

CS 5153/5053 Network Security, Spring 2023

Project 4: Local DNS Cache Poisoning

Report

Student: Austin Tyler Conn

Contents

Link to Source Code.....	3
Host Environment Used	3
Docker Information.....	3
Assumptions.....	3
How do you setup the User machine and Server machine?	4
User Machine (Task 1).....	4
Server Machine (Task 2).....	5
<u> </u> Disclaimer:	5
How do you perform the attack in your VM?	11
Screenshots of each step	18
Was the attack successful?	19

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 20 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:

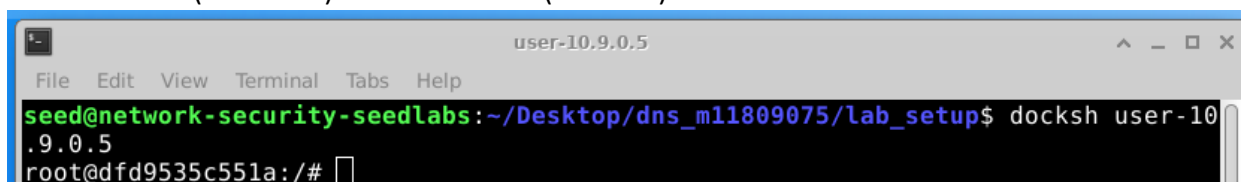
- [seed-labs/seedvm-cloud.md at master · seed-labs/seed-labs \(github.com\)](#)
- [seed-labs/create_vm_gcp.md at master · seed-labs/seed-labs \(github.com\)](#)

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$
```

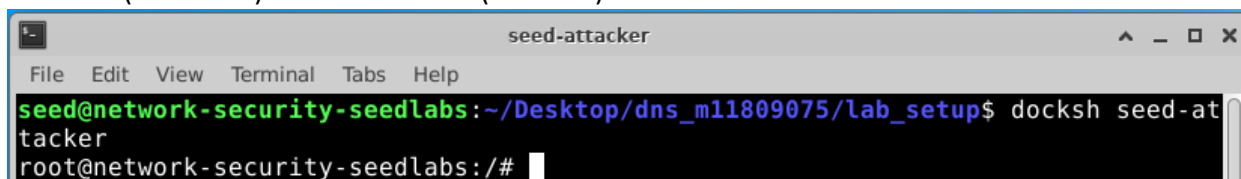
Assumptions

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



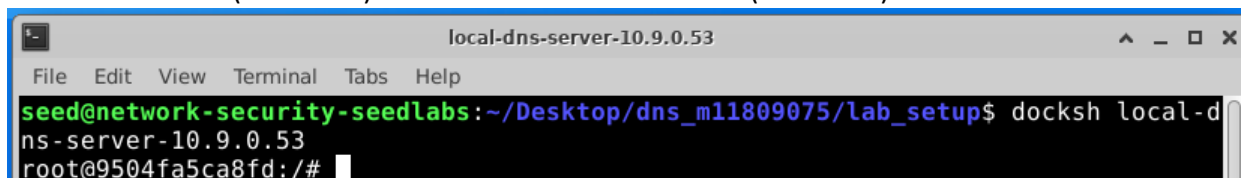
```
user-10.9.0.5
File Edit View Terminal Tabs Help
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ docksh user-10.9.0.5
root@dfd9535c551a:/#
```

- 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-attacker
root@network-security-seedlabs:/#
```

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

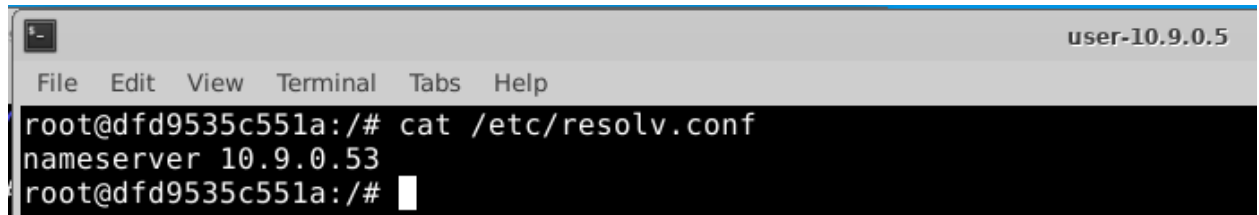


```
local-dns-server-10.9.0.53
File Edit View Terminal Tabs Help
seed@network-security-seedlabs:~/Desktop/dns_m11809075/lab_setup$ docksh local-dns-server-10.9.0.53
root@9504fa5ca8fd:/#
```

How do you setup the User machine and Server machine?

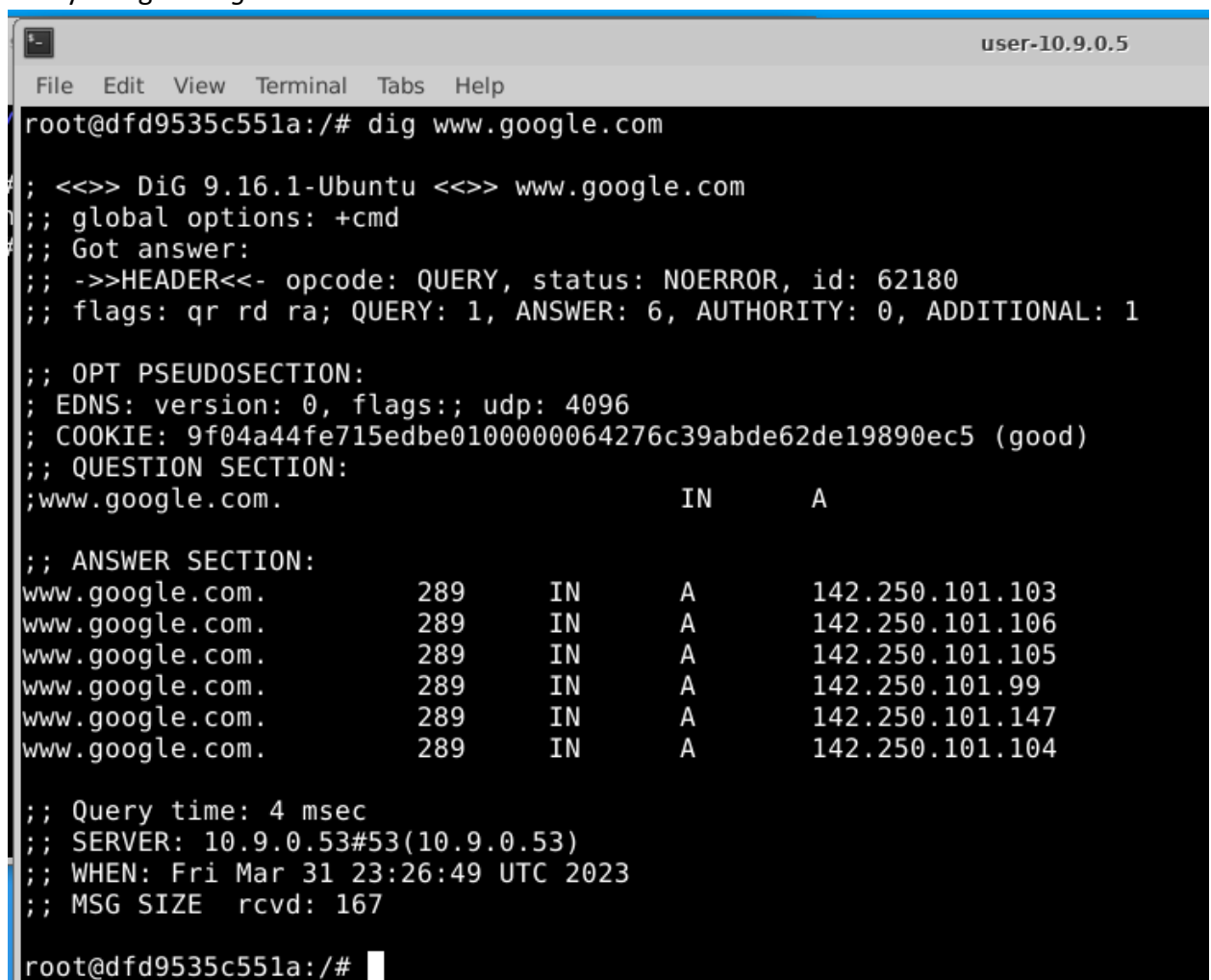
User Machine (Task 1)

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



```
user-10.9.0.5
File Edit View Terminal Tabs Help
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.



```
user-10.9.0.5
File Edit View Terminal Tabs Help
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

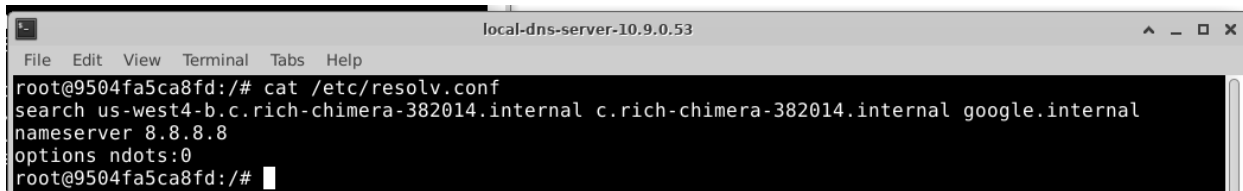
root@dfd9535c551a:/#
```

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)

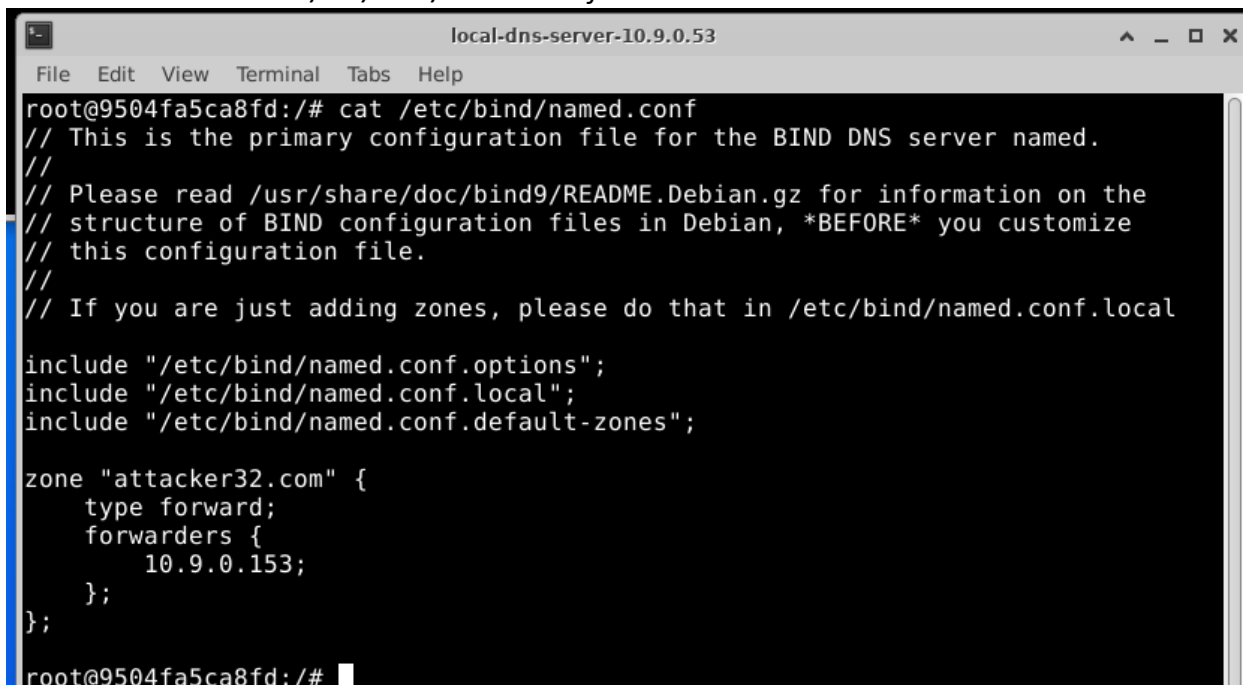
Disclaimer: Due to using SEED Labs Docker Containers, the server DOES need to have its /etc/resolv.conf file updated to use a DNS server OTHER than docker's internal DNS server. Without doing this, the local-dns-server-10.9.0.53 will not reach out for queries, thus inhibiting the attacker being able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

A terminal window titled 'local-dns-server-10.9.0.53' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is 'root@9504fa5ca8fd:/' and the command 'cat /etc/resolv.conf' has been executed. The output shows a search path for 'us-west4-b.c.rich-chimera-382014.internal', 'c.rich-chimera-382014.internal', and 'google.internal', followed by 'nameserver 8.8.8.8' and 'options ndots:0'.

```
root@9504fa5ca8fd:/# cat /etc/resolv.conf
search us-west4-b.c.rich-chimera-382014.internal c.rich-chimera-382014.internal google.internal
nameserver 8.8.8.8
options ndots:0
root@9504fa5ca8fd:/#
```

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

A terminal window titled 'local-dns-server-10.9.0.53' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is 'root@9504fa5ca8fd:/' and the command 'cat /etc/bind/named.conf' has been executed. The output shows a BIND configuration file with comments, include statements for options, local, and default zones, and a zone definition for 'attacker32.com' pointing to 10.9.0.153.

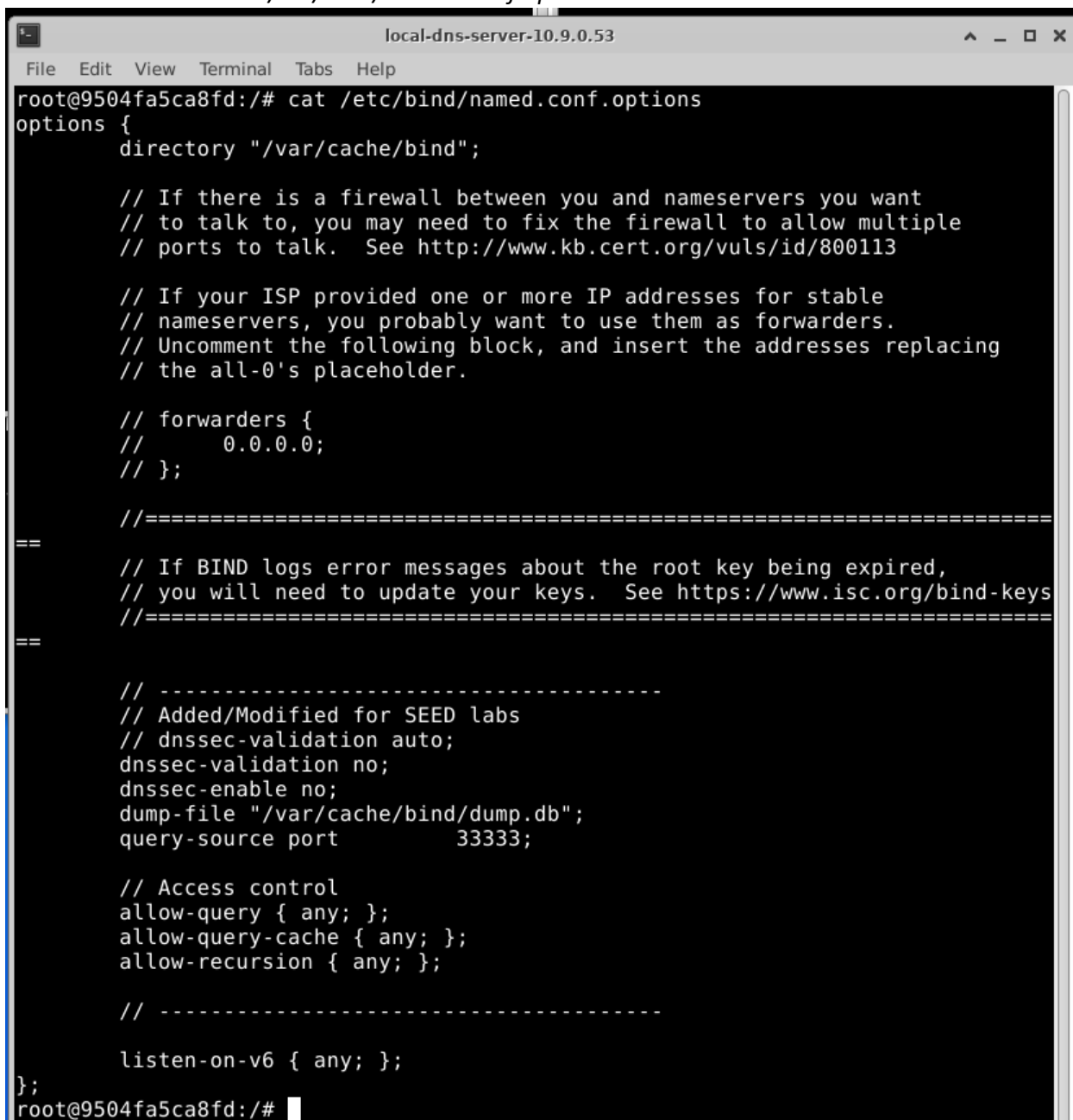
```
root@9504fa5ca8fd:/# cat /etc/bind/named.conf
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "attacker32.com" {
    type forward;
    forwarders {
        10.9.0.153;
    };
};

root@9504fa5ca8fd:/#
```

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



The screenshot shows a terminal window titled "local-dns-server-10.9.0.53". The user is at the root prompt and has executed the command `cat /etc/bind/named.conf.options`. The output shows the configuration file's contents, which include comments about firewall rules, forwarders, and logging, as well as configuration lines for `directory`, `dnssec-validation`, `dump-file`, `query-source`, `allow-query`, and `listen-on-v6`.

```
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:/#
```

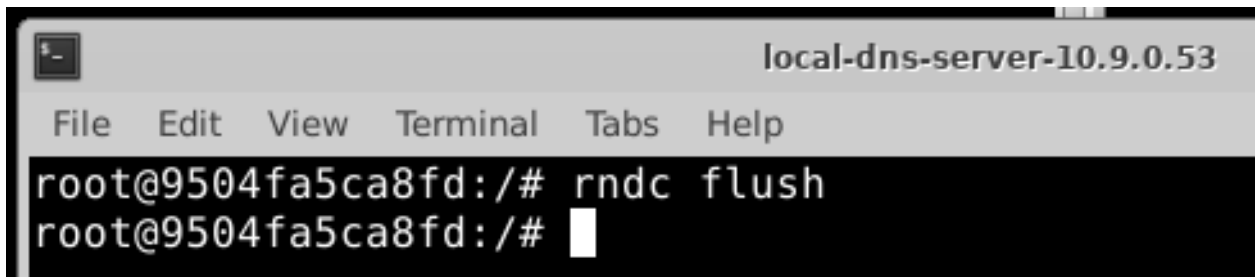
4. Dump the DNS cache to the file specified by the `dump-file` line in step 2 (`/var/cache/bind/dump.db`).



The screenshot shows a terminal window titled "local-dns-server-10.9.0.53". The user is at the root prompt and has executed the command `rndc dumpdb -cache`. The prompt has changed to `root@9504fa5ca8fd:/#` and a tilde character is visible at the end of the line.

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

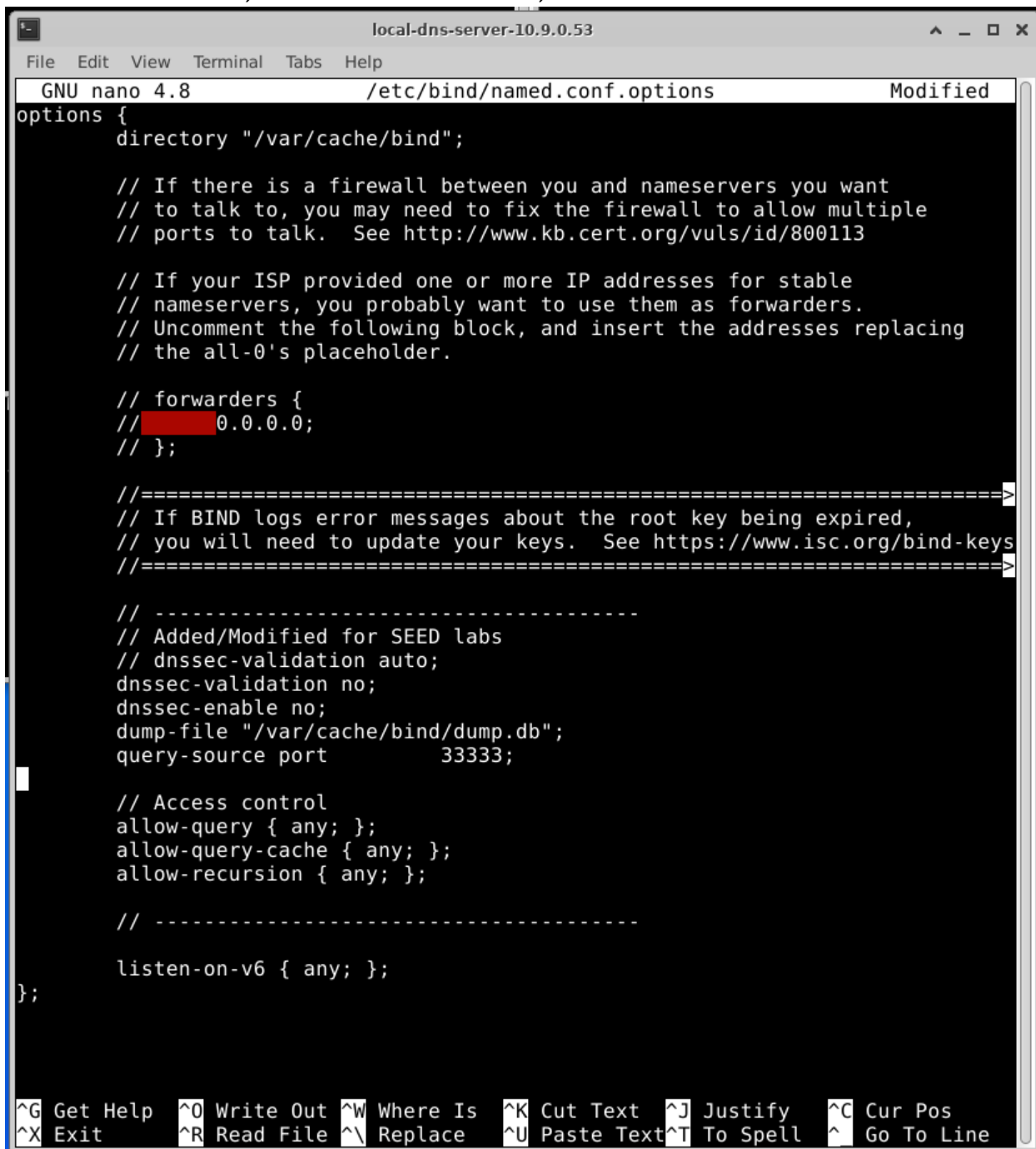
5. Flush the DNS cache.



A terminal window titled "local-dns-server-10.9.0.53" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the prompt "root@9504fa5ca8fd:/#" followed by the command "rndc flush" and a new prompt "root@9504fa5ca8fd:/#" with a cursor.

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

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
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*.

```
user-10.9.0.5
File Edit View Terminal Tabs Help
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: ec5bb6b0618999460100000064277293d7dc362a4b842336 (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.

The image shows a Wireshark network capture of a DNS query and response. The packet list on the left shows several packets, with packet 3827 (a DNS query) and packet 3828 (a DNS response) highlighted. The packet details pane on the right shows the structure of these packets. The query (3827) is for 'www.facebook.com' and the response (3828) contains the IP address '104.134.31.12'.

No.	Time	Source	Destination	Protocol	Length	Info
3738	2023-03-31 23:58:...	10.9.0.5	10.9.0.53	DNS	101	Standard query 0x8753 A www.facebook.com OPT
3731	2023-03-31 23:58:...	10.9.0.5	10.9.0.53	DNS	101	Standard query 0x8753 A www.facebook.com OPT
3732	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	DNS	99	Standard query 0x8dcf A _._.facebook.com OPT
3733	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	DNS	99	Standard query 0x8dcf A _._.facebook.com OPT
3743	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	76	54475 → 53 [SYN] Seq=1031200112 Win=64240 Len=0 MSS=1460 SACK...
3744	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	76	[TCP Out-Of-Order] 54475 → 53 [SYN] Seq=1031200112 Win=64240...
3758	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	54475 → 53 [ACK] Seq=1031200113 Ack=2750211540 Win=64240 Len=...
3759	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	[TCP Dup ACK 375841] 54475 → 53 [ACK] Seq=1031200113 Ack=2750...
3763	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	DNS	125	Standard query 0x8c24 A _._.facebook.com OPT
3764	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	125	[TCP Retransmit] 54475 → 53 [ACK] Seq=1031200113 Ack=2750...
3774	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	54475 → 53 [ACK] Seq=1031200170 Ack=2750212377 Win=63612 Len=...
3775	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	[TCP Dup ACK 377481] 54475 → 53 [ACK] Seq=1031200170 Ack=2750...
3779	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	54475 → 53 [FIN, ACK] Seq=1031200170 Ack=2750212377 Win=63612...
3780	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	[TCP Out-Of-Order] 54475 → 53 [FIN, ACK] Seq=1031200170 Ack=2...
3784	2023-03-31 23:58:...	10.9.0.53	185.89.218.12	DNS	101	Standard query 0x8c24 A www.facebook.com OPT
3785	2023-03-31 23:58:...	10.9.0.53	185.89.218.12	DNS	101	Standard query 0x8c24 A www.facebook.com OPT
3794	2023-03-31 23:58:...	10.9.0.53	129.134.31.12	DNS	104	Standard query 0x8c2e A _._.cl0r.facebook.com OPT
3795	2023-03-31 23:58:...	10.9.0.53	129.134.31.12	DNS	104	Standard query 0x8c2e A _._.cl0r.facebook.com OPT
3806	2023-03-31 23:58:...	10.9.0.53	185.89.218.11	DNS	112	Standard query 0x8d3a A star-mini.cl0r.facebook.com OPT
3807	2023-03-31 23:58:...	10.9.0.53	185.89.218.11	DNS	112	Standard query 0x8d3a A star-mini.cl0r.facebook.com OPT
3816	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	54475 → 53 [ACK] Seq=1031200171 Ack=2750212378 Win=63612 Len=...
3817	2023-03-31 23:58:...	10.9.0.53	192.5.6.30	TCP	68	[TCP Dup ACK 381641] 54475 → 53 [ACK] Seq=1031200171 Ack=2750...
3827	2023-03-31 23:58:...	10.9.0.53	10.9.0.5	DNS	174	Standard query response 0x8753 A www.facebook.com CNAME star-...
3828	2023-03-31 23:58:...	10.9.0.5	10.9.0.53	DNS	174	Standard query response 0x8753 A www.facebook.com CNAME star-...

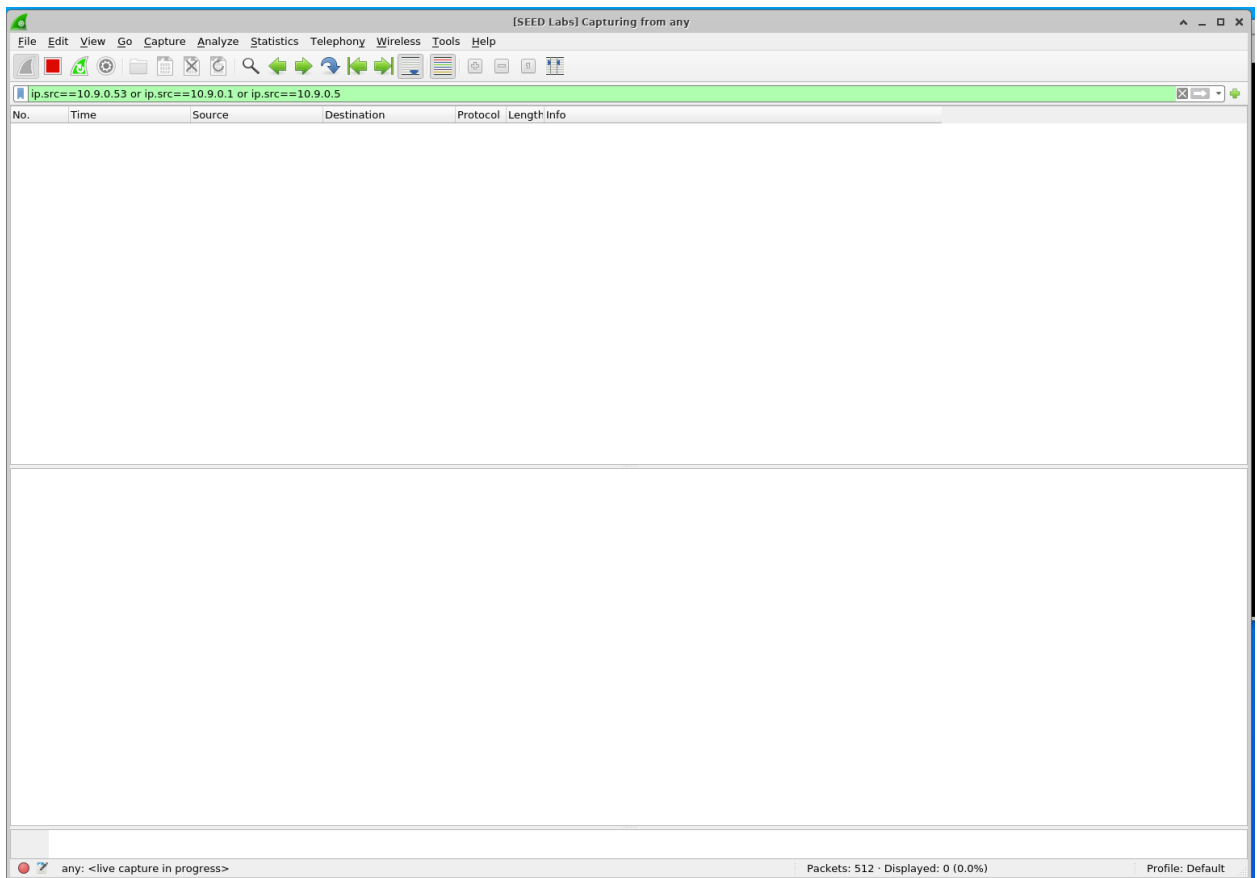
Frame 3730: 101 bytes on wire (808 bits), 101 bytes captured (808 bits) on interface any, id 0
 Linux cooked capture
 Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.53
 User Datagram Protocol, Src Port: 57763, Dst Port: 53
 Source Port: 57763
 Destination Port: 53
 Length: 65
 Checksum: 0x149e [unverified]
 [Checksum Status: Unverified]
 [Stream index: 0]
 [Timestamps]
 Domain Name System (query)
 Transaction ID: 0x8753
 Flags: 0x0120 Standard query
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 1
 Queries
 www.facebook.com: type A, class IN
 Additional records
 *Root: type OPT
 [Response In: 3827]

0030 00 01 00 00 00 00 01 03 77 77 08 66 61 63www-fac
 0040 55 62 6f 6f 6b 03 6f 6d 00 00 01 00 00 ebook.co m-----

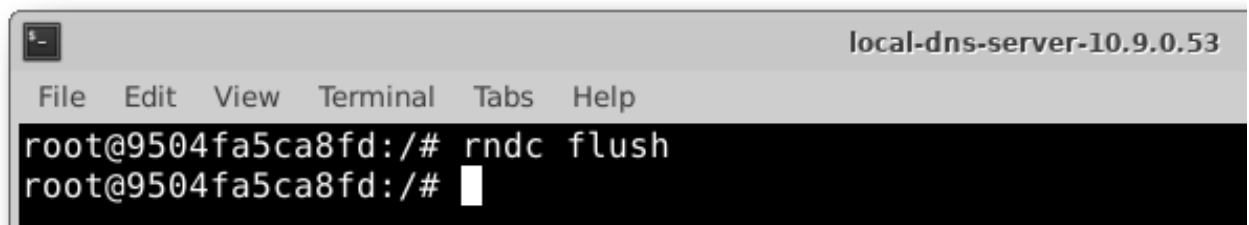
Text item (text), 22 byte(s) Packets: 8393 · Displayed: 24 (0.3%) Profile: Default

How do you perform the attack in your VM?

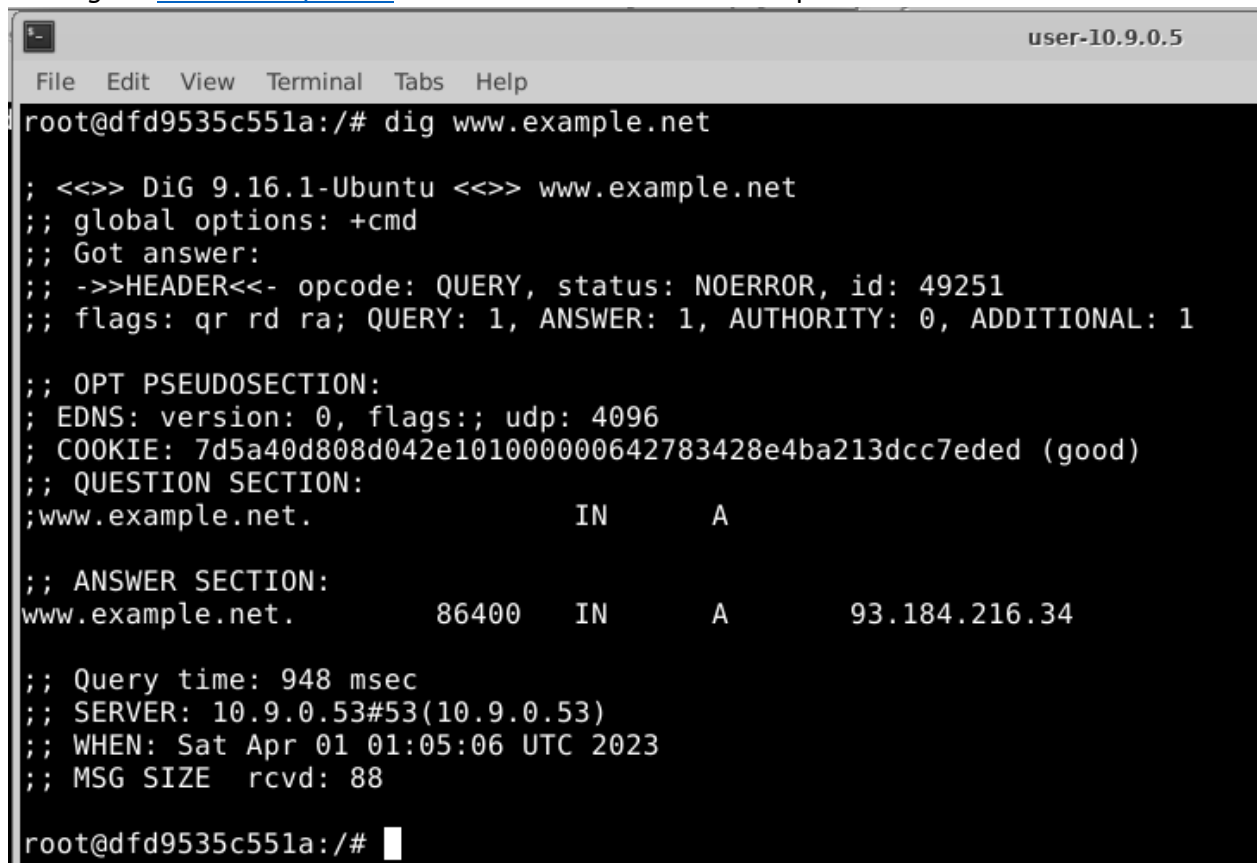
1. Start Wireshark monitoring traffic between seed-attacker, seed-user, and local-dns-server-10.9.0.53.



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: 7d5a40d808d042e101000000642783428e4ba213dcc7eded (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:/#
```

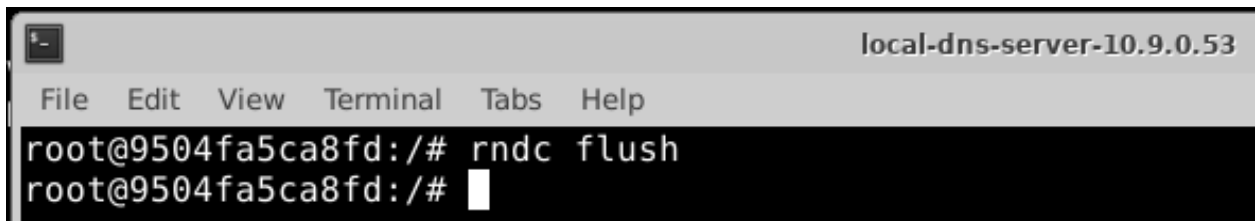
4. Review Wireshark to see the conversation between seed-user and local-dns-server-10.9.0.53.

The top screenshot shows a Wireshark packet capture with a filter for IP addresses 10.9.0.53 and 10.9.0.5. The packet list shows a series of DNS queries and responses. The packet details pane shows the structure of a DNS query, including the transaction ID, flags, questions, and additional records. The packet bytes pane shows the raw data of the packet.

The bottom screenshot shows another Wireshark packet capture with a similar filter. The packet list shows a series of DNS queries and responses. The packet details pane shows the structure of a DNS query, including the transaction ID, flags, questions, and additional records. The packet bytes pane shows the raw data of the packet.

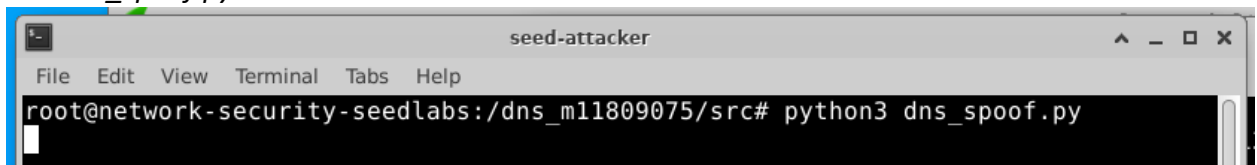
The image displays two screenshots of the Wireshark network protocol analyzer. The top screenshot shows a packet capture on interface 'any' with a filter for 'ip.src==10.9.0.53 or ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.dst==10.9.0.53'. Packet 2278 is selected, showing details for an Ethernet II frame, an IPv4 packet from 199.43.135.53 to 10.9.0.53, and a DNS response for 'www.example.net'. The bottom screenshot shows packet 2280 selected, which is a DNS response for 'www.example.net' from 10.9.0.53 to 10.9.0.5. The details pane for packet 2280 shows a standard query response with no error, and the raw data pane shows the packet structure in hexadecimal and ASCII.

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



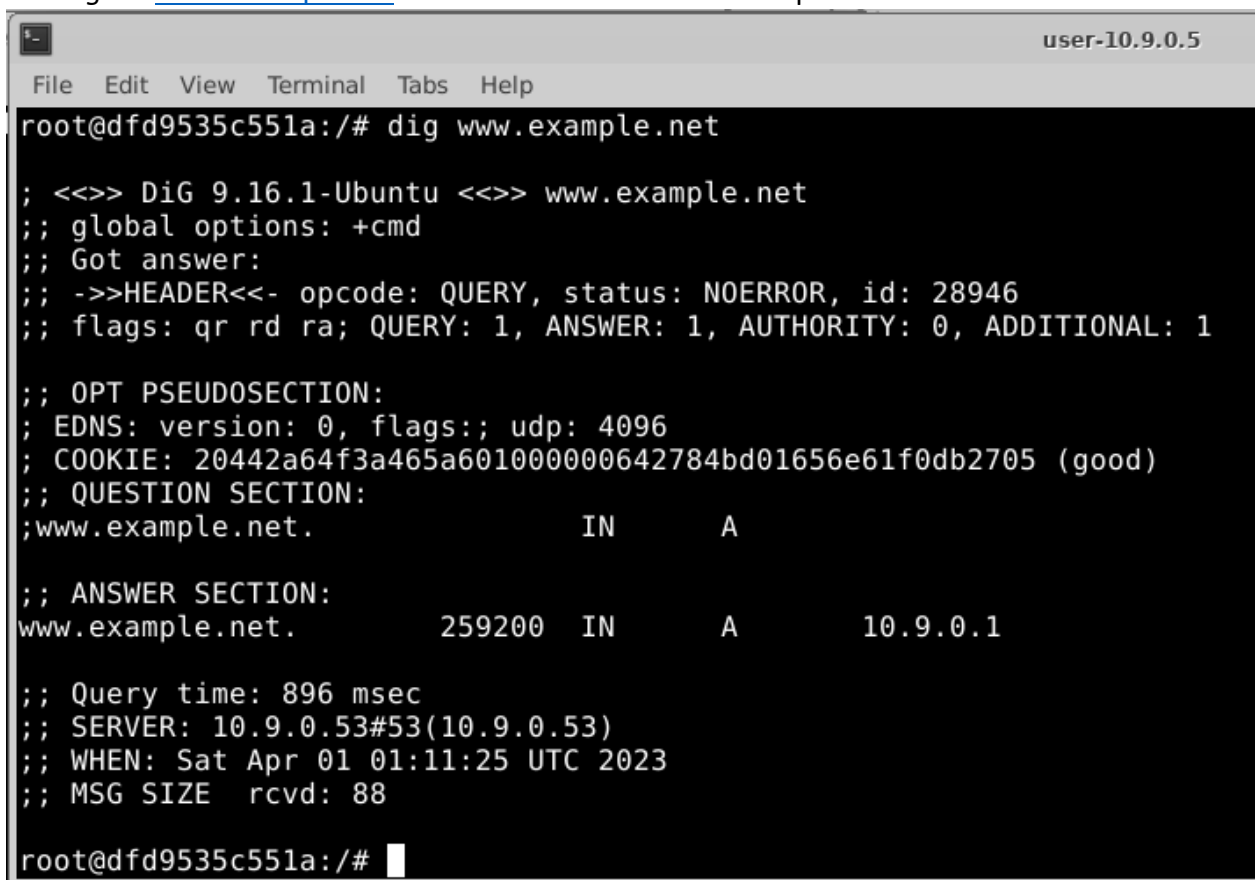
```
local-dns-server-10.9.0.53
File Edit View Terminal Tabs Help
root@9504fa5ca8fd:/# rndc flush
root@9504fa5ca8fd:/#
```

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

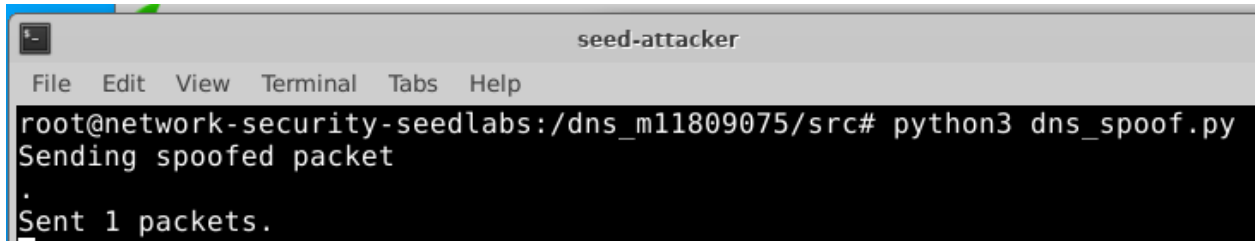
;; ANSWER SECTION:
www.example.net.                259200  IN      A      10.9.0.1

;; 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.



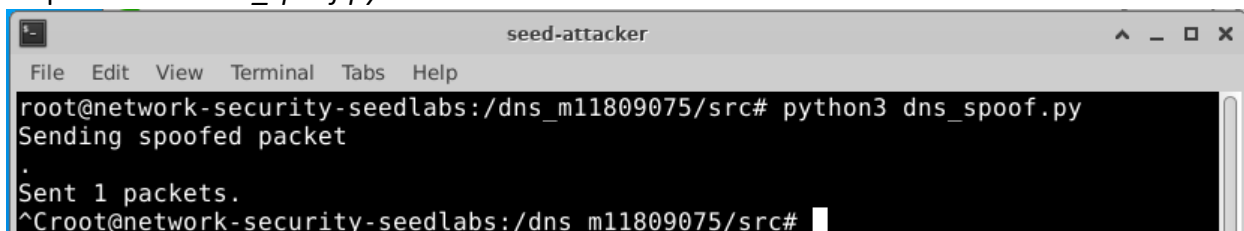
```
seed-attacker
File Edit View Terminal Tabs Help
root@network-security-seedlabs:/dns_m11809075/src# python3 dns_spoof.py
Sending spoofed packet
.
Sent 1 packets.
```


- 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.

The top screenshot shows a Wireshark packet capture with the filter `ip.src==10.9.0.53 or ip.dst==10.9.0.53`. The packet list shows several DNS and TCP packets. Packet 31792 is a DNS Standard query response from 10.9.0.53 to 199.43.135.53. The packet details pane shows the Domain Name System (response) section, indicating a successful query for `www.example.net` with IP address 10.9.0.1.

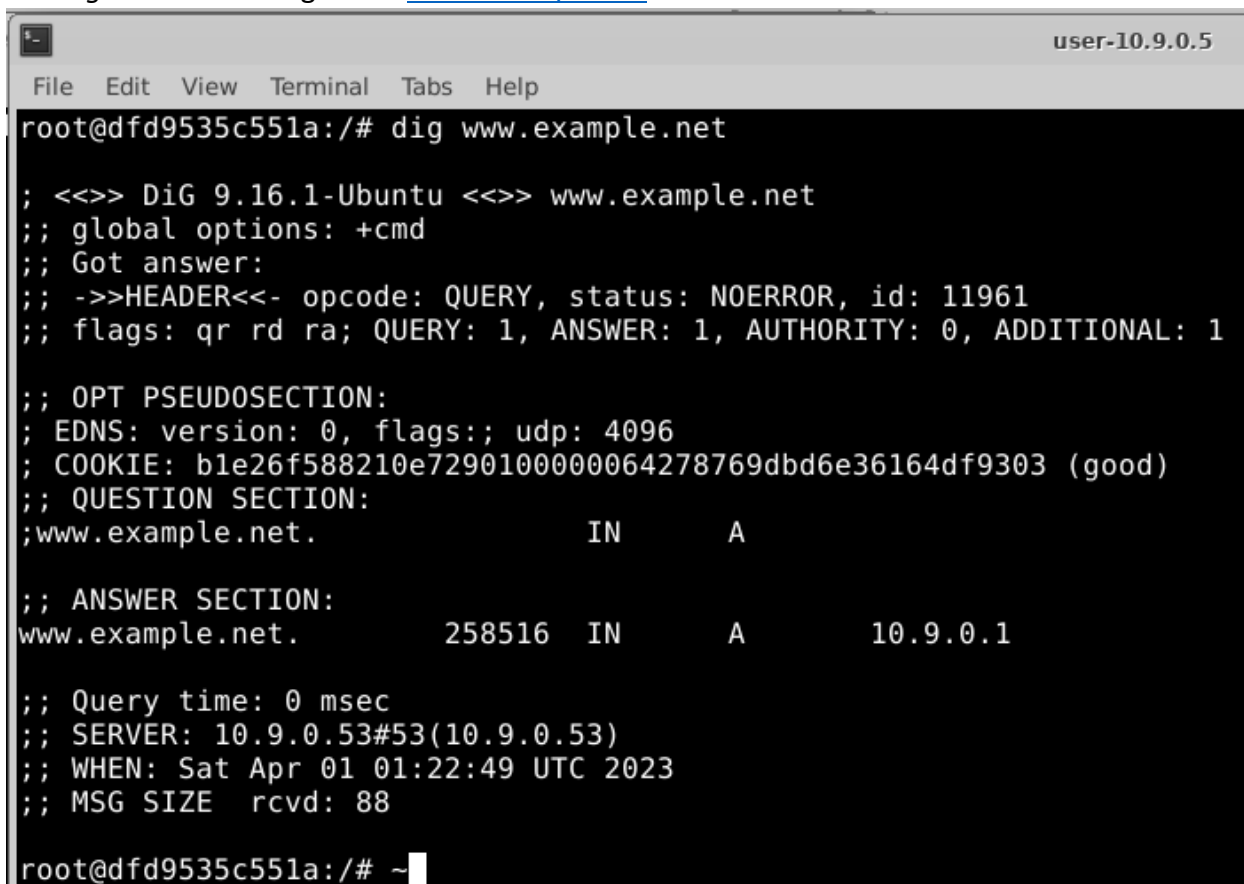
The bottom screenshot shows a Wireshark packet capture with the filter `ip.src==10.9.0.53`. The packet list shows several TCP packets. Packet 36052 is a TCP RST (Reset) packet from 10.9.0.53 to 192.5.5.241. The packet details pane shows the Domain Name System (response) section, indicating a successful query for `www.example.net` with IP address 10.9.0.1.

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?"