882ad29487583 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1839 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 09 09:31:46 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```

**Displays the spoofed IP address for example.com**




**We can see that the nameserver for google.com is also the attacker's name server ns.attacker32.com in the authoritative section of the dns query.**

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**

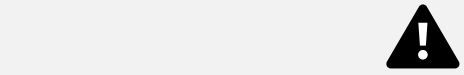
**# cat /var/cache/bind/dump.db | grep example**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>cat /var/cache/bind/dump.db | grep example
example.com.            777468  NS      ns.attacker32.com.
www.example.com.        863871  A       1.1.1.1
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>
```

**The spoofed Ns record for google.com is not stored in the local dns cache.It is only stored for example.com. This is because google.com is out of zone and hence it is not cached.**

## 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.



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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>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:**

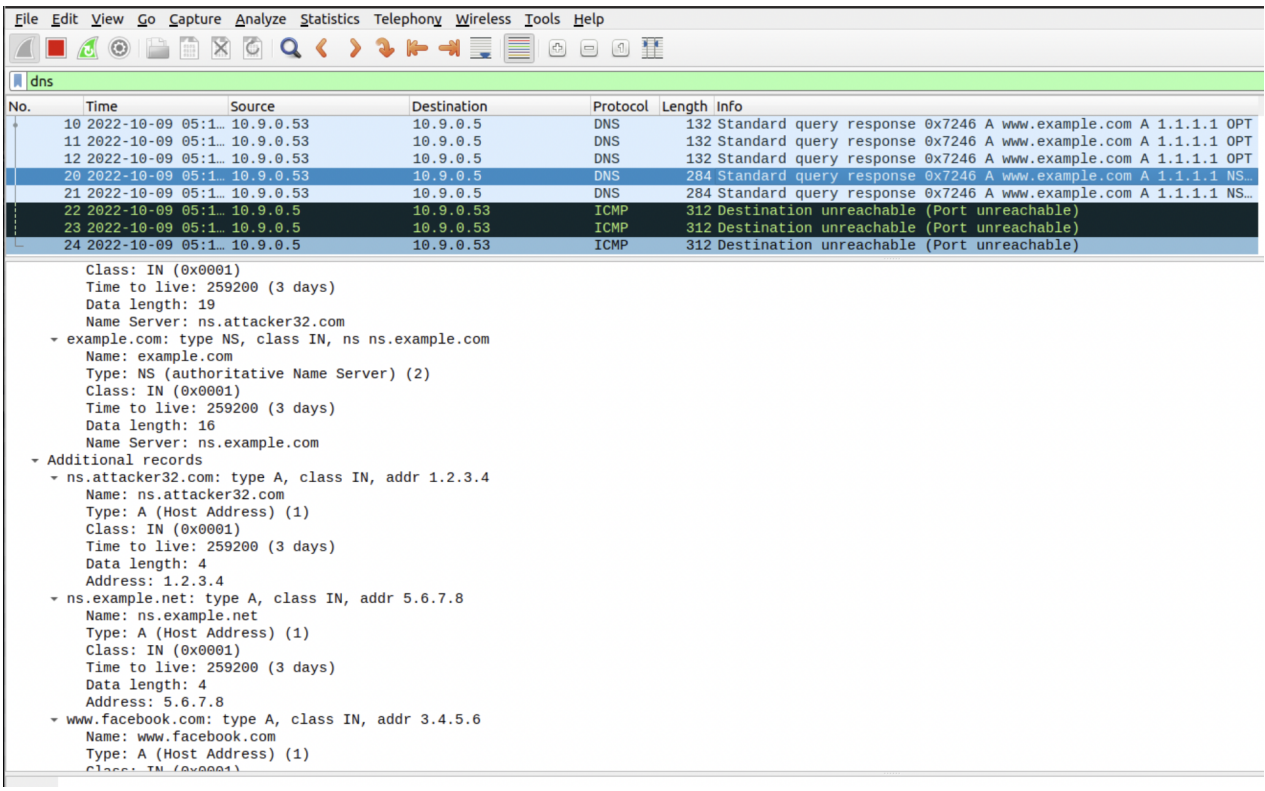
**# python3 task5.py**

```
seed-attacker:PES1UG20CS243:Mahika/volumes
#>python3 task5.py
.
Sent 1 packets.
```

**On the victim terminal run the command**:

**# dig www.example.com**

```
user-10.9.0.5:PES1UG20CS243:Mahika/
#>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29254
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e7ef718adf9e4f1101000000063429070afe16b5bd3b96eab (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259135  IN      A       1.1.1.1

;; Query time: 7 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Oct 09 09:12:16 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:PES1UG20CS243:Mahika/
#>
```




**Here we can see that in additional records, facebook.com is also mentioned with a spoofed IP 3.4.5.6, along with spoofed IPs of the nameserver of attacker and example.com.**

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**

```
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>rndc dumpdb -cache
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>cat /var/cache/bind/dump.db | grep example
example.com.            777396  NS      ns.attacker32.com.
www.example.com.        863799  A       1.1.1.1
local-dns-server-10.9.0.53:PES1UG20CS243:Mahika/
#>
```

**As we can see here, the spoofed IPs stored in the additional section of the DNS query are not cached in the local dns. It does not store in cache as it is considered to be out of zone.**

Describe your results and explain as to why you got the results you did.