CS 5153/5053 Network Security, Spring 2023

Project 4: Local DNS Cache Poisoning

Report

Student: Austin Tyler Conn

Contents

Link to Source Code https://github.com/austinc3030/dns_m11809075

Host Environment Used

Operating System: Ubuntu 20.04 LTS

```
seed@network-security-seedlabs:/home/austinc3030$ uname -a
Linux network-security-seedlabs 5.15.0-1030-gcp #37~20.04.1-Ubuntu SMP Mon Feb 2
0 04:30:57 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
seed@network-security-seedlabs:/home/austinc3030$
```

Hardware: Google Cloud E2 Instance

Links Used for Environment Setup:

- seed-labs/seedvm-cloud.md at master · seed-labs/seed-labs (github.com)
- seed-labs/create_vm_gcp.md at master · seed-labs/seed-labs (github.com)

Docker Information

```
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ dockps
d40f8d8f8516  attacker-ns-10.9.0.153
9504fa5ca8fd  local-dns-server-10.9.0.53
098b8cff96fe  seed-attacker
ad7db9d346d5  seed-router
dfd9535c551a  user-10.9.0.5
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
99eac2f26d16   bridge         bridge    local
ba0612588179   host           host      local
c4f2a8af80e2   net-10.8.0.0   bridge    local
3761e470b177   net-10.9.0.0   bridge    local
bca514a37034   none           null      local
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$
```

Assumptions

a. User Machine (10.0.2.18) = user-10.9.0.5 (10.9.0.5)

```
user-10.9.0.5
File  Edit  View  Terminal  Tabs  Help
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ docksh user-10
.9.0.5
root@dfd9535c551a:/#
```

b. Attacker (10.0.2.17) = seed-attacker (10.9.0.1)

```
seed-attacker
File  Edit  View  Terminal  Tabs  Help
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ docksh seed-at
tacker
root@network-security-seedlabs:/#
```

c. Local DNS Server (10.0.2.16) = local-dns-server-10.9.0.53 (10.9.0.53)

```
local-dns-server-10.9.0.53
File  Edit  View  Terminal  Tabs  Help
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ docksh local-d
ns-server-10.9.0.53
root@9504fa5ca8fd:/#
```

How do you setup the User machine and Server machine?

User Machine (Task 1)

1. Look at the contents of */etc/resolv.conf*.

```
root@dfd9535c551a:/# cat /etc/resolv.conf
nameserver 10.9.0.53
root@dfd9535c551a:/#
```

2. It appears that the SEED Labs Docker Image is already configured to use the local-dns-server-10.9.0.53 as the DNS server. Note that *resolvconf* is not installed in this image. Verify using the *dig* command that 10.9.0.53 is the DNS server in use.

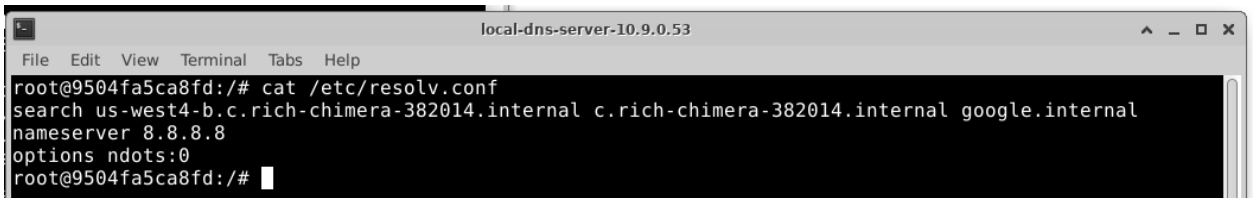
```
root@dfd9535c551a:/# dig www.google.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62180
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9f04a44fe715edbe0100000064276c39abde62de19890ec5 (good)
;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.         289     IN      A       142.250.101.103
www.google.com.         289     IN      A       142.250.101.106
www.google.com.         289     IN      A       142.250.101.105
www.google.com.         289     IN      A       142.250.101.99
www.google.com.         289     IN      A       142.250.101.147
www.google.com.         289     IN      A       142.250.101.104

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Mar 31 23:26:49 UTC 2023
;; MSG SIZE  rcvd: 167

root@dfd9535c551a:/#
```

Note: *as seen above, the server used by dig is 10.9.0.53, the IP address of local-dns-server-10.9.0.53*.

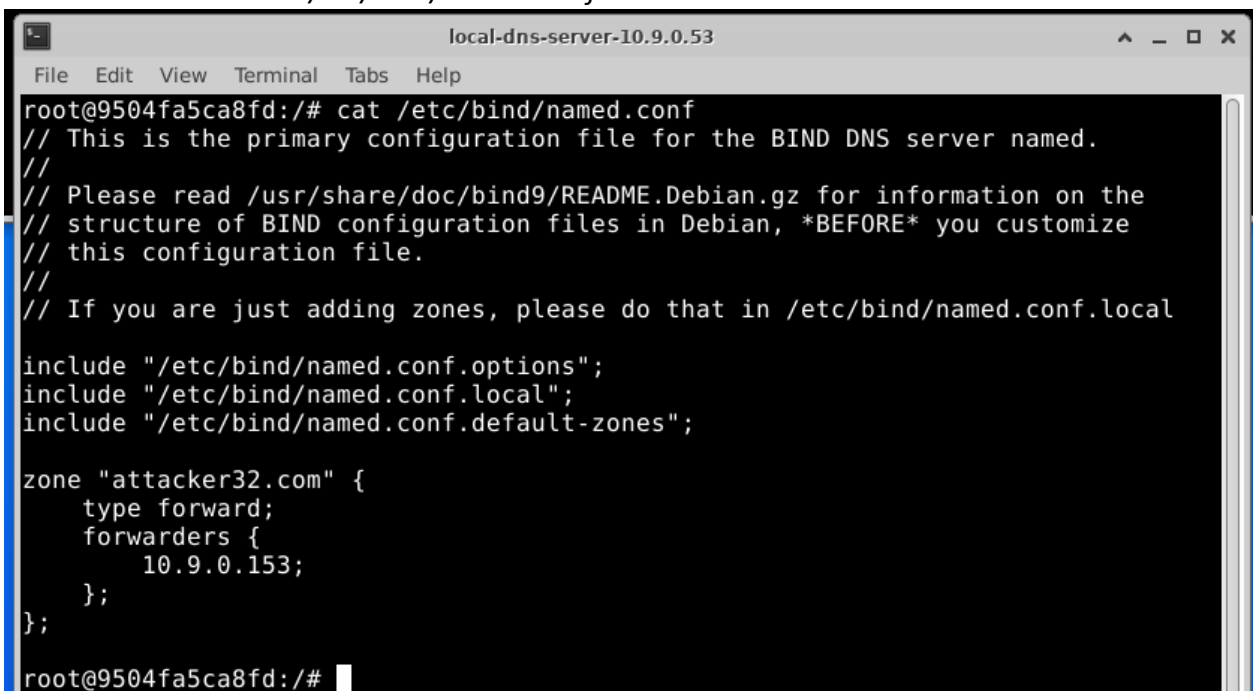
Server Machine (Task 2)

Disclaimer: *Due to using SEED Labs Docker Containers, the server DOES need to have its /etc/resolv.conf file updated to use a DNS server OTHER than docker's internal DNS server. Without doing this, the local-dns-server-10.9.0.53 will not reach out for queries, thus inhibiting the attacker being able to spoof the reply to local-dns-server-10.9.0.53.*

1. As mentioned in the disclaimer above, change the address in */etc/resolv.conf* to something other than docker's internal DNS server.



2. Look at the contents of */etc/bind/named.conf*.



3. Look at the contents of */etc/bind/named.conf.options*.

```
root@9504fa5ca8fd:/# cat /etc/bind/named.conf.options
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================

        // ---------------------------------------
        // Added/Modified for SEED labs
        // dnssec-validation auto;
        dnssec-validation no;
        dnssec-enable no;
        dump-file "/var/cache/bind/dump.db";
        query-source port         33333;

        // Access control
        allow-query { any; };
        allow-query-cache { any; };
        allow-recursion { any; };

        // ---------------------------------------

        listen-on-v6 { any; };
};
root@9504fa5ca8fd:/#
```

4. Dump the DNS cache to the file specified by the *dump-file* line in step 2 (*/var/cache/bind/dump.db*).

```
root@9504fa5ca8fd:/# rndc dumpdb -cache
root@9504fa5ca8fd:/# ~
```

5. Flush the DNS cache.

6. Turn off DNSSEC by modifying the */etc/bind/named.conf.options* file to comment out *dnssec-validation auto;* and add *dnssec-enable no;*.

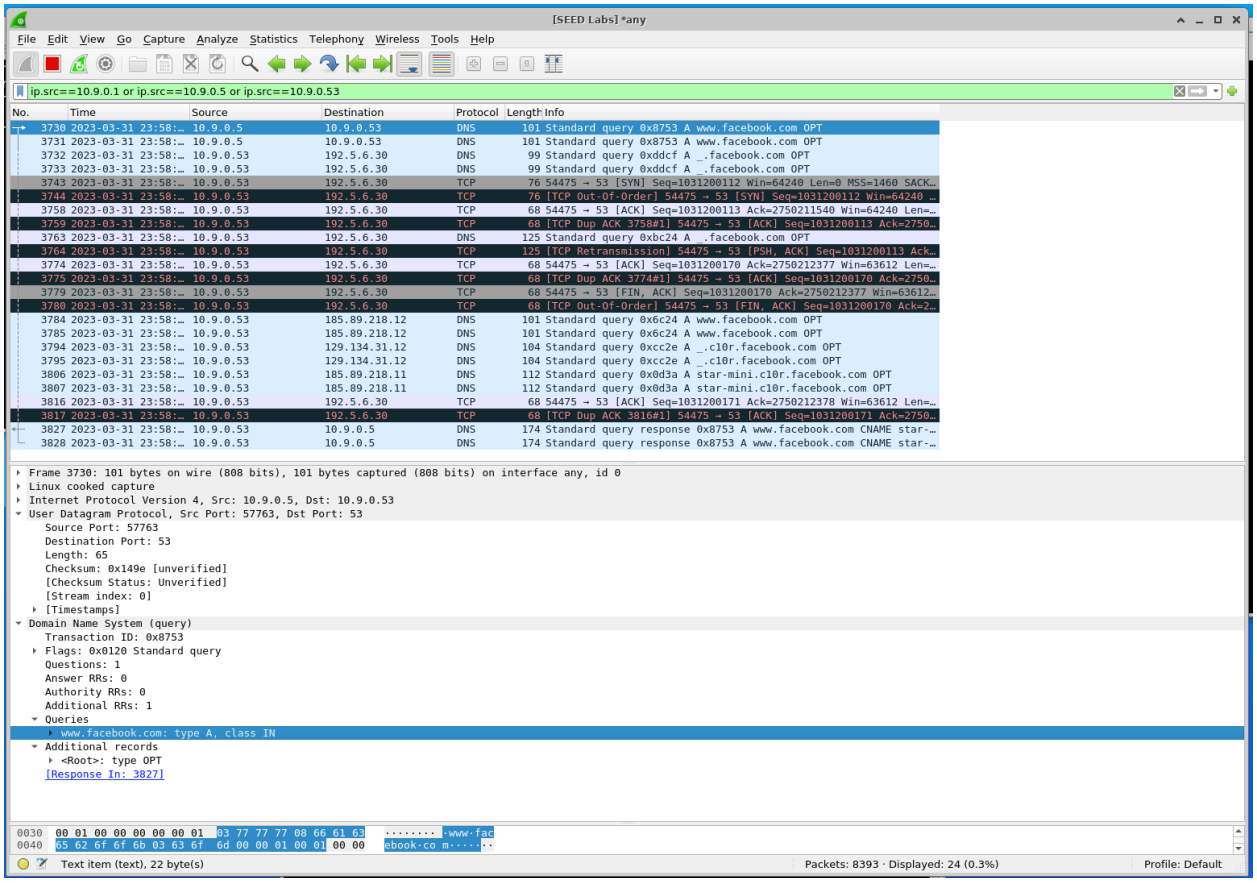
```
local-dns-server-10.9.0.53
File  Edit  View  Terminal  Tabs  Help
  GNU nano 4.8              /etc/bind/named.conf.options              Modified
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================>
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================>

        // ---------------------------------------
        // Added/Modified for SEED labs
        // dnssec-validation auto;
        dnssec-validation no;
        dnssec-enable no;
        dump-file "/var/cache/bind/dump.db";
        query-source port         33333;

        // Access control
        allow-query { any; };
        allow-query-cache { any; };
        allow-recursion { any; };

        // ---------------------------------------

        listen-on-v6 { any; };
};

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Note: *this appears to already be configured in the SEED Labs Docker Images provided.*

7. From the seed-user, test that DNS is working properly using *dig*.

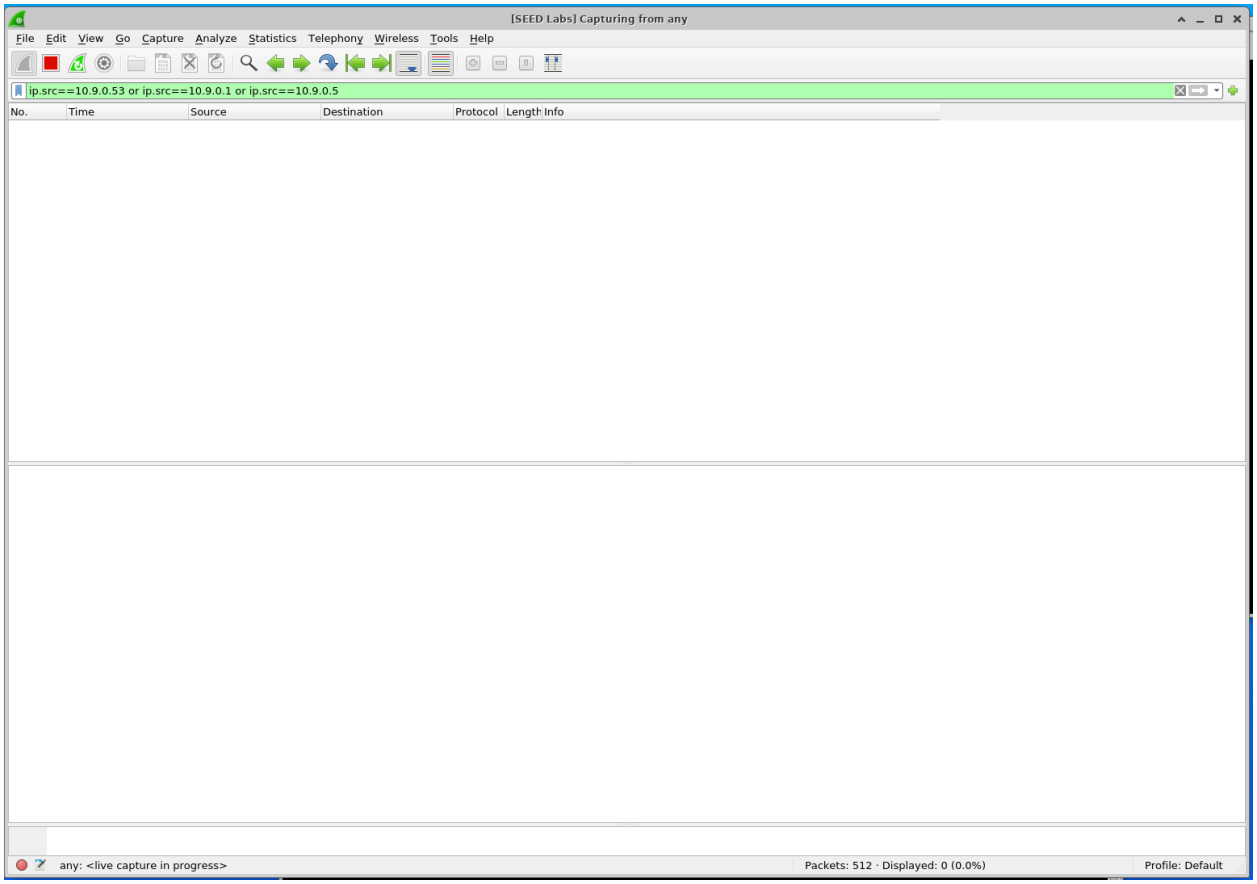
```
root@dfd9535c551a:/# dig www.google.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40854
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ec5bb6b06189994601000000064277293d7dc362a4b842336 (good)
;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.         300     IN      A       142.250.101.105
www.google.com.         300     IN      A       142.250.101.99
www.google.com.         300     IN      A       142.250.101.103
www.google.com.         300     IN      A       142.250.101.104
www.google.com.         300     IN      A       142.250.101.147
www.google.com.         300     IN      A       142.250.101.106

;; Query time: 76 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Mar 31 23:53:55 UTC 2023
;; MSG SIZE  rcvd: 167

root@dfd9535c551a:/#
```

8. Use Wireshark to verify a DNS query is made when running *dig* on seed-user.

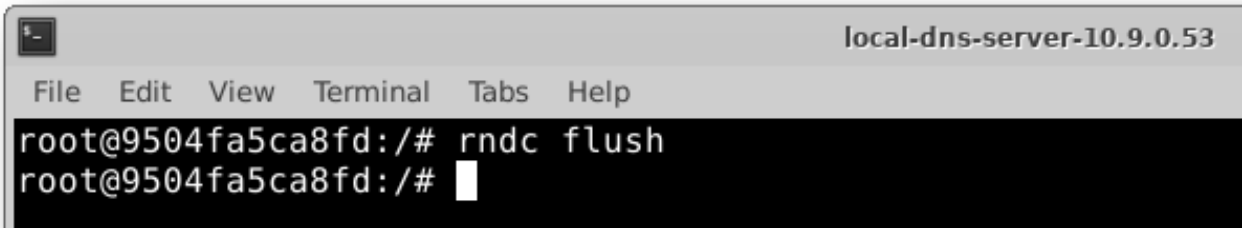


How do you perform the attack in your VM?

1. Start Wireshark monitoring traffic between seed-attacker, seed-user, and local-dns-server-10.9.0.53.



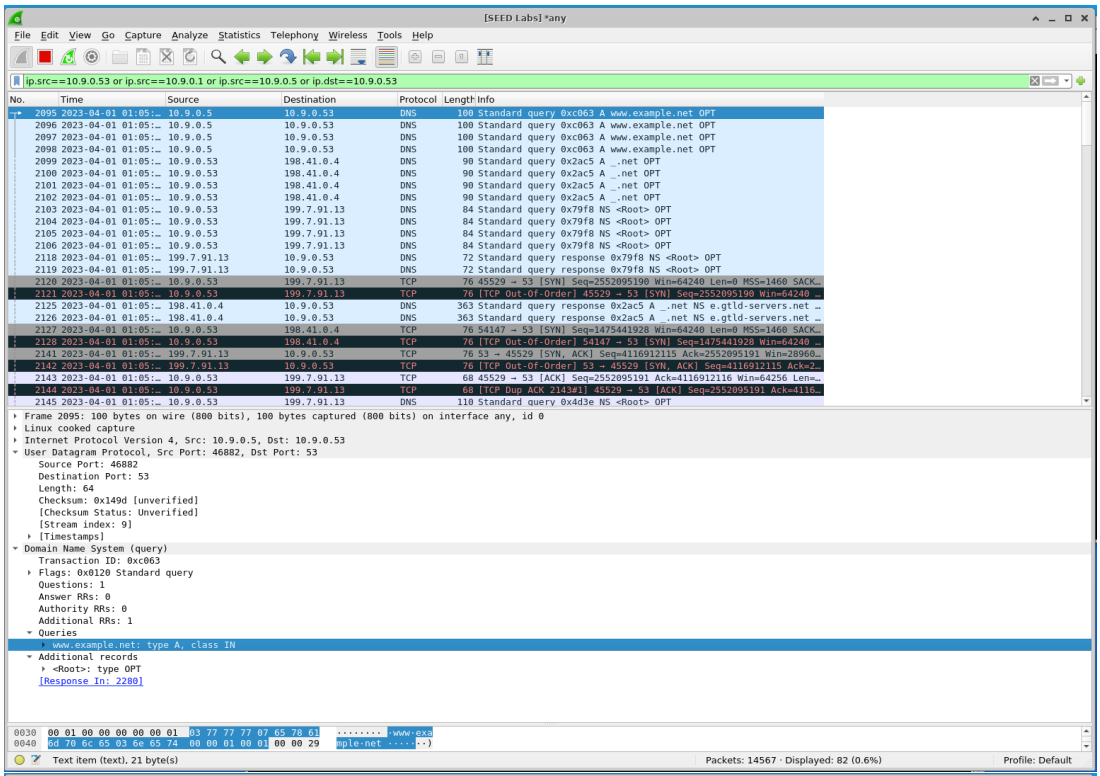
2. Flush DNS cache on local-dns-server-10.9.0.53.

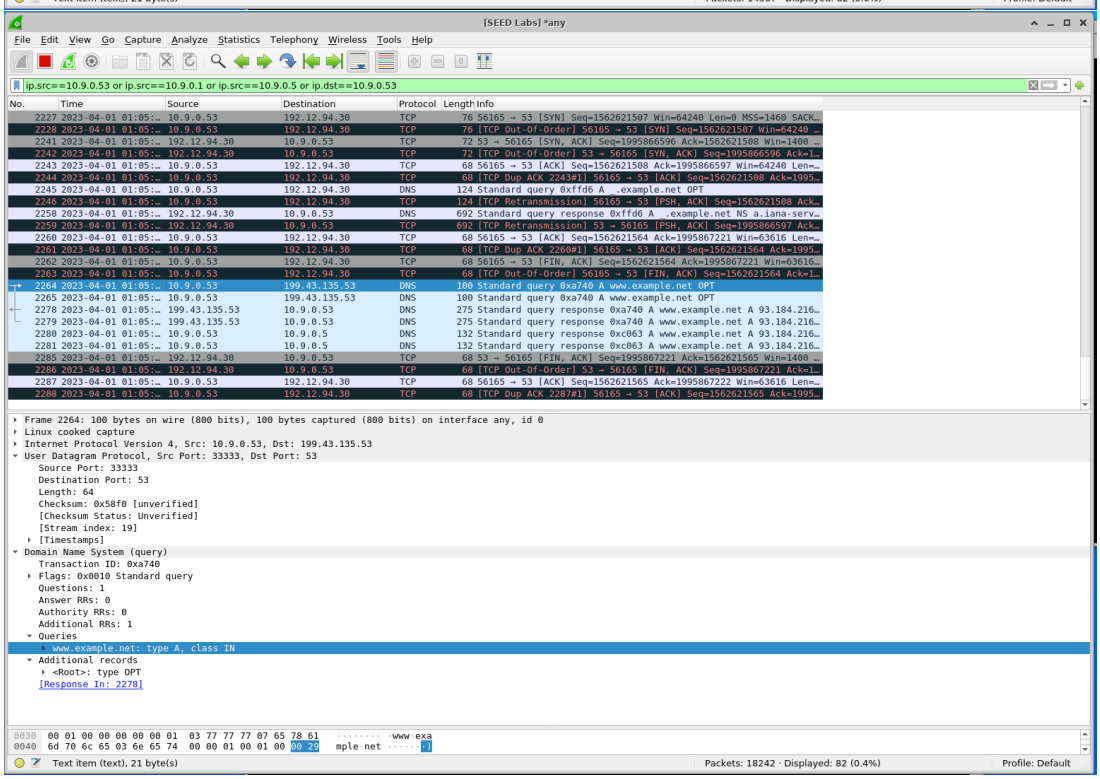


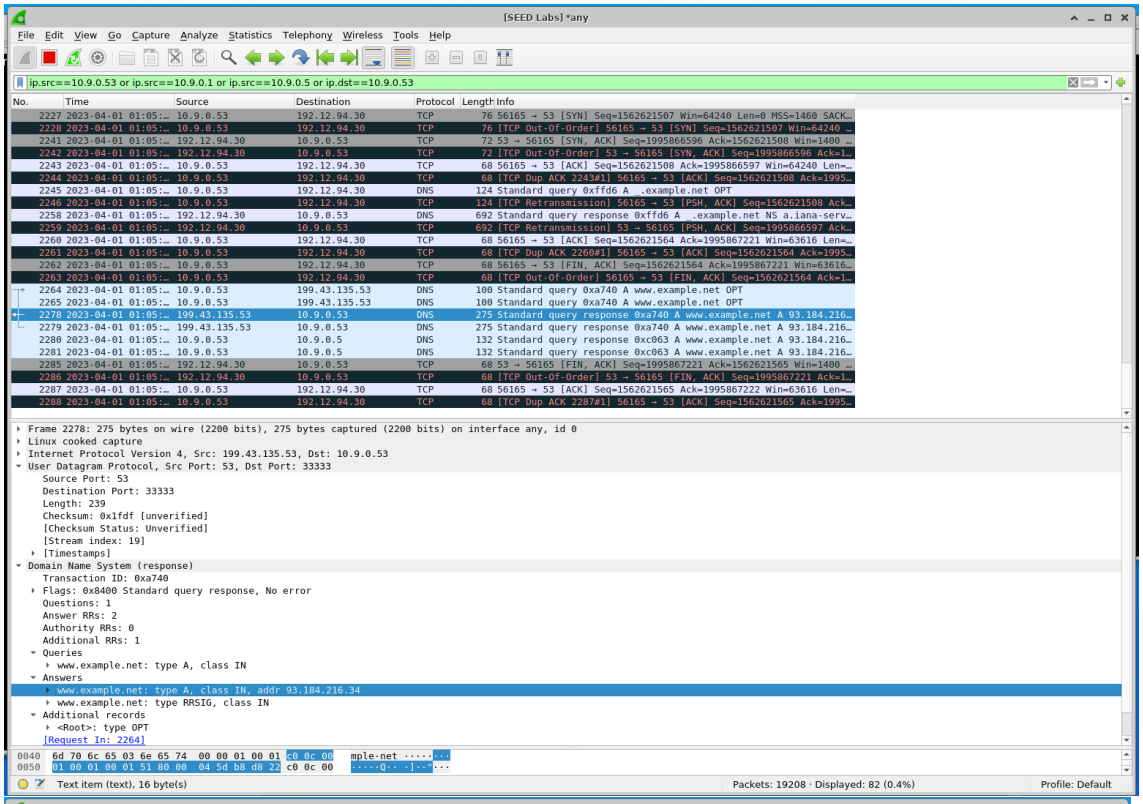
3. Run *dig* for www.example.net on seed-user and note the output.

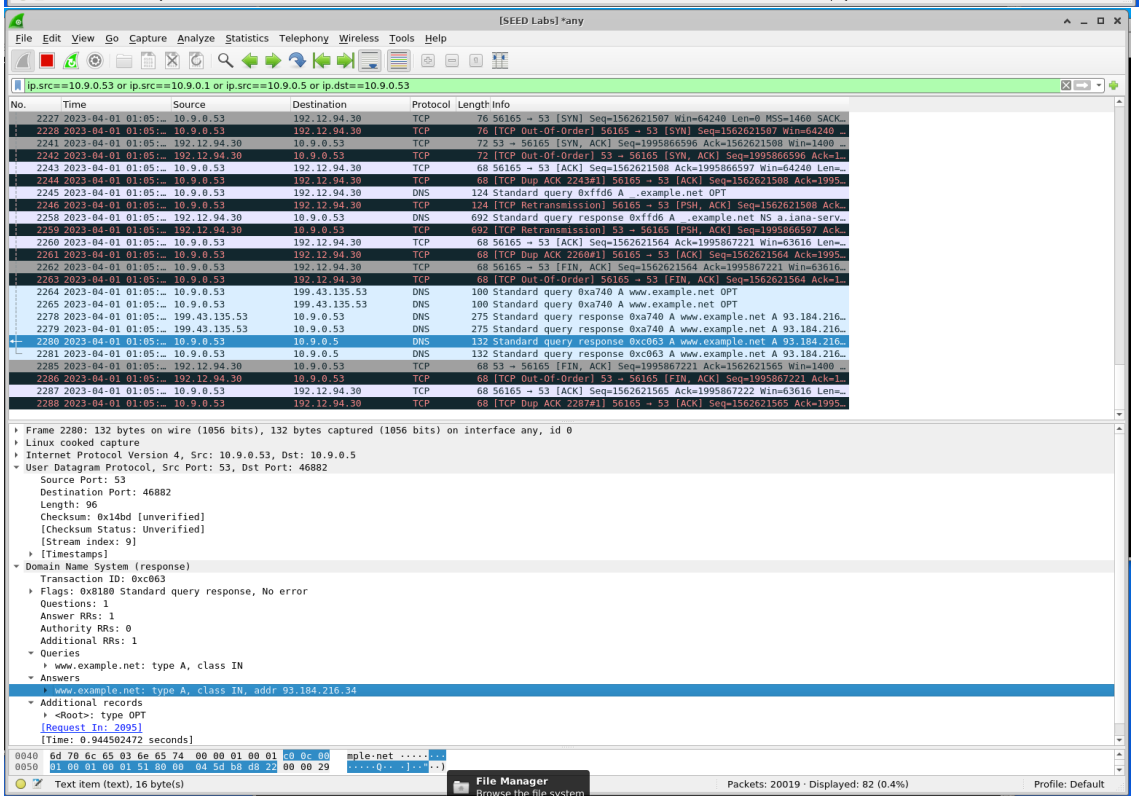
```
root@dfd9535c551a:/# dig www.example.net

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49251
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7d5a40d808d042e1010000006427834 28e4ba213dcc7eded (good)
;; QUESTION SECTION:
;www.example.net.               IN      A

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

;; Query time: 948 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 01 01:05:06 UTC 2023
;; MSG SIZE  rcvd: 88

root@dfd9535c551a:/#
```

4. Review Wireshark to see the conversation between seed-user and local-dns-server-10.9.0.53.






[SEED Labs] *any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.src==10.9.0.53 or ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.dst==10.9.0.53
No. Time Source Destination Protocol Length Info
2227 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 76 56165 → 53 [SYN] Seq=1562621507 Win=64240 Len=0 MSS=1460 SACK…
2228 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 76 [TCP Out-Of-Order] 56165 → 53 [SYN] Seq=1562621507 Win=64240 …
2241 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 72 53 → 56165 [SYN, ACK] Seq=1995866596 Ack=1562621508 Win=1400 …
2242 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 72 [TCP Out-Of-Order] 53 → 56165 [SYN, ACK] Seq=1995866596 Ack=1…
2243 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [ACK] Seq=1562621508 Ack=1995866597 Win=64240 Len=…
2244 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Dup ACK 2243#1] 56165 → 53 [ACK] Seq=1562621508 Ack=1995…
2245 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 DNS 124 Standard query 0xffd6 A .example.net OPT
2246 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 124 [TCP Retransmission] 56165 → 53 [PSH, ACK] Seq=1562621508 Ack…
2258 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 DNS 692 Standard query response 0xffd6 A .example.net NS a.iana-serv…
2259 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 692 [TCP Retransmission] 53 → 56165 [PSH, ACK] Seq=1995866597 Ack…
2260 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [ACK] Seq=1562621564 Ack=1995867221 Win=63616 Len=…
2261 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Dup ACK 2260#1] 56165 → 53 [ACK] Seq=1562621564 Ack=1995…
2262 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [FIN, ACK] Seq=1562621564 Ack=1995867221 Win=63616…
2263 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Out-Of-Order] 56165 → 53 [FIN, ACK] Seq=1562621564 Ack=1…
2264 2023-04-01 01:05:… 10.9.0.53 199.43.135.53 DNS 100 Standard query 0xa740 A www.example.net OPT
2265 2023-04-01 01:05:… 10.9.0.53 199.43.135.53 DNS 100 Standard query 0xa740 A www.example.net OPT
2278 2023-04-01 01:05:… 199.43.135.53 10.9.0.53 DNS 275 Standard query response 0xa740 A www.example.net A 93.184.216…
2279 2023-04-01 01:05:… 199.43.135.53 10.9.0.53 DNS 275 Standard query response 0xa740 A www.example.net A 93.184.216…
2280 2023-04-01 01:05:… 10.9.0.53 10.9.0.5 DNS 132 Standard query response 0xc063 A www.example.net A 93.184.216…
2281 2023-04-01 01:05:… 10.9.0.53 10.9.0.5 DNS 132 Standard query response 0xc063 A www.example.net A 93.184.216…
2285 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 68 53 → 56165 [FIN, ACK] Seq=1995867221 Ack=1562621565 Win=1400 …
2286 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 68 [TCP Out-Of-Order] 53 → 56165 [FIN, ACK] Seq=1995867221 Ack=1…
2287 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [ACK] Seq=1562621565 Ack=1995867222 Win=63616 Len=…
2288 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Dup ACK 2287#1] 56165 → 53 [ACK] Seq=1562621565 Ack=1995…
Frame 2278: 275 bytes on wire (2200 bits), 275 bytes captured (2200 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 199.43.135.53, Dst: 10.9.0.53
User Datagram Protocol, Src Port: 53, Dst Port: 33333
Source Port: 53
Destination Port: 33333
Length: 239
Checksum: 0x1fdf [unverified]
[Checksum Status: Unverified]
[Stream index: 19]
[Timestamps]
Domain Name System (response)
Transaction ID: 0xa740
Flags: 0x8400 Standard query response, No error
Questions: 1
Answer RRs: 2
Authority RRs: 0
Additional RRs: 1
Queries
www.example.net: type A, class IN
Answers
www.example.net: type A, class IN, addr 93.184.216.34
www.example.net: type RRSIG, class IN
Additional records
<Root>: type OPT
[Request In: 2264]
0040 6d 70 6c 65 03 6e 65 74 00 00 01 00 01 c0 0c 00 mple·net ····· ··
0050 01 00 01 00 01 51 80 00 04 5d b8 d8 22 c0 0c 00 ·····Q·· ·]··· ··
Text item (text), 16 byte(s)
Packets: 19208 · Displayed: 82 (0.4%)
Profile: Default


[SEED Labs] *any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.src==10.9.0.53 or ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.dst==10.9.0.53
No. Time Source Destination Protocol Length Info
2227 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 76 56165 → 53 [SYN] Seq=1562621507 Win=64240 Len=0 MSS=1460 SACK…
2228 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 76 [TCP Out-Of-Order] 56165 → 53 [SYN] Seq=1562621507 Win=64240 …
2241 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 72 53 → 56165 [SYN, ACK] Seq=1995866596 Ack=1562621508 Win=1400 …
2242 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 72 [TCP Out-Of-Order] 53 → 56165 [SYN, ACK] Seq=1995866596 Ack=1…
2243 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [ACK] Seq=1562621508 Ack=1995866597 Win=64240 Len=…
2244 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Dup ACK 2243#1] 56165 → 53 [ACK] Seq=1562621508 Ack=1995…
2245 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 DNS 124 Standard query 0xffd6 A .example.net OPT
2246 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 124 [TCP Retransmission] 56165 → 53 [PSH, ACK] Seq=1562621508 Ack…
2258 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 DNS 692 Standard query response 0xffd6 A .example.net NS a.iana-serv…
2259 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 692 [TCP Retransmission] 53 → 56165 [PSH, ACK] Seq=1995866597 Ack…
2260 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [ACK] Seq=1562621564 Ack=1995867221 Win=63616 Len=…
2261 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Dup ACK 2260#1] 56165 → 53 [ACK] Seq=1562621564 Ack=1995…
2262 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [FIN, ACK] Seq=1562621564 Ack=1995867221 Win=63616…
2263 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Out-Of-Order] 56165 → 53 [FIN, ACK] Seq=1562621564 Ack=1…
2264 2023-04-01 01:05:… 10.9.0.53 199.43.135.53 DNS 100 Standard query 0xa740 A www.example.net OPT
2265 2023-04-01 01:05:… 10.9.0.53 199.43.135.53 DNS 100 Standard query 0xa740 A www.example.net OPT
2278 2023-04-01 01:05:… 199.43.135.53 10.9.0.53 DNS 275 Standard query response 0xa740 A www.example.net A 93.184.216…
2279 2023-04-01 01:05:… 199.43.135.53 10.9.0.53 DNS 275 Standard query response 0xa740 A www.example.net A 93.184.216…
2280 2023-04-01 01:05:… 10.9.0.53 10.9.0.5 DNS 132 Standard query response 0xc063 A www.example.net A 93.184.216…
2281 2023-04-01 01:05:… 10.9.0.53 10.9.0.5 DNS 132 Standard query response 0xc063 A www.example.net A 93.184.216…
2285 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 68 53 → 56165 [FIN, ACK] Seq=1995867221 Ack=1562621565 Win=1400 …
2286 2023-04-01 01:05:… 192.12.94.30 10.9.0.53 TCP 68 [TCP Out-Of-Order] 53 → 56165 [FIN, ACK] Seq=1995867221 Ack=1…
2287 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 56165 → 53 [ACK] Seq=1562621565 Ack=1995867222 Win=63616 Len=…
2288 2023-04-01 01:05:… 10.9.0.53 192.12.94.30 TCP 68 [TCP Dup ACK 2287#1] 56165 → 53 [ACK] Seq=1562621565 Ack=1995…
Frame 2280: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.9.0.53, Dst: 10.9.0.5
User Datagram Protocol, Src Port: 53, Dst Port: 46882
Source Port: 53
Destination Port: 46882
Length: 96
Checksum: 0x14bd [unverified]
[Checksum Status: Unverified]
[Stream index: 9]
[Timestamps]
Domain Name System (response)
Transaction ID: 0xc063
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 1
Queries
www.example.net: type A, class IN
Answers
www.example.net: type A, class IN, addr 93.184.216.34
Additional records
<Root>: type OPT
[Request In: 2095]
[Time: 0.944502472 seconds]
0040 6d 70 6c 65 03 6e 65 74 00 00 01 00 01 c0 0c 00 mple·net ····· ··
0050 01 00 01 00 01 51 80 00 04 5d b8 d8 22 00 00 29 ·····Q·· ·]··"··)
Text item (text), 16 byte(s)
File Manager
Browse the file system
Packets: 20019 · Displayed: 82 (0.4%)
Profile: Default

5. Flush DNS cache on local-dns-server-10.9.0.53.

```
local-dns-server-10.9.0.53
File  Edit  View  Terminal  Tabs  Help
root@9504fa5ca8fd:/# rndc flush
root@9504fa5ca8fd:/#
```

6. Run *dns_spoof.py* on seed-attacker.

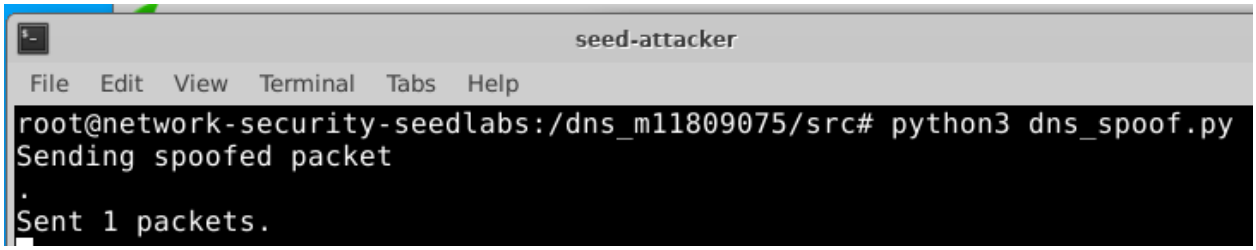
```
seed-attacker
File  Edit  View  Terminal  Tabs  Help
root@network-security-seedlabs:/dns_m11809075/src# python3 dns_spoof.py
```

7. Run *dig* for www.example.net on seed-user and note the output.

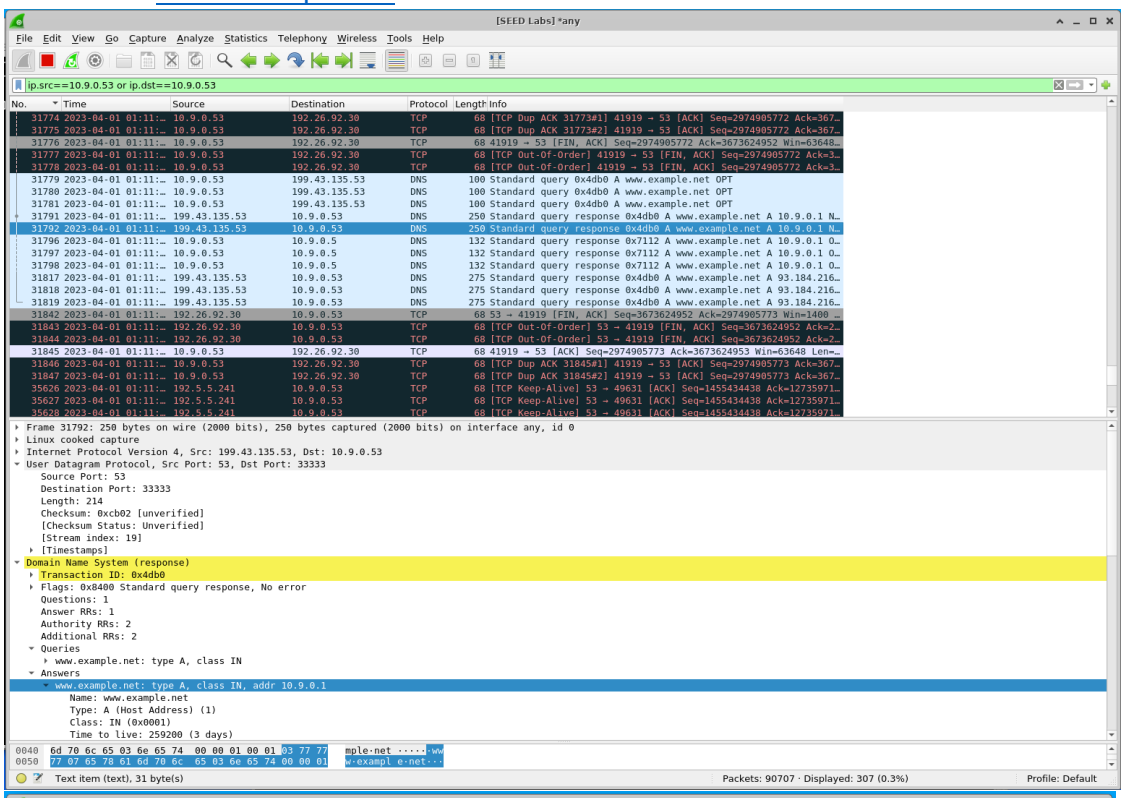
```
user-10.9.0.5
File  Edit  View  Terminal  Tabs  Help
root@dfd9535c551a:/# dig www.example.net

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28946
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 20442a64f3a465a601000000642784bd01656e61f0db2705 (good)
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        259200  IN      A       10.9.0.1

;; Query time: 896 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 01 01:11:25 UTC 2023
;; MSG SIZE  rcvd: 88

root@dfd9535c551a:/#
```

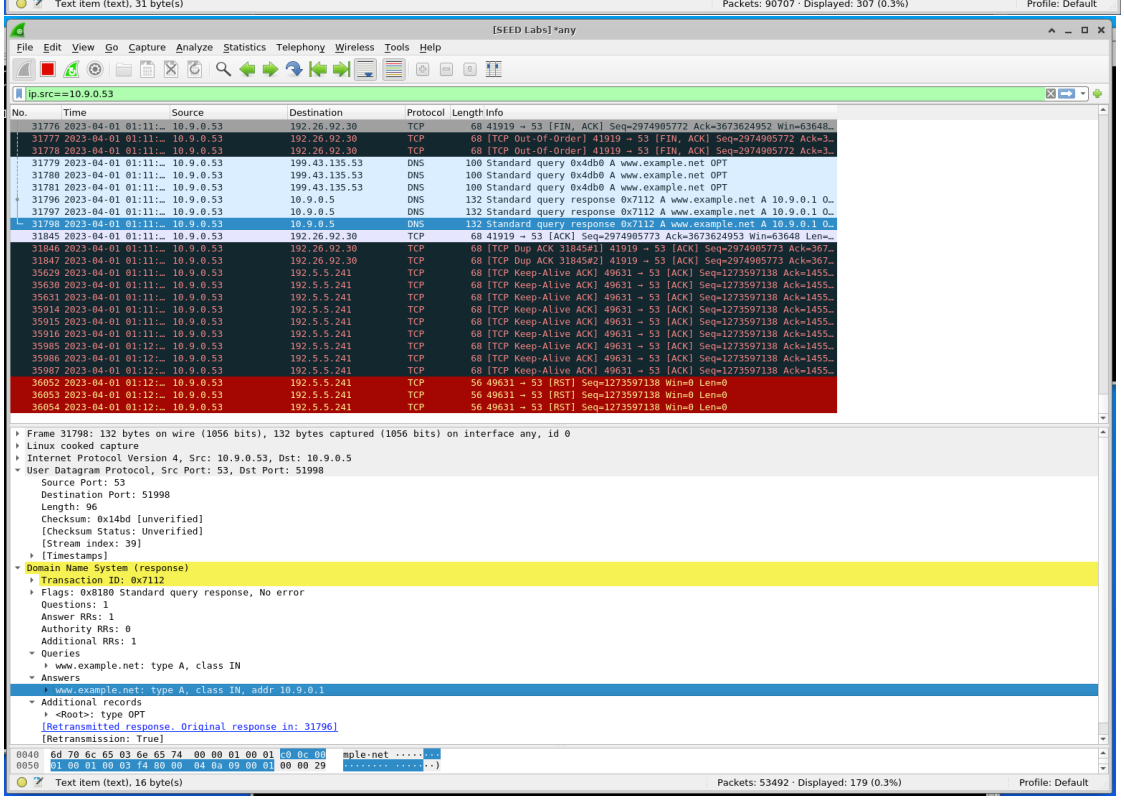
8. Compare the outputs from steps 3 and 7. The output of step 7 should indicate that the IP addressed returned to seed-user is now the malicious IP.

   The IP address returned prior to the attack was 93.184.216.34, the IP address returned after the attack is 10.9.0.1, the malicious IP of seed-attacker.



9. Review Wireshark to see the conversation between seed-user and local-dns-server-10.9.0.53. Note how seed-attacker is the sender of the reply to local-dns-server-10.9.0.53, which local-dns-server-10.9.0.53 then sends back to seed-user as the IP address for www.example.net.





10. Stop the attack *dns_spoof.py*.

```
seed-attacker
File  Edit  View  Terminal  Tabs  Help
root@network-security-seedlabs:/dns_m11809075/src# python3 dns_spoof.py
Sending spoofed packet
.
Sent 1 packets.
^Croot@network-security-seedlabs:/dns_m11809075/src#
```

11. Run *dig* on seed-user again for www.example.net

```
user-10.9.0.5
File  Edit  View  Terminal  Tabs  Help
root@dfd9535c551a:/# dig www.example.net

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11961
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b1e26f588210e7290100000064278769dbd6e36164df9303 (good)
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        258516  IN      A       10.9.0.1

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 01 01:22:49 UTC 2023
;; MSG SIZE  rcvd: 88

root@dfd9535c551a:/# ~
```

Note: even without the attack running, our malicious IP address is still returned when seed-user requests a lookup for www.example.net.

Screenshots of each step

See screenshots shown with steps in “How do you setup the User machine and Server machine?” and “How do you perform the attack in your VM?”

Was the attack successful?

Include screenshots to show the attack is successful and can render an incorrect IP on both the User machine and Server machine

Yes, the attack was successful. When the attack is running and local-dns-server receives a DNS Query for www.example.net, local-dns-server sends a query to it's upstream DNS server. The attack sees this and replies to local-dns-server acting as the upstream DNS server and provides a result for the query. The result contains our malicious IP address of seed-attacker which local-dns-server then stores in it's cache. It then sends this response to seed-user as it was the one who requested the lookup. The attack can then be stopped and can still be seen that the malicious IP is returned when a lookup is requested.

See screenshots in steps 8, 9, and 11 of "How do you perform the attack in your VM?"