

COMPUTER NETWORK SECURITY LABORATORY

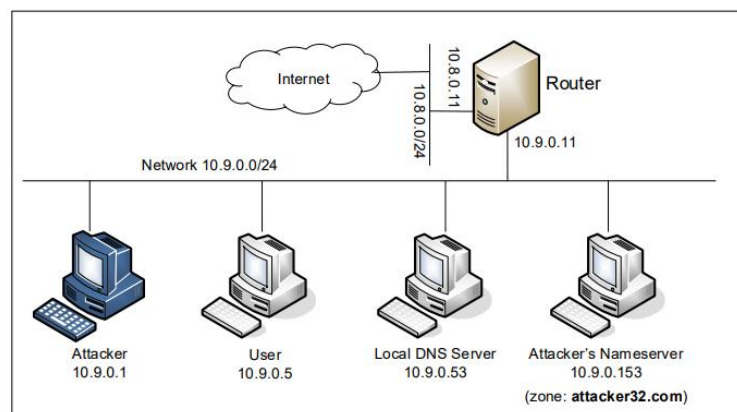
NAME : PREM SAGAR J S

SRN : PES1UG20CS825

SEC : H

Local DNS Attack Lab

Lab Environment Setup



Verification of the DNS setup

Getting the IP address of ns.attacker32.com

On the victim terminal run the command:

dig ns.attacker32.com

```
User:PES1UG20CS825:Prem Sagar J S/
#dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25395
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7a3aafbe2d87bd35010000006345a6e7938350593fab7df0 (good)
;; QUESTION SECTION:
;ns.attacker32.com.          IN      A

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

;; Query time: 44 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Oct 11 17:24:55 UTC 2022
;; MSG SIZE  rcvd: 90
```

Getting the IP address of www.example.com

On the victim terminal run the commands:

dig www.example.com

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62584
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 53256d674614773f010000006345a736b3ef71e657bleac8 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

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

;; Query time: 2019 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Oct 11 17:26:14 UTC 2022
;; MSG SIZE  rcvd: 88
```

dig @ns.attacker32.com www.example.com

```
User:PES1UG20CS825:Prem Sagar J S/
#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: 46963
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f701f59cc783736a010000006345a76da8cdee974715b493 (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: Tue Oct 11 17:27:00 UTC 2022
;; MSG SIZE  rcvd: 88
```

Attacks on DNS

Task 1: Directly Spoofing Response to User

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.

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

```
# rndc flush
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#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
```

```
User:PES1UG20CS825:Prem Sagar J S/  
#dig www.example.com  
  
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62664  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 4096  
; COOKIE: fd02911928a6b929010000006345a8a873de5d4b305c6823 (good)  
;; QUESTION SECTION:  
;www.example.com.                IN      A  
  
;; ANSWER SECTION:  
www.example.com.                86400   IN      A      93.184.216.34  
  
;; Query time: 500 msec  
;; SERVER: 10.9.0.53#53(10.9.0.53)  
;; WHEN: Tue Oct 11 17:32:24 UTC 2022  
;; MSG SIZE  rcvd: 88
```

Wireshark Screenshot of capturing dns packets:

The screenshot shows the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The packet list pane displays a list of captured packets, with the following columns: No., Time, Source, Destination, Protocol, Length, and Info. The selected packet is a DNS query from 10.8.0.11 to 10.8.0.11, with a length of 95 bytes. The packet details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, and User Datagram Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
76	2022-10-11 13:3...	10.8.0.11	198.41.0.4	DNS	95	Standard query 0x4a2d A ns.icann.org OPT
77	2022-10-11 13:3...	10.8.0.11	198.41.0.4	DNS	95	Standard query 0x4a18 AAAA ns.icann.org OPT
78	2022-10-11 13:3...	10.8.0.11	199.43.134.53	DNS	101	Standard query 0x4dd1 A a.iana-servers.net OPT
79	2022-10-11 13:3...	199.9.14.201	10.8.0.11	TCP	54	53 → 49707 [FIN, ACK] Seq=267393207 Ack=2779073744 Win=65535
80	2022-10-11 13:3...	10.8.0.11	199.9.14.201	TCP	54	49707 → 53 [ACK] Seq=2779073744 Ack=267393208 Win=63865 Len=0
81	2022-10-11 13:3...	198.97.190.53	10.8.0.11	DNS	1231	Standard query response 0x0826 AAAA a.iana-servers.net NS a.
82	2022-10-11 13:3...	10.8.0.11	198.97.190.53	TCP	54	57171 → 53 [ACK] Seq=500371367 Ack=267457179 Win=63558 Len=0
83	2022-10-11 13:3...	10.8.0.11	198.97.190.53	TCP	54	57171 → 53 [FIN, ACK] Seq=500371367 Ack=267457179 Win=63558
84	2022-10-11 13:3...	10.8.0.11	199.43.133.53	DNS	101	Standard query 0xa2ba AAAA a.iana-servers.net OPT
85	2022-10-11 13:3...	198.97.190.53	10.8.0.11	TCP	54	53 → 57171 [ACK] Seq=267457179 Ack=500371368 Win=65535 Len=0
86	2022-10-11 13:3...	198.41.0.4	10.8.0.11	DNS	285	Standard query response 0x4a2d A ns.icann.org NS d0.org.afil
87	2022-10-11 13:3...	10.8.0.11	198.41.0.4	TCP	74	58873 → 53 [SYN] Seq=1507842314 Win=64240 Len=0 MSS=1460 SACK
88	2022-10-11 13:3...	199.43.134.53	10.8.0.11	DNS	529	Standard query response 0x4dd1 A a.iana-servers.net A 199.43
89	2022-10-11 13:3...	10.8.0.11	199.43.135.53	DNS	98	Standard query 0x70e4 A www.example.com OPT
90	2022-10-11 13:3...	10.8.0.11	202.12.27.33	DNS	101	Standard query 0xbbaa A b.iana-servers.net OPT
91	2022-10-11 13:3...	10.8.0.11	202.12.27.33	DNS	101	Standard query 0xe7a8 AAAA b.iana-servers.net OPT
92	2022-10-11 13:3...	198.97.190.53	10.8.0.11	TCP	54	53 → 57171 [FIN, ACK] Seq=267457179 Ack=500371368 Win=65535
93	2022-10-11 13:3...	10.8.0.11	198.97.190.53	TCP	54	57171 → 53 [ACK] Seq=500371368 Ack=267457180 Win=63558 Len=0

Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface br-240455fb39ec, id 0
Ethernet II, Src: 02:42:0a:08:00:0b (02:42:0a:08:00:0b), Dst: 02:42:83:1a:0f:21 (02:42:83:1a:0f:21)
Internet Protocol Version 4, Src: 10.8.0.11, Dst: 192.58.128.30
User Datagram Protocol, Src Port: 33333, Dst Port: 53

0000 02 42 83 1a 0f 21 02 42 0a 08 00 0b 08 00 45 00 .B...!.B.....E.
0010 00 44 18 8f 00 00 3f 11 18 af 0a 08 00 0b c0 3a .D....?.....:

br-240455fb39ec: <live capture in progress> Packets: 119 · Displayed: 119 (100.0%) Profile: Default

Before launching the attack, making sure that the cache in the local DNS server is cleaned.

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

```
# rndc flush
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#rndc flush
```

On the attacker terminal run the command:

```
# python3 task1.py
```

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code  
#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       = 30452  
  flags    =  
  frag     = 0  
  ttl      = 64  
  proto    = udp  
  checksum = 0xef59  
  src      = 10.9.0.5  
  dst      = 10.9.0.53  
  \options \  
####[ UDP ]####  
  .
```

On the victim terminal run the command:

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

```
User:PES1UG20CS825:Prem Sagar J S/  
#dig www.example.com  
  
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62425  
;; 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: 84 msec  
;; SERVER: 10.9.0.53#53(10.9.0.53)  
;; WHEN: Tue Oct 11 17:57:43 UTC 2022  
;; MSG SIZE rcvd: 64  
  
User:PES1UG20CS825:Prem Sagar J S/  
..
```

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.

Wireshark Screenshot :

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-10-11 13:5...	10.8.0.11	199.9.14.201	DNS	82	Standard query 0x4a98 NS <Root> OPT
2	2022-10-11 13:5...	10.8.0.11	199.9.14.201	DNS	88	Standard query 0x7309 A __.com OPT
3	2022-10-11 13:5...	199.9.14.201	10.8.0.11	DNS	98	Standard query response 0x4a98 RRset exists NS <Root> OPT
4	2022-10-11 13:5...	10.8.0.11	199.9.14.201	DNS	98	Standard query 0x397f NS <Root> OPT
5	2022-10-11 13:5...	199.9.14.201	10.8.0.11	DNS	501	Standard query response 0x397f NS <Root> NS a.root-servers.r
6	2022-10-11 13:5...	10.8.0.11	199.9.14.201	TCP	74	34365 → 53 [SYN] Seq=1161670184 Win=64240 Len=0 MSS=1460 SA
7	2022-10-11 13:5...	199.9.14.201	10.8.0.11	TCP	58	53 → 34365 [SYN, ACK] Seq=454400001 Ack=1161670185 Win=65535
8	2022-10-11 13:5...	10.8.0.11	199.9.14.201	TCP	54	34365 → 53 [ACK] Seq=1161670185 Ack=454400002 Win=64240 Len=
9	2022-10-11 13:5...	10.8.0.11	199.9.14.201	DNS	112	Standard query 0xa15c NS <Root> OPT
10	2022-10-11 13:5...	199.9.14.201	10.8.0.11	TCP	54	53 → 34365 [ACK] Seq=454400002 Ack=1161670243 Win=65535 Len=
11	2022-10-11 13:5...	10.8.0.11	192.33.4.12	DNS	88	Standard query 0x2612 A __.com OPT
12	2022-10-11 13:5...	192.33.4.12	10.8.0.11	DNS	104	Standard query response 0x2612 A __.com OPT
13	2022-10-11 13:5...	10.8.0.11	192.33.4.12	TCP	74	39335 → 53 [SYN] Seq=1926742807 Win=64240 Len=0 MSS=1460 SA
14	2022-10-11 13:5...	199.9.14.201	10.8.0.11	DNS	1373	Standard query response 0xa15c NS <Root> NS a.root-servers.r
15	2022-10-11 13:5...	10.8.0.11	199.9.14.201	TCP	54	34365 → 53 [ACK] Seq=1161670243 Ack=454401321 Win=63312 Len=
16	2022-10-11 13:5...	10.8.0.11	199.9.14.201	TCP	54	34365 → 53 [FIN, ACK] Seq=1161670243 Ack=454401321 Win=63312
17	2022-10-11 13:5...	199.9.14.201	10.8.0.11	TCP	54	53 → 34365 [ACK] Seq=454401321 Ack=1161670244 Win=65535 Len=
18	2022-10-11 13:5...	192.33.4.12	10.8.0.11	TCP	58	53 → 39335 [SYN, ACK] Seq=454528001 Ack=1926742808 Win=65535

Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface br-240455fb39ec, id 0
 Ethernet II, Src: 02:42:0a:08:00:0b (02:42:0a:08:00:0b), Dst: 02:42:83:1a:0f:21 (02:42:83:1a:0f:21)
 Internet Protocol Version 4, Src: 10.8.0.11, Dst: 199.9.14.201
 User Datagram Protocol, Src Port: 33333, Dst Port: 53

0000 02 42 83 1a 0f 21 02 42 0a 08 00 0b 08 00 45 00 ·B···!·B·····E·
 0010 00 44 7b f5 00 00 3f 11 1f cf 0a 08 00 0b c7 09 ·D{··?········

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:PES1UG20CS825:Prem Sagar J S/
#rndc dumpdb -cache
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#cat /var/cache/bind/dump.db | grep example
example.com.          691005  NS      a.iana-servers.net.
                      20221022214625 20221001223409 1686 example.com.
www.example.com.      691005  A       93.184.216.34
                      20221022040544 20220930163209 1686 example.com.
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#
```

A potential issue

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.

```
Router:PES1UG20CS825:Prem Sagar J S/
#tc qdisc add dev eth0 root netem delay 100ms
Router:PES1UG20CS825:Prem Sagar J S/
#tc qdisc del dev eth0 root netem
Router:PES1UG20CS825:Prem Sagar J S/
#tc qdisc show dev eth0
qdisc noqueue 0: root refcnt 2
Router:PES1UG20CS825:Prem Sagar J S/
#
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

A 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.

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

```
# rndc flush
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#rndc flush
```

On the attacker terminal run the command:

```
# python3 task2.py
```

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#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       = 8997
  flags    =
  frag     = 0
  ttl      = 64
  proto    = udp
  chksum   = 0xd6
  src      = 10.9.0.53
  dst      = 199.43.133.53
  \options \
####[ UDP ]####
  --+--  +-----+
  | 0000 | 00000000 |
  +-----+-----+
```

The victim machine sends out a DNS query to the local DNS server using the dig command.

On the victim terminal run the command:

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

```
User:PES1UG20CS825:Prem Sagar J S/  
#dig www.example.com
```

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30961  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 4096  
; COOKIE: 41cb9863b3af90730100000006345b297411c86eee23977fc (good)  
;; QUESTION SECTION:  
;www.example.com.                IN      A  
  
;; ANSWER SECTION:  
www.example.com.                259200  IN      A      1.1.1.1  
  
;; Query time: 2918 msec  
;; SERVER: 10.9.0.53#53(10.9.0.53)  
;; WHEN: Tue Oct 11 18:14:47 UTC 2022  
;; MSG SIZE  rcvd: 88
```

the spoofed packet captured on wireshark and the cache of the local DNS server

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

```
# rndc dumpdb -cache
```

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

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#rndc dumpdb -cache  
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#cat /var/cache/bind/dump.db | grep example  
example.com.                777575  NS      a.iana-servers.net.  
www.example.com.            863976  A       1.1.1.1  
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#
```

Wireshark screenshot :

The screenshot shows a Wireshark capture of a network packet. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
51	2022-10-11 14:1...	10.8.0.11	202.12.27.33	TCP	54	53611 → 53 [FIN, ACK] Seq=4094198282 Ack=582081185 Win=63882
52	2022-10-11 14:1...	202.12.27.33	10.8.0.11	TCP	54	53 → 53611 [ACK] Seq=582081185 Ack=4094198283 Win=65535 Len=
53	2022-10-11 14:1...	10.8.0.11	192.48.79.30	DNS	101	Standard query 0x30cc A b.iana-servers.net OPT
54	2022-10-11 14:1...	202.12.27.33	10.8.0.11	TCP	54	53 → 53611 [FIN, ACK] Seq=582081185 Ack=4094198283 Win=65535
55	2022-10-11 14:1...	10.8.0.11	202.12.27.33	TCP	54	53611 → 53 [ACK] Seq=4094198283 Ack=582081186 Win=63882 Len=
56	2022-10-11 14:1...	192.48.79.30	10.8.0.11	DNS	536	Standard query response 0x30cc A b.iana-servers.net NS ns.ic
57	2022-10-11 14:1...	10.8.0.11	192.5.5.241	DNS	95	Standard query 0x7d4a A ns.icann.org OPT
58	2022-10-11 14:1...	10.8.0.11	192.5.5.241	DNS	95	Standard query 0x76f7 AAAA ns.icann.org OPT
59	2022-10-11 14:1...	10.8.0.11	199.43.134.53	DNS	101	Standard query 0x6718 A b.iana-servers.net OPT
60	2022-10-11 14:1...	192.5.5.241	10.8.0.11	DNS	418	Standard query response 0x7d4a A ns.icann.org DS RRSIG OPT
61	2022-10-11 14:1...	10.8.0.11	192.5.5.241	DNS	101	Standard query 0x4540 AAAA b.iana-servers.net OPT
62	2022-10-11 14:1...	10.8.0.11	192.5.5.241	DNS	101	Standard query 0x7a3c A a.iana-servers.net OPT
63	2022-10-11 14:1...	10.8.0.11	192.5.5.241	DNS	101	Standard query 0x46cb AAAA a.iana-servers.net OPT
64	2022-10-11 14:1...	10.8.0.11	192.5.5.241	TCP	74	52921 → 53 [SYN] Seq=290650863 Win=64240 Len=0 MSS=1460 SACK
65	2022-10-11 14:1...	192.5.5.241	10.8.0.11	TCP	58	53 → 52921 [SYN, ACK] Seq=582208001 Ack=290650864 Win=65535
66	2022-10-11 14:1...	10.8.0.11	192.5.5.241	TCP	54	52921 → 53 [ACK] Seq=290650864 Ack=582208002 Win=64240 Len=0
67	2022-10-11 14:1...	10.8.0.11	192.5.5.241	DNS	109	Standard query 0x87ef A ns.icann.org OPT
68	2022-10-11 14:1...	192.5.5.241	10.8.0.11	TCP	54	53 → 52921 [ACK] Seq=582208002 Ack=290650819 Win=65535 Len=0

Frame 1: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface br-240455fb39ec, id 0
Ethernet II, Src: 02:42:0a:08:00:0b (02:42:0a:08:00:0b), Dst: 02:42:83:1a:0f:21 (02:42:83:1a:0f:21)
Internet Protocol Version 4, Src: 10.8.0.11, Dst: 192.203.230.10
User Datagram Protocol, Src Port: 33333, Dst Port: 53

0000 02 42 83 1a 0f 21 02 42 0a 08 00 0b 08 00 45 00 .B...!B.....E.
0010 00 4a d2 c6 00 00 3f 11 f7 f3 0a 08 00 0b c0 cb .J....?.....

Task 3: Spoofing NS Records

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.
```

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

```
# rndc flush
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#rndc flush
```

On the attacker terminal run the command:

```
# python3 task3.py
```

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#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       = 40068
  flags    =
  frag     = 0
  ttl      = 64
  proto    = udp
  chksum   = 0x8576
  src      = 10.9.0.53
  dst      = 199.43.135.53
  \options \
###[ UDP ]###
  .....
```

On the victim terminal run the command:

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

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32197
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 92c71113514cf543010000006345b444c388985d7ecd07b4 (good)
```

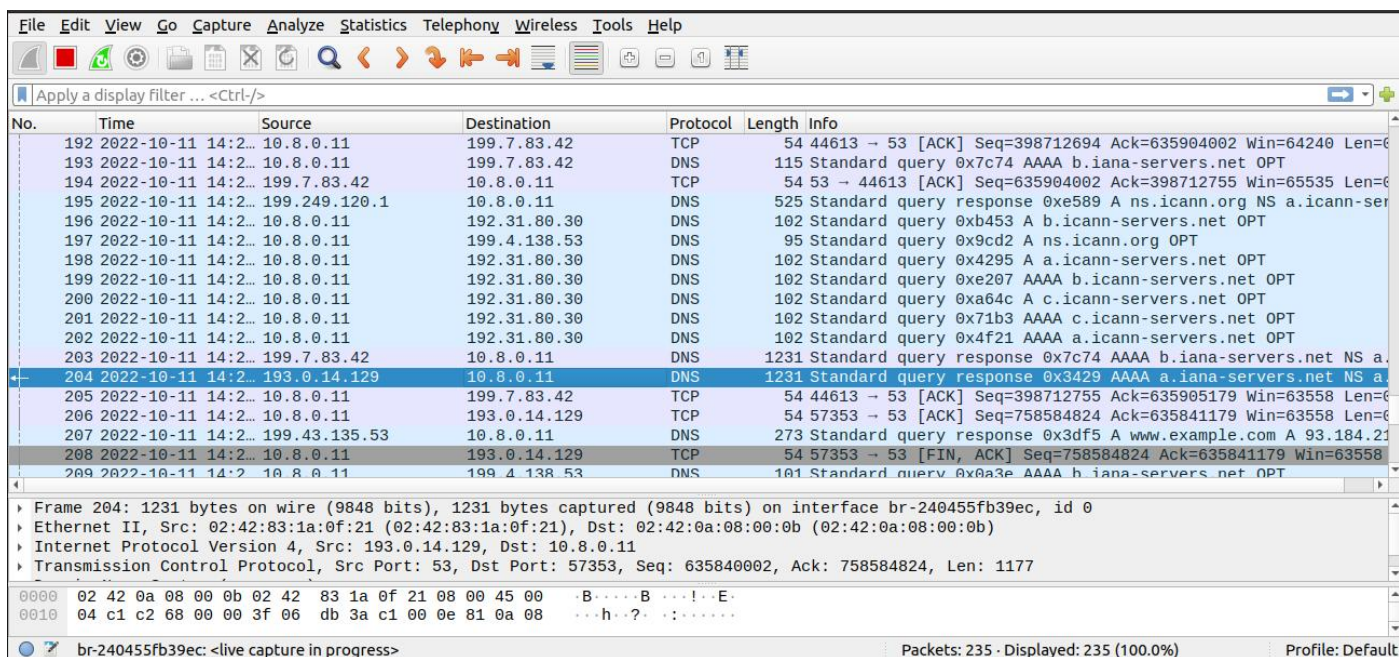


```
;; QUESTION SECTION:
www.example.com.                IN      A

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

;; Query time: 3999 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Oct 11 18:21:56 UTC 2022
;;
```

Wireshark Screenshot :



If my attack is successful, when i 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
```

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 27285
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e313b3a2df8751ea010000006345b54e93e210a161ec24e0 (good)
;; QUESTION SECTION:
www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.                258934 IN      A      1.1.1.1

;; Query time: 3 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Oct 11 18:26:22 UTC 2022
;;
```

```
# dig ftp.example.com
```

```
User:PES1UG20CS825:Prem Sagar J S/
#dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9785
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c44d5d3815016d2c010000006345b57a65c604fcf94734ac (good)
;; QUESTION SECTION:
;ftp.example.com.                IN      A

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

;; Query time: 31 msec
;; SERVER: 10.0.0.53#53(10.0.0.53)
```

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

```
# rndc dumpdb -cache
```

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

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#rndc dumpdb -cache
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#cat /var/cache/bind/dump.db | grep example
example.com.                777211  NS      ns.attacker32.com.
ftp.example.com.            863923  A       1.2.3.6
www.example.com.            863613  A       1.1.1.1
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#
```

Task 4: Spoofing NS Records for Another Domain

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
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#rndc flush
```

On the attacker terminal run the command:

```
# python3 task4.py
```

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#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       = 2636
  flags    =
  frag     = 0
  ttl      = 64
  proto    = udp
  chksum   = 0x19af
  src      = 10.9.0.53
  dst      = 199.43.133.53
  \options \
#### UDP ####
```

On the victim terminal run the command:

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

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61087
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ad5c2ad19b521167010000006345b7c3f8260d9d66cd354a (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

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

;; Query time: 3452 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Oct 11 18:36:51 UTC 2022
;; MSG SIZE  rcv=
```

the spoofed packet captured on wireshark and the cache of the local DNS server

Wireshark screenshot :

No.	Time	Source	Destination	Protocol	Length	Info
337	2022-10-11 14:3...	192.58.128.30	10.8.0.11	TCP	54	53 → 41219 [ACK] Seq=747520783 Ack=1211295755 Win=65535 Len=
338	2022-10-11 14:3...	10.8.0.11	199.249.120.1	DNS	95	Standard query 0x6ee1 A ns.icann.org OPT
339	2022-10-11 14:3...	10.8.0.11	199.19.56.1	DNS	95	Standard query 0xc76d AAAA ns.icann.org OPT
340	2022-10-11 14:3...	10.8.0.11	192.58.128.30	TCP	54	52365 → 53 [FIN, ACK] Seq=775513487 Ack=747584783 Win=63960
341	2022-10-11 14:3...	10.8.0.11	199.43.133.53	DNS	98	Standard query 0x3f39 A www.example.com OPT
342	2022-10-11 14:3...	192.58.128.30	10.8.0.11	TCP	54	53 → 52365 [ACK] Seq=747584783 Ack=775513488 Win=65535 Len=6
343	2022-10-11 14:3...	199.19.56.1	10.8.0.11	DNS	513	Standard query response 0xc76d AAAA ns.icann.org NS c.icann-
344	2022-10-11 14:3...	10.8.0.11	192.5.6.30	DNS	102	Standard query 0x85fa A b.icann-servers.net OPT
345	2022-10-11 14:3...	10.8.0.11	199.4.138.53	DNS	95	Standard query 0xba57 AAAA ns.icann.org OPT
346	2022-10-11 14:3...	10.8.0.11	192.5.6.30	DNS	102	Standard query 0xd4a4 AAAA b.icann-servers.net OPT
347	2022-10-11 14:3...	10.8.0.11	192.5.6.30	DNS	102	Standard querv 0x8821 A c.icann-servers.net OPT

349	2022-10-11 14:3...	10.8.0.11	192.5.6.30	DNS	102 Standard query 0xf588 A a.icann-servers.net OPT
350	2022-10-11 14:3...	10.8.0.11	192.5.6.30	DNS	102 Standard query 0xf305 AAAA a.icann-servers.net OPT
351	2022-10-11 14:3...	10.8.0.11	192.58.128.30	DNS	101 Standard query 0x4816 A a.iana-servers.net OPT
352	2022-10-11 14:3...	192.5.6.30	10.8.0.11	DNS	537 Standard query response 0x85fa A b.icann-servers.net NS ns.j
353	2022-10-11 14:3...	192.58.128.30	10.8.0.11	DNS	310 Standard query response 0x4816 A a.iana-servers.net NS a.gt]
354	2022-10-11 14:3...	10.8.0.11	192.58.128.30	TCP	74 37789 → 53 [SYN] Seq=474129566 Win=64240 Len=0 MSS=1460 SACK

Frame 204: 1231 bytes on wire (9848 bits), 1231 bytes captured (9848 bits) on interface br-240455fb39ec, id 0	
Ethernet II, Src: 02:42:83:1a:0f:21 (02:42:83:1a:0f:21), Dst: 02:42:0a:08:00:0b (02:42:0a:08:00:0b)	
Internet Protocol Version 4, Src: 193.0.14.129, Dst: 10.8.0.11	
Transmission Control Protocol, Src Port: 53, Dst Port: 57353, Seq: 635840002, Ack: 758584824, Len: 1177	

0000	02 42 0a 08 00 0b 02 42	83 1a 0f 21 08 00 45 00	.B....B...!...E.
0010	04 c1 c2 68 00 00 3f 06	db 3a c1 00 0e 81 0a 08	...h..?..:.....

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

```
# rndc dumpdb -cache
# cat /var/cache/bind/dump.db | grep example
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#rndc dumpdb -cache
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#cat /var/cache/bind/dump.db | grep example
example.com.          777446 NS      ns.attacker32.com.
www.example.com.      863848 A       1.1.1.1
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#
```

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
```

clearing the cache on the local DNS server first

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

```
# rndc flush
```

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/
#rndc flush
```

On the attacker terminal run the command:

```
# python3 task5.py
```

```
Attacker:PES1UG20CS825:Prem Sagar J S/volumes/Code
#python3 task5.py
.
Sent 1 packets.
.
Sent 1 packets.
█
```

The victim machine sends out a DNS query to the local DNS server using the dig command

On the victim terminal run the command:

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

```
User:PES1UG20CS825:Prem Sagar J S/
#dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1021
;; 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
```

Wireshark screenshot :

The Wireshark screenshot displays a network traffic capture. The packet list pane shows a series of DNS queries and responses. The packet details pane shows the structure of a DNS packet, including the Ethernet II header, Internet Protocol Version 4 header, and Transmission Control Protocol header.

No.	Time	Source	Destination	Protocol	Length	Info
557	2022-10-11 14:5...	192.33.4.12	10.8.0.11	TCP	54	53 → 37927 [ACK] Seq=859712816 Ack=2907843690 Win=65535 Len=...
558	2022-10-11 14:5...	10.8.0.11	199.19.53.1	DNS	95	Standard query 0x9e97 A ns.icann.org OPT
559	2022-10-11 14:5...	199.19.53.1	10.8.0.11	DNS	513	Standard query response 0x9e97 A ns.icann.org NS ns.icann.org
560	2022-10-11 14:5...	10.8.0.11	192.42.93.30	DNS	102	Standard query 0x7282 A b.icann-servers.net OPT
561	2022-10-11 14:5...	10.8.0.11	199.4.138.53	DNS	95	Standard query 0x370c A ns.icann.org OPT
562	2022-10-11 14:5...	10.8.0.11	192.42.93.30	DNS	102	Standard query 0x0610 A a.icann-servers.net OPT
563	2022-10-11 14:5...	10.8.0.11	192.42.93.30	DNS	102	Standard query 0x582f AAAA a.icann-servers.net OPT
564	2022-10-11 14:5...	10.8.0.11	192.42.93.30	DNS	102	Standard query 0x7fc2 AAAA b.icann-servers.net OPT
565	2022-10-11 14:5...	10.8.0.11	192.42.93.30	DNS	102	Standard query 0xf0a6 A c.icann-servers.net OPT
566	2022-10-11 14:5...	10.8.0.11	192.42.93.30	DNS	102	Standard query 0x1197 AAAA c.icann-servers.net OPT
567	2022-10-11 14:5...	192.33.4.12	10.8.0.11	TCP	54	53 → 37927 [FIN, ACK] Seq=859712816 Ack=2907843690 Win=65535 Len=...
568	2022-10-11 14:5...	10.8.0.11	192.33.4.12	TCP	54	37927 → 53 [ACK] Seq=2907843690 Ack=859712817 Win=63492 Len=...
569	2022-10-11 14:5...	192.42.93.30	10.8.0.11	DNS	537	Standard query response 0x7282 A b.icann-servers.net NS ns.i
570	2022-10-11 14:5...	10.8.0.11	199.43.135.53	DNS	102	Standard query 0x7041 A b.icann-servers.net OPT
571	2022-10-11 14:5...	10.8.0.11	192.5.5.241	DNS	95	Standard query 0x2e12 AAAA ns.icann.org OPT
572	2022-10-11 14:5...	199.4.138.53	10.8.0.11	DNS	516	Standard query response 0x370c A ns.icann.org A 199.4.138.53
573	2022-10-11 14:5...	192.5.5.241	10.8.0.11	DNS	418	Standard query response 0x2e12 AAAA ns.icann.org DS RRSIG OF
574	2022-10-11 14:5...	10.8.0.11	192.5.5.241	TCP	74	49175 → 53 [SYN] Seq=2682088529 Win=64240 Len=0 MSS=1460 SAF

Frame 204: 1231 bytes on wire (9848 bits), 1231 bytes captured (9848 bits) on interface br-240455fb39ec, id 0

- Ethernet II, Src: 02:42:83:1a:0f:21 (02:42:83:1a:0f:21), Dst: 02:42:0a:08:00:0b (02:42:0a:08:00:0b)
- Internet Protocol Version 4, Src: 193.0.14.129, Dst: 10.8.0.11
- Transmission Control Protocol, Src Port: 53, Dst Port: 57353, Seq: 635840002, Ack: 758584824, Len: 1177

0000 02 42 0a 08 00 0b 02 42 83 1a 0f 21 08 00 45 00 .B....B....E.

0010 04 c1 c2 68 00 00 3f 06 db 3a c1 00 0e 81 0a 08 ...h...?..:.....

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

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

```
# rndc dumpdb -cache
```

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

```
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#rndc dumpdb -cache  
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#cat /var/cache/bind/dump.db | grep example  
example.com.          777595  NS      ns.example.com.  
www.example.com.      863996  A       1.1.1.1  
Local-DNS:PES1UG20CS825:Prem Sagar J S/  
#
```