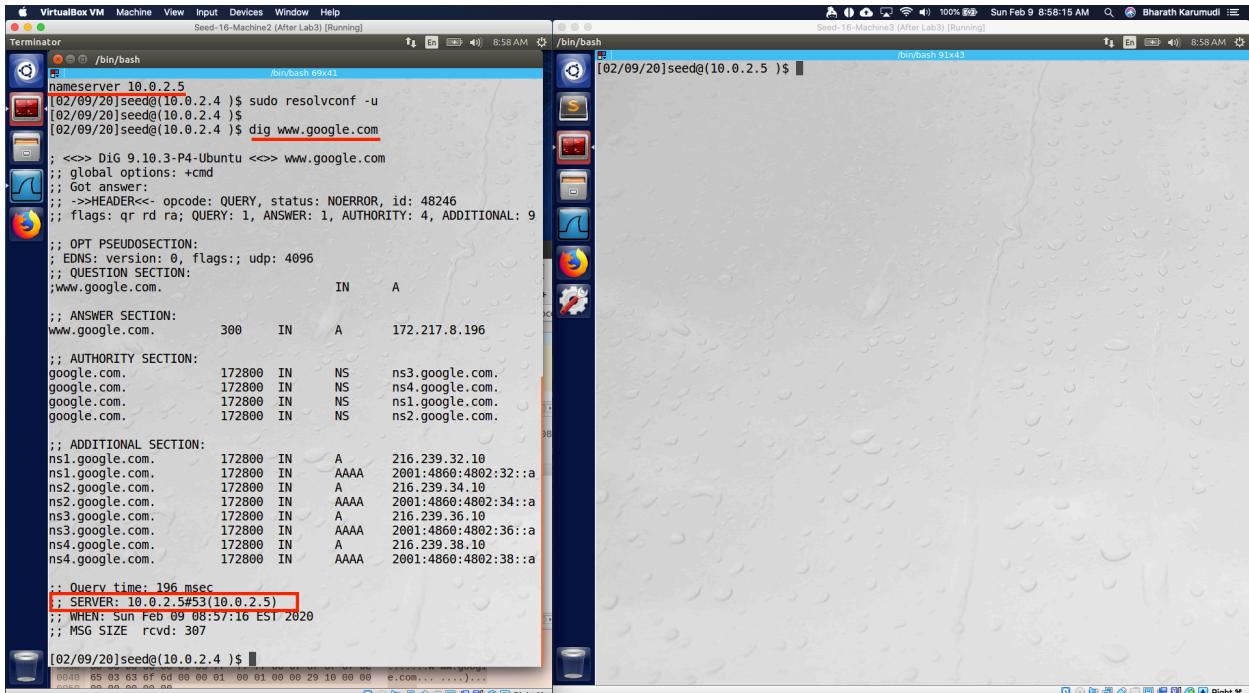


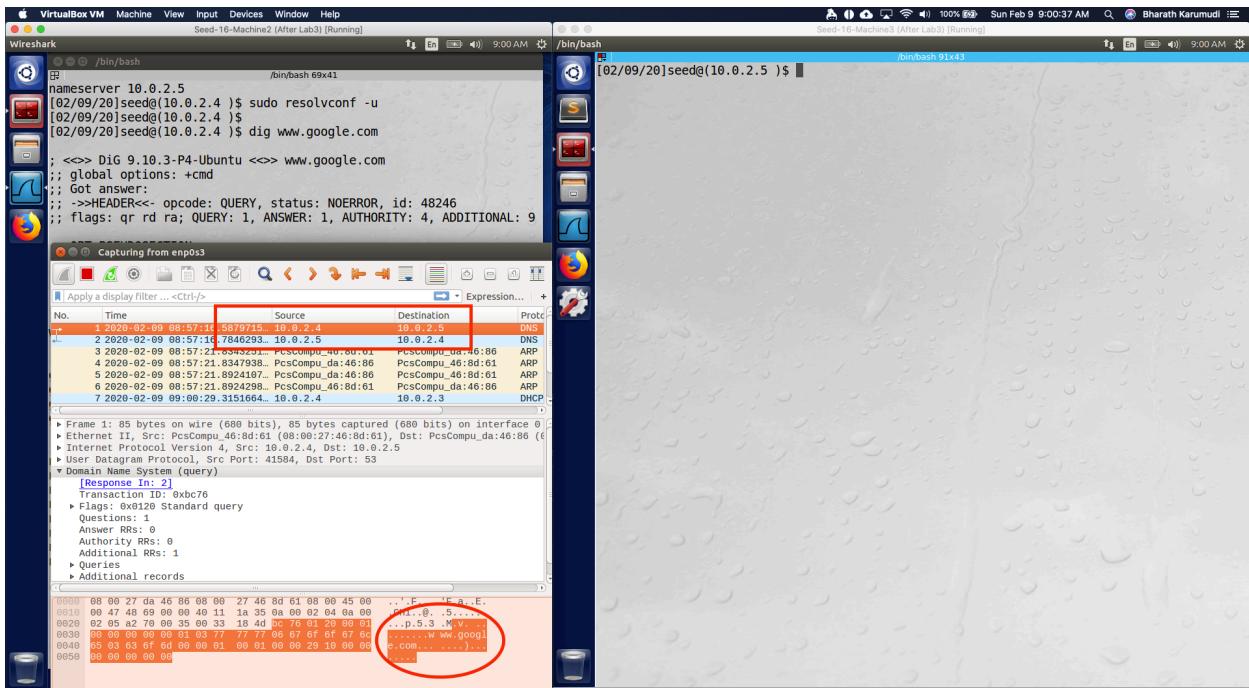
**Name:** Bharath Karumudi  
**Lab:** Local DNS Attacks

**Task 1: Configure the User Machine**



```
[02/09/20]seed@(10.0.2.4) $ dig www.google.com
; <>> Dig 9.10.3-P4-Ubuntu <>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>>HEADER<- opcode: QUERY, status: NOERROR, id: 48246
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com.          IN      A
;; ANSWER SECTION:
www.google.com.        300     IN      A      172.217.8.196
;; AUTHORITY SECTION:
google.com.            172800  IN      NS     ns3.google.com.
google.com.            172800  IN      NS     ns4.google.com.
google.com.            172800  IN      NS     ns1.google.com.
google.com.            172800  IN      NS     ns2.google.com.
;; ADDITIONAL SECTION:
ns1.google.com.        172800  IN      A      216.239.32.10
ns1.google.com.        172800  IN      AAAA   2001:4860:4802:32::a
ns2.google.com.        172800  IN      A      216.239.34.10
ns2.google.com.        172800  IN      AAAA   2001:4860:4802:34::a
ns3.google.com.        172800  IN      A      216.239.36.10
ns3.google.com.        172800  IN      AAAA   2001:4860:4802:36::a
ns4.google.com.        172800  IN      A      216.239.38.10
ns4.google.com.        172800  IN      AAAA   2001:4860:4802:38::a
;; Query time: 196 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Sun Feb 09 08:57:16 EST 2020
;; MSG SIZE rcvd: 307
[02/09/20]seed@(10.0.2.4) $
```

Fig1: Dig results after modifying the /resolv.conf.d/head to use 10.0.2.5 as local DNS (Right User)



No.	Time	Source	Destination	Protocol
1	2020-02-09 08:57:14.500000000	10.0.2.4	10.0.2.5	DNS
2	2020-02-09 08:57:14.500000000	10.0.2.4	10.0.2.4	DNS
3	2020-02-09 08:57:21.8347938	PcsCompu_da:46:86	PcsCompu_da:46:86	ARP
4	2020-02-09 08:57:21.8347938	PcsCompu_da:46:86	PcsCompu_da:46:86	ARP
5	2020-02-09 08:57:21.8924197	PcsCompu_da:46:86	PcsCompu_da:46:86	ARP
6	2020-02-09 08:57:21.8924298	PcsCompu_da:46:86	PcsCompu_da:46:86	ARP
7	2020-02-09 08:57:29.3151664	10.0.2.4	10.0.2.4	DHCP

Frame 1: 88 bytes on wire (680 bits), 88 bytes captured (680 bits) on interface 0
 Ethernet II, Src: PcsCompu\_da:46:86 (00:0c:29:21:89:24), Dst: PcsCompu\_da:46:86 (00:0c:29:21:89:24)
 Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.5
 User Datagram Protocol, Src Port: 41584, Dst Port: 53
 Domain Name System (query)
 [Request Info]
 Transaction ID: 0xb0c76
 Flags: 0x0100 Standard query
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 1
 > Question
 > Additional records
 0000: 08 00 27 da 46 86 00 00 27 46 8d 61 00 00 45 00 ..F..E.a.E.
 0010: 00 47 48 69 00 00 40 11 1a 35 9a 00 02 04 00 ..@.0...5...
 0020: 00 05 a2 00 00 35 00 30 18 4d 9c 00 20 00 00 ..p.5.3.Mv...
 0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..www.google...
 0040: 00 03 03 ef 6d 00 00 01 00 01 00 00 29 10 00 00 e.com...
 0050: 00 00 00 00

Fig2: Wireshark showing the DNS request sent to 10.0.2.5 (Right User machine)

**Observation:** After modifying the /etc/resolvconf/resolv.conf.d/head file to use the 10.0.2.5 as nameserver, a dig to [www.google.com](http://www.google.com) went through the new nameserver. The DNS request was observed in Wireshark also.

**Explanation:** The /etc/resolvconf/resolv.conf.d/head was edited and added the new nameserver 10.0.2.5 and then to take the changes effect ran the “sudo resolvconf -u”. To test the changes, ran the dig to [www.google.com](http://www.google.com) and in the response, I can see it came from 10.0.2.5. Also, in the Wireshark a DNS request sent from 10.0.2.4 to 10.0.2.5 as shown in Fig2. With this, the validated the configuration of user machine.

## Task 2: Set up a Local DNS Server

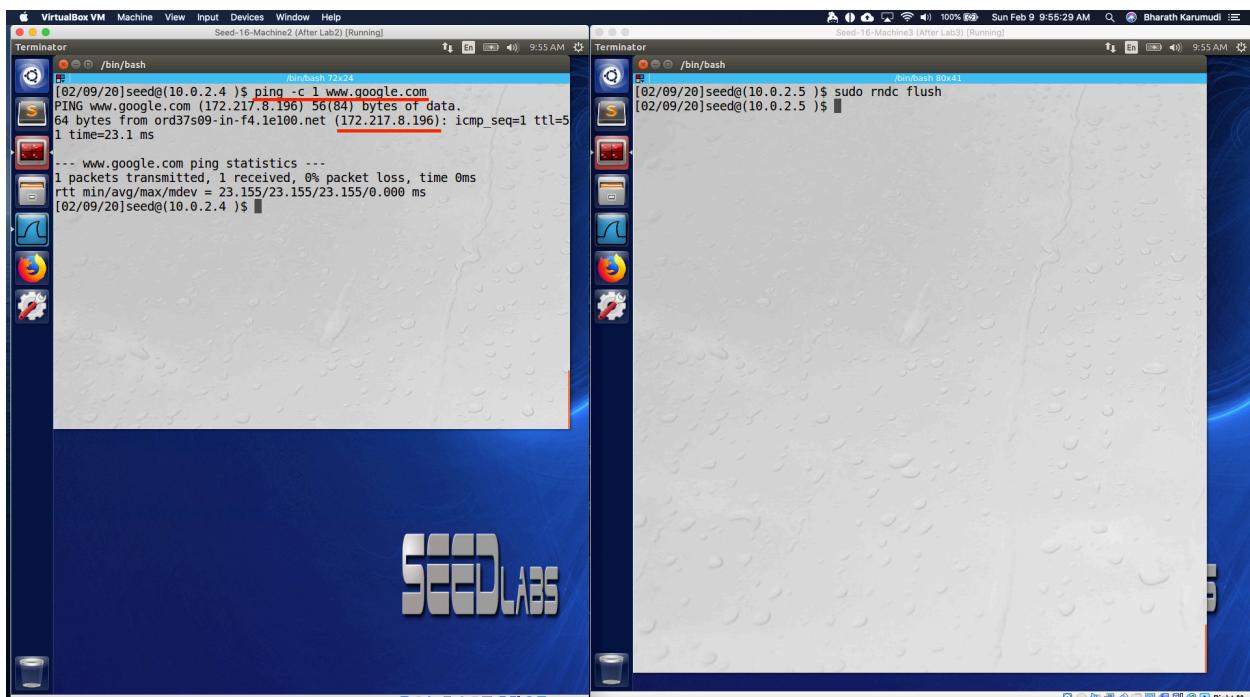


Fig1: Sent a ping to [www.google.com](http://www.google.com) (Left User and Right DNS)

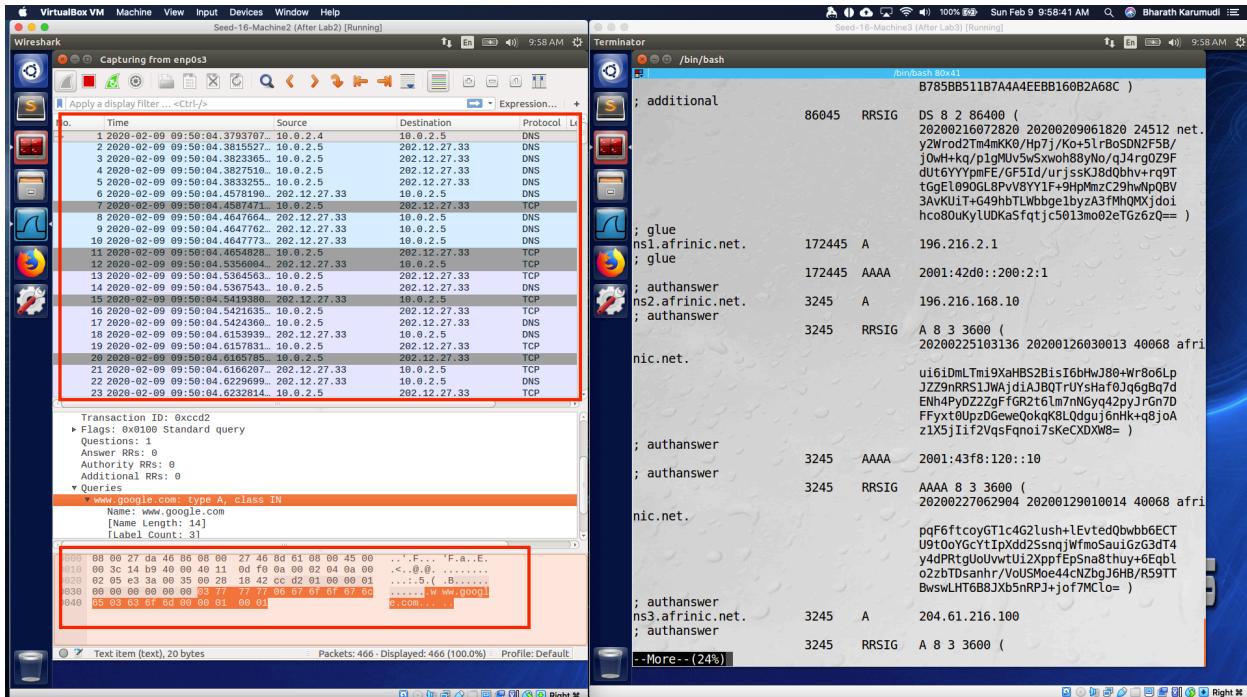


Fig2: DNS requests were sent to 10.0.2.5 (Left User and Right DNS)

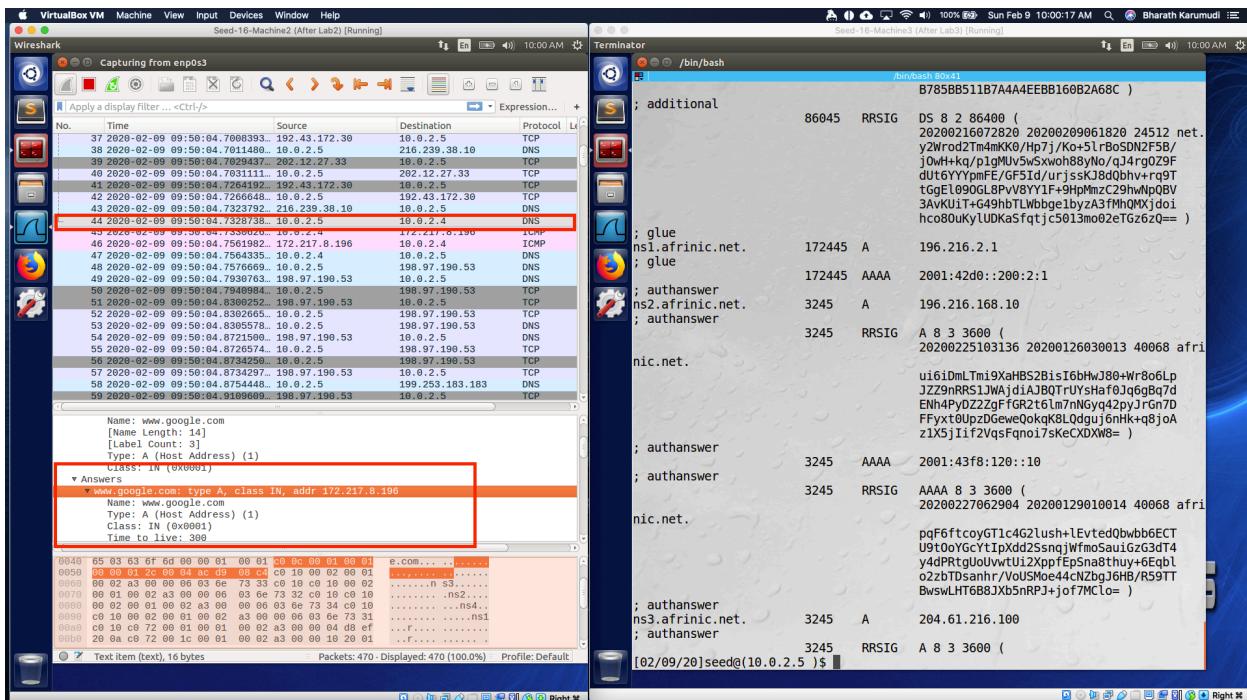


Fig3: The DNS server responded with Answer (Left User and Right DNS)

**Observation:** The Bind9 server configuration was updated to add the dump file path and restarted the Bind9, once the ping request sent to [www.google.com](http://www.google.com) there was multiple DNS requests sent out and the Answer as shown in Fig3.

### Explanation:

1. Added the dump-file path to /etc/bind/named.conf.options file to send the dumpdb file.
2. Verified by creating a dump using the “sudo rndc dumpdb -cache”
3. Verified the DNSSEC and restarted the bind9 server to take the changes effect “sudo service bind9 restart”.
4. Sent a ping to www.google.com; the local DNS server and the when checked in Wireshark the local DNS sent the request first to root server and the root server responded with TLD server and the TLD gave the NS of Google and the local DNS sent another request to Google DNS.
5. Finally, the Answer came to user machine 10.0.2.4 from 10.0.2.5 (DNS server).

### Task 3: Host a Zone in the Local DNS Server

The screenshot shows two terminal windows side-by-side. The left terminal window is running on a user machine (10.0.2.4) and displays the output of a 'dig' command. The right terminal window is running on a DNS server (10.0.2.5) and displays the contents of the 'example.com.db' zone file.

**User Machine Terminal Output (Left):**

```
[02/09/20]seed@(10.0.2.4)$ dig www.example.com
; <>> DIG 9.10.3-P4-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>>HEADER<- opcode: QUERY, status: NOERROR, id: 48955
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com. IN A
;; ANSWER SECTION:
www.example.com. 259200 IN A 192.168.0.101
;; AUTHORITY SECTION:
example.com. 259200 IN NS ns.example.com.
;; ADDITIONAL SECTION:
ns.example.com. 259200 IN A 192.168.0.10
;; Query time: 0 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Sun Feb 09 10:34:49 EST 2020
;; MSG SIZE rcvd: 93
[02/09/20]seed@(10.0.2.4)$
```

**DNS Server Terminal Output (Right):**

```
[02/09/20]seed@(10.0.2.5)$ cat example.com.db
$TTL 3D ; default expiration time of all resource records without
          ; their own TTL
@ IN SOA ns.example.com. admin.example.com. (
        1 ; Serial
        8H ; Refresh
        2H ; Retry
        4W ; Expire
        1D ) ; Minimum
@ IN NS ns.example.com. ;Address of name
server
@ IN MX 10 mail.example.com. ;Primary Mail Ex
changer
www IN A 192.168.0.101 ;Address of www.example.
.com
mail IN A 192.168.0.102 ;Address of mail.example
.com
ns IN A 192.168.0.10 ;Address of ns.e
xample.com
*.example.com. in A 192.168.0.100 ;Address for other URL in the example.co
m domain
[02/09/20]seed@(10.0.2.5)$ cat 192.168.0.db
$TTL 3D
@ IN SOA ns.example.com. admin.example.com. (
        1
        8H
        2H
        4W
        1D )
@ IN NS ns.example.com.
101 IN PTR www.example.com.
102 IN PTR mail.example.com.
108 IN PTR ns.example.com.
[02/09/20]seed@(10.0.2.5)$
```

Fig1: Created a Zone file for example.com (Left User and Right DNS)

The screenshot shows two terminal windows side-by-side. The left terminal window, titled 'Terminator /bin/bash', displays the output of the 'dig' command for 'www.example.com'. The right terminal window, also titled 'Terminator /bin/bash', shows the contents of the '/etc/bind/named.conf' file and the forward and reverse lookup zone files ('example.com.db' and '192.168.0.db'). Both terminals are running on a Linux desktop environment.

```
[02/09/20]seed@(10.0.2.4)$ dig www.example.com
; <>> DiG 9.10.3-P4-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; -->HEADER-- opcode: QUERY, status: NOERROR, id: 21987
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com. IN A
;; ANSWER SECTION:
www.example.com. 259200 IN A 192.168.0.101
;; AUTHORITY SECTION:
example.com. 259200 IN NS ns.example.com.
;; ADDITIONAL SECTION:
ns.example.com. 259200 IN A 192.168.0.10
;; Query time: 0 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Sun Feb 09 10:28:03 EST 2020
;; MSG SIZE rcvd: 93
[02/09/20]seed@(10.0.2.4)$

[02/09/20]seed@(10.0.2.5)$ ls -lrt
total 60
-rw-r--r-- 1 root root 1317 Jun 29 2017 zones.rfc1918
-rw-r--r-- 1 root bind 165 Jun 29 2017 named.conf.local
-rw-r--r-- 1 root bind 490 Jun 29 2017 named.conf.default-zones
-rw-r--r-- 1 root root 3171 Jun 29 2017 db.root
-rw-r--r-- 1 root root 270 Jun 29 2017 db.local
-rw-r--r-- 1 root root 353 Jun 29 2017 db.empty
-rw-r--r-- 1 root root 237 Jun 29 2017 db.255
-rw-r--r-- 1 root root 271 Jun 29 2017 db.127
-rw-r--r-- 1 root root 237 Jun 29 2017 db.0
-rw-r--r-- 1 root root 2389 Jun 29 2017 bind.keys
-rw-r--r-- 1 root bind 77 Jul 25 2017 rndc.key
-rw-r--r-- 1 bind bind 979 Feb 9 09:48 named.conf.options
-rw-r--r-- 1 root bind 620 Feb 9 10:12 named.conf
-rw-r--r-- 1 root bind 194 Feb 9 10:24 192.168.0.db
-rw-r--r-- 1 root root 578 Feb 9 10:24 example.com.db
[02/09/20]seed@(10.0.2.5)$ sudo service bind9 restart
[02/09/20]seed@(10.0.2.5)$ sudo rndc flush
[02/09/20]seed@(10.0.2.5)$ sudo chown root:root bind example.com.db
[02/09/20]seed@(10.0.2.5)$ sudo service bind9 restart
[02/09/20]seed@(10.0.2.5)$ sudo rndc flush
[02/09/20]seed@(10.0.2.5)$ vi example.com.db
[02/09/20]seed@(10.0.2.5)$ sudo !
sudo vi example.com.db
[02/09/20]seed@(10.0.2.5)$ sudo service bind9 restart
[02/09/20]seed@(10.0.2.5)$
```

Fig2: Dig to example.com (Left User and Right DNS)

**Observation:** Created the Zones for forward and reverse lookup and also zone files for [www.example.com](http://www.example.com). Once the files are created, restarted the Bind server and from User machine made a dig to [www.example.com](http://www.example.com) and got the response from the local DNS server with the IP (192.168.0.101) that was set.

#### Explanation:

1. Added the zones for example.com to `/etc/bind/named.conf`
2. Created the forward lookup zone file for example.net domain with the entries of name `"/etc/bind/example.com.db"`. This file will be used for DNS resolution from name to IP address.
3. Created the reverse lookup zone file for the domain example.com of file `/etc/bind/192.168.0.db` and this file will be used for DNS resolution from IP address to name.
4. Once these files are created, restarted the Bind9 server with "`sudo service bind9 restart`"
5. From the user machine, made a dig to example.com and the local DNS which is acting as an Authoritative nameserver for example.com responded with an Answer as shown in Fig2.

## Task 4: Modifying the Host File

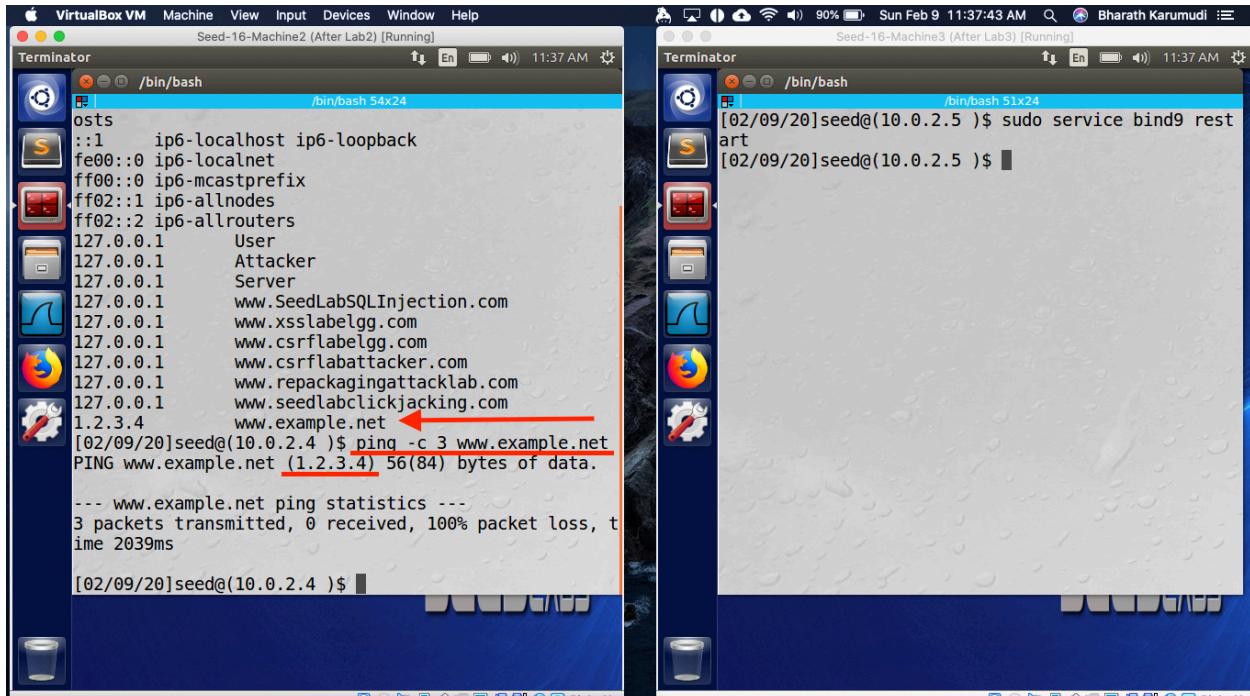


Fig1: Ping to example.net after updating the hosts file (Left User and Right DNS)

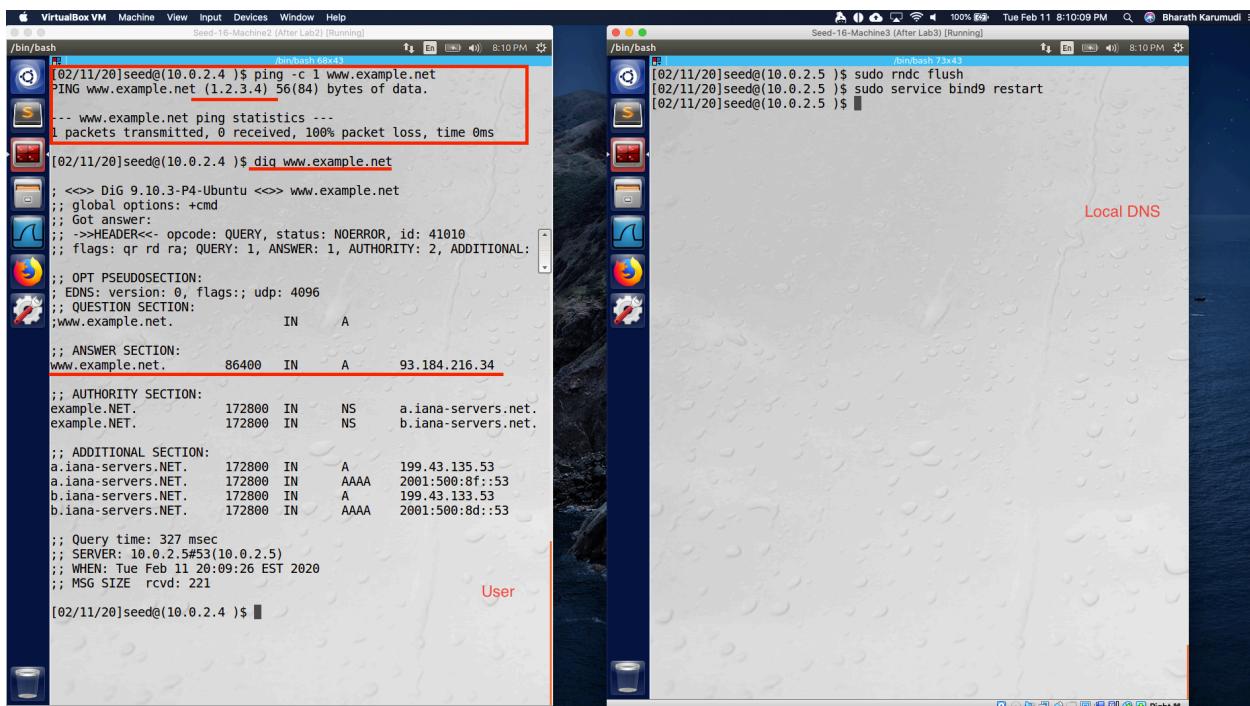


Fig2: Difference between ping and dig resolutions (Left User and Right DNS)

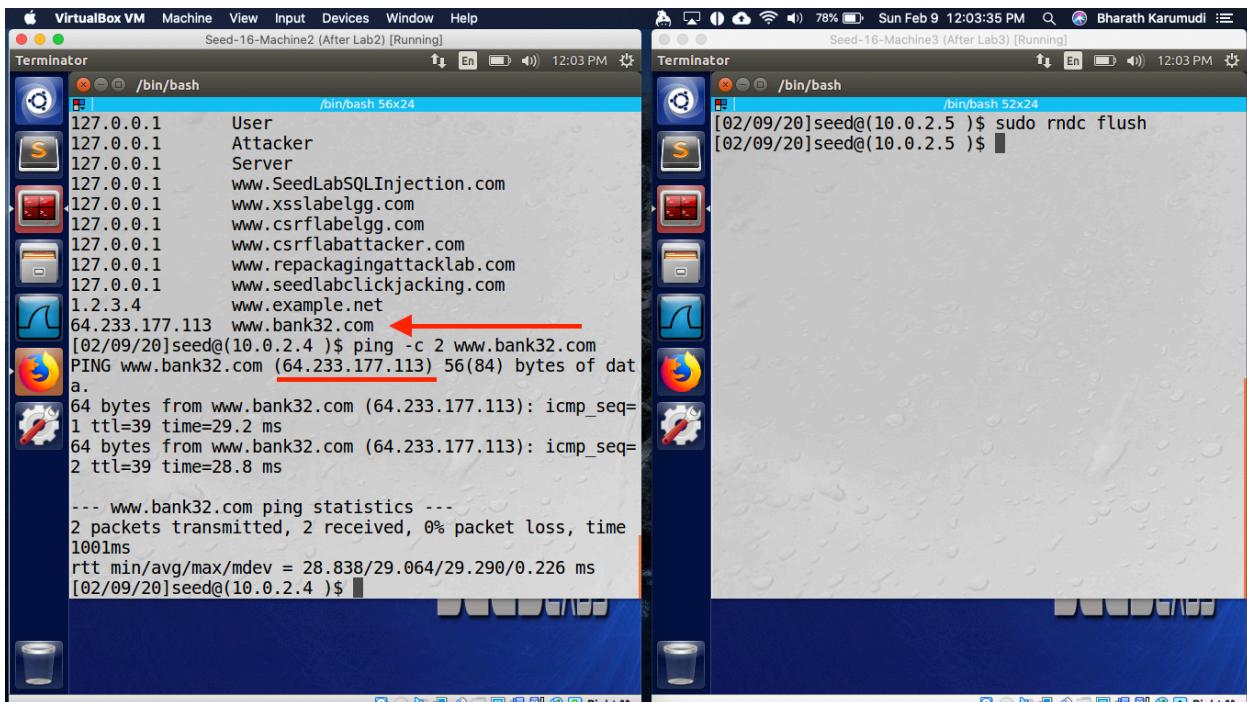


Fig3: Ping to [www.bank32.com](http://www.bank32.com) goes to google.com (Left User and Right DNS)

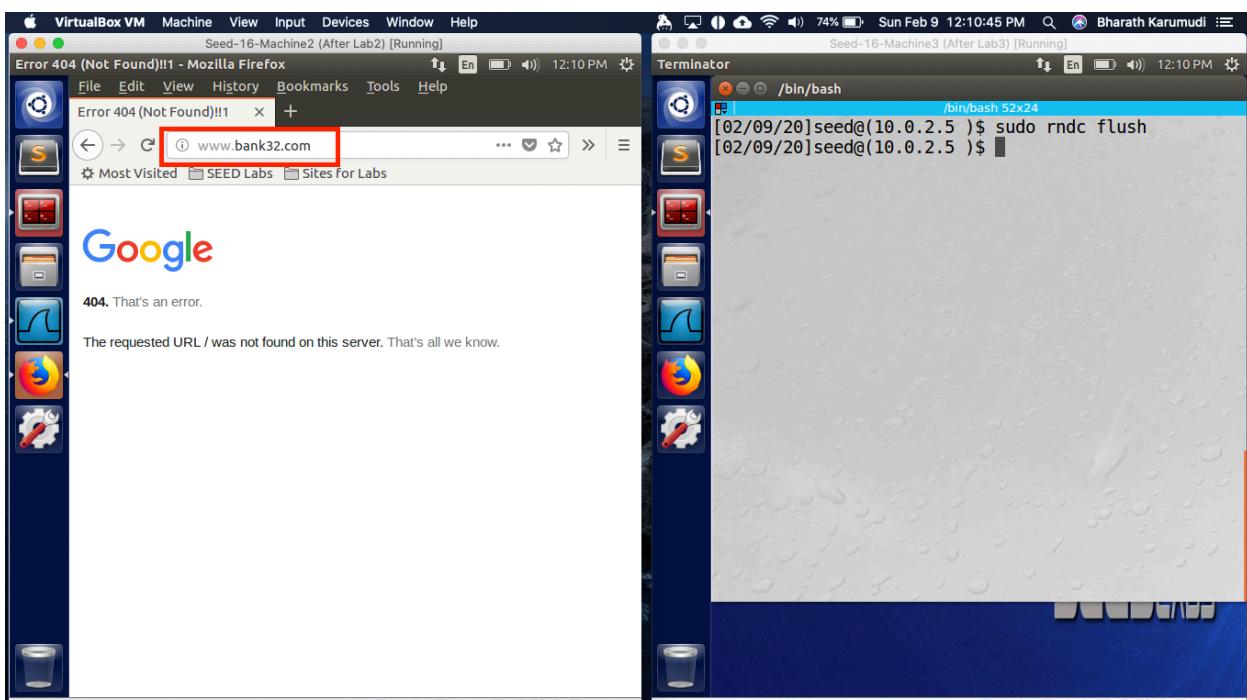


Fig4: Redirected to Google.com (Left User and Right DNS)

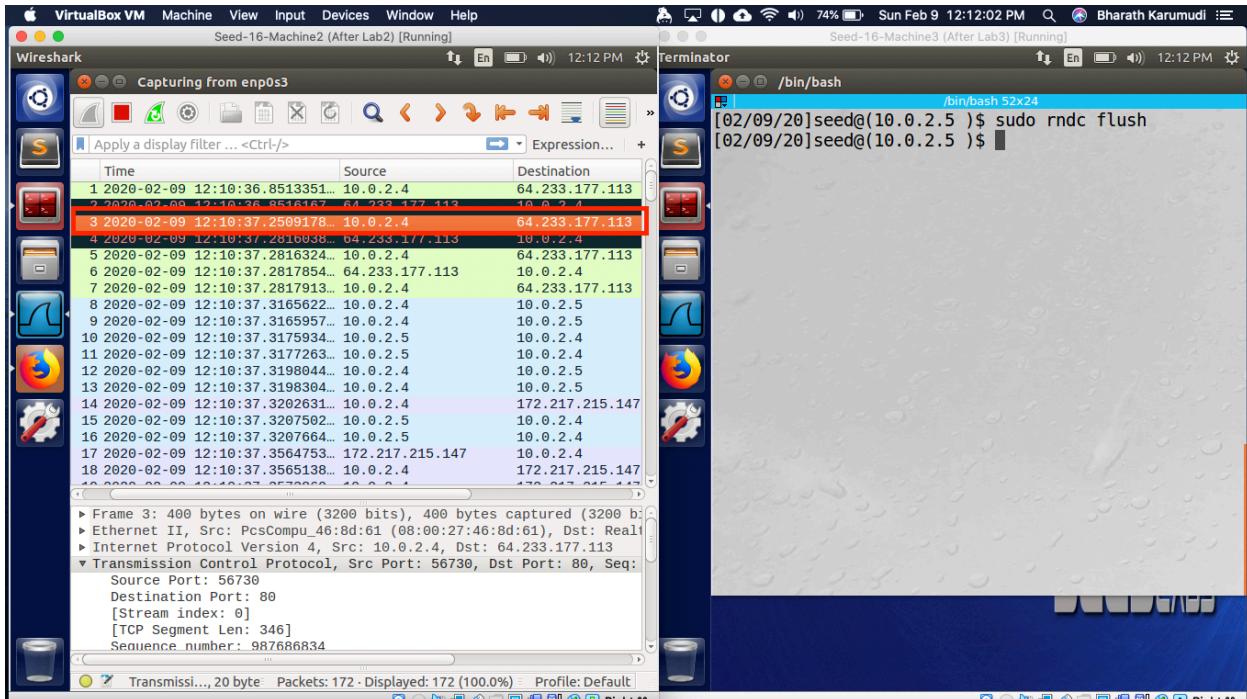


Fig5: Observation in Wireshark (Left User and Right DNS)

**Observation:** Modified the hosts file and added the entries for example.net pointing to 1.2.3.4 IP and for bank32.com added 64.233.177.113. Once added the entries, the ping to example.net sent to 1.2.3.4 and for bank32.com in browser it is redirected to Google.

**Explanation:** The hosts file in the Operating System has the static entries for a domain and its IP address. When a request sent out, the user machine will first consider the entries in hosts file rather than asking local DNS server. But in case of dig it is an exception.

In this case,

1. The entry was added to example.net and pointed to 1.2.3.4 as shown in Fig1; when a ping sent out from user machine, it tried to ping the 1.2.3.4 IP address.
2. Sent a dig to example.net domain from the User machine and it got the answer of actual example.net and ignored the hosts file entry (as shown in Fig 2).
3. Another entry was added for bank32.com and pointed to Google IP (64.233.177.113); from the browser when accessed the bank32.com it went to Google site and can also see in Wireshark as shown in Fig 5.

## Task 5: Directly Spoofing Response to User

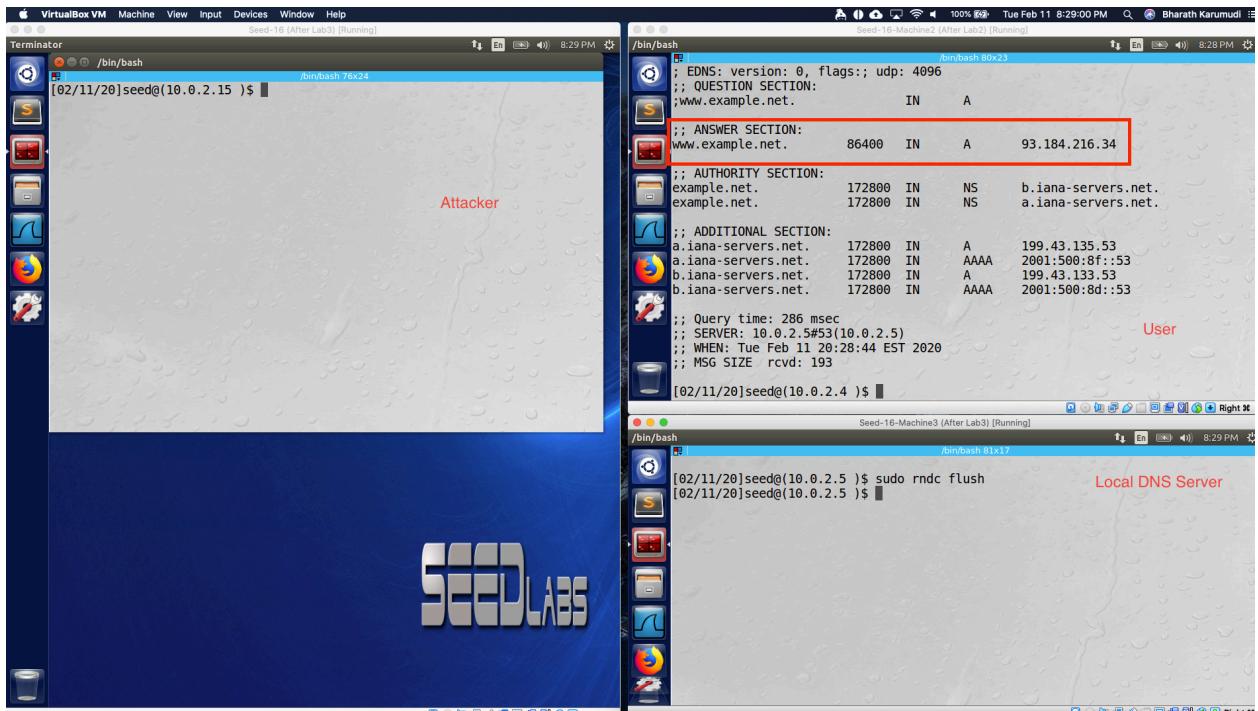


Fig1: Before Attack, user got the proper response for [www.example.net](http://www.example.net) with dig.

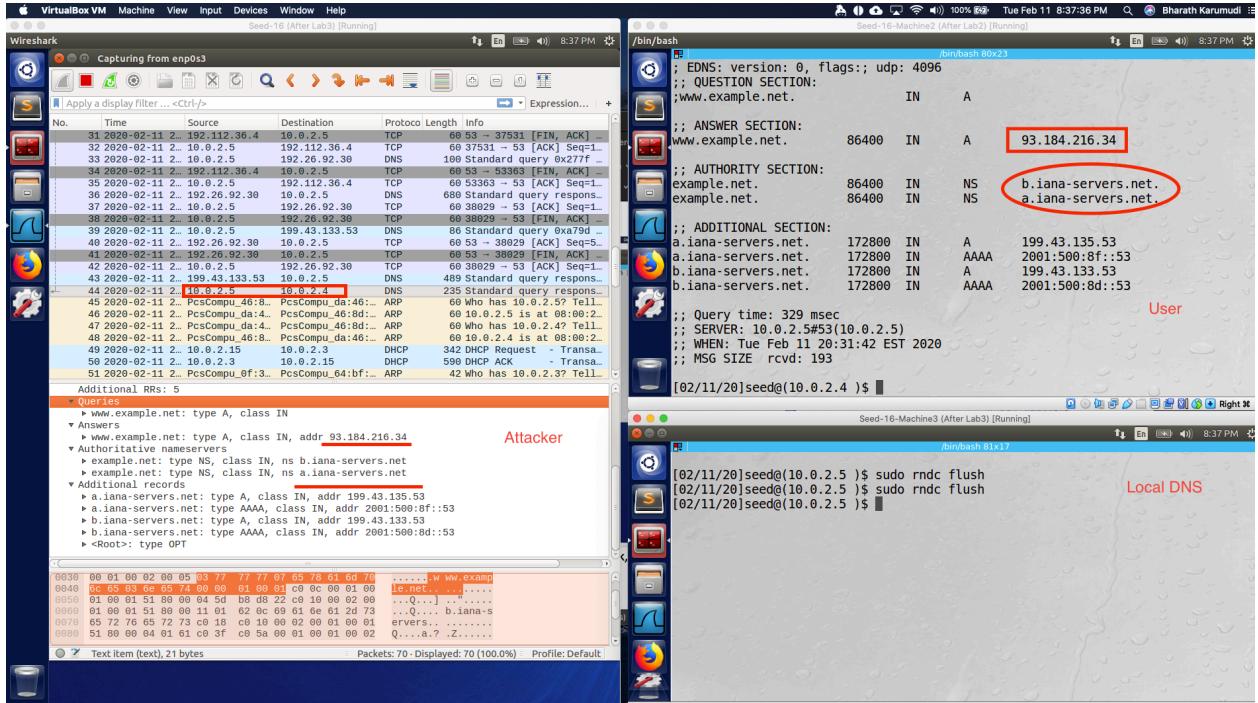


Fig2: Attacker sniffing the DNS traffic from 10.0.2.4 (User) and capturing the required data

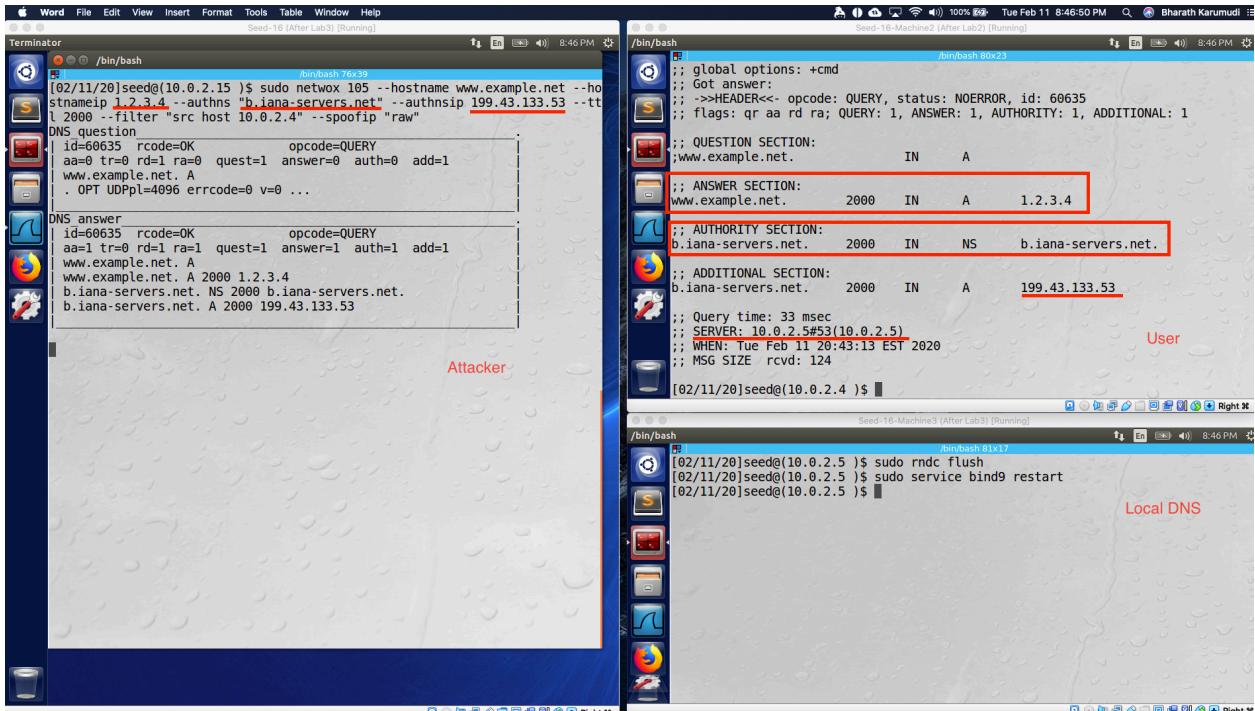


Fig3: Attacker Sniffed and Spoofed a DNS reply successfully with answer IP as 1.2.3.4

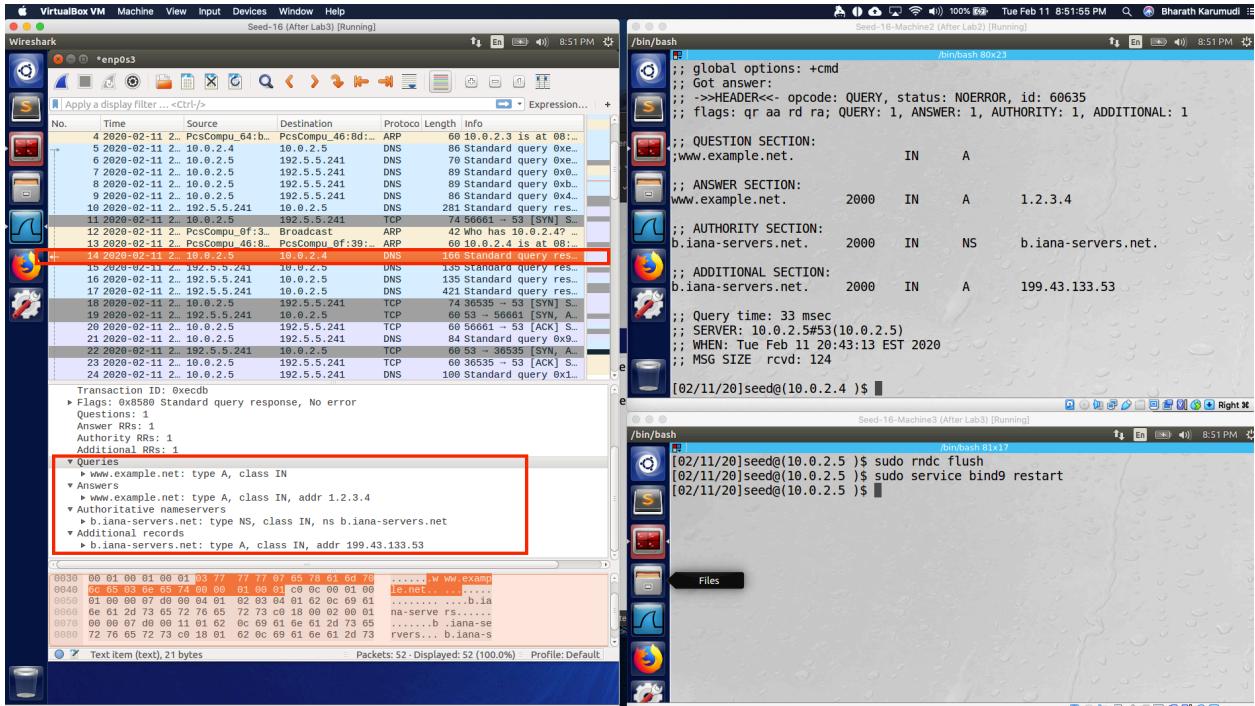


Fig4: Observation in Wireshark in Attacker Machine for the Spoofed DNS Answer

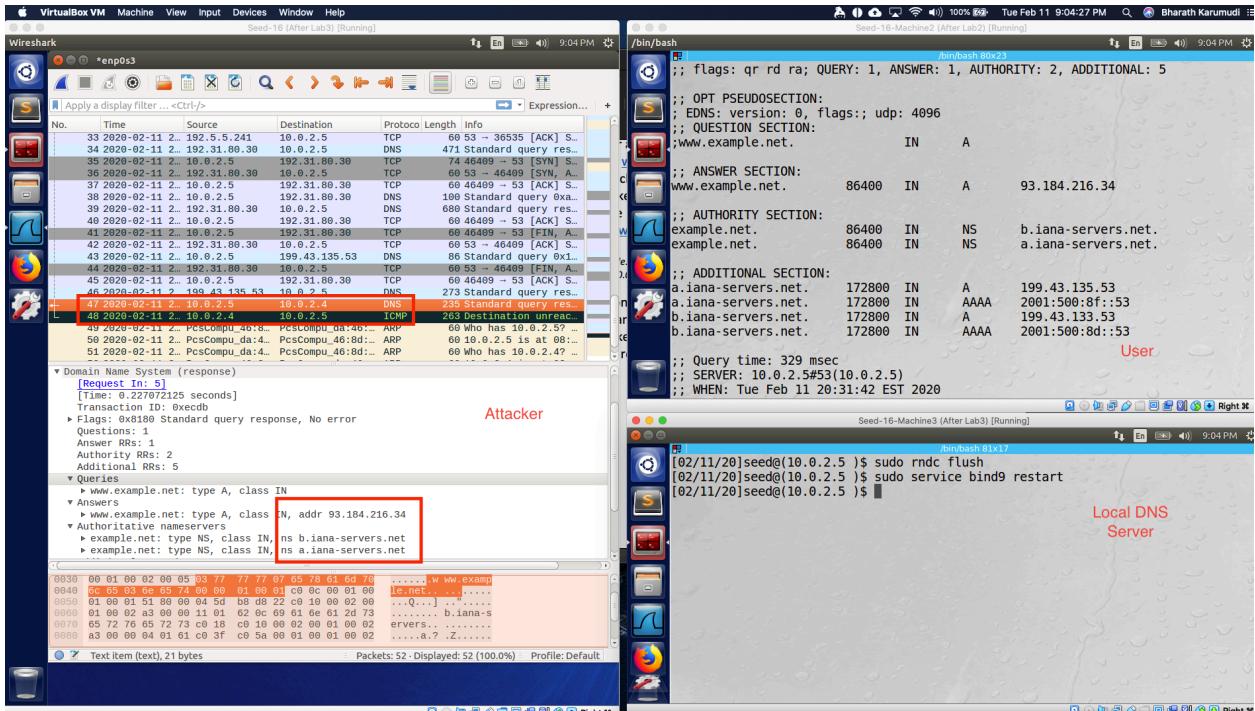


Fig5: Observation in Wireshark for real response

**Observation:** When user sent a DNS Query for [www.example.net](http://www.example.net) for the first time, the user got the correct answer from the local DNS server which is “93.184.216.34”. For the next time, when the attacker done the Snooping, the user got the fake answer for the [www.example.net](http://www.example.net) as “1.2.3.4”.

### Explanation:

To demonstrate:

1. First, set up the local DNS server and removed the cache and restarted the Bind9.
2. Then User sent a DNS Query for [www.example.net](http://www.example.net) and the local DNS responded with an Answer as “93.184.216.34”, which is correct one.
3. During the first query, the attacker is Sniffing the local network traffic and captured the required information like Source IP, Authoritative Name server details to perform the DNS Snooping attack for the [www.example.net](http://www.example.net)

```
sudo netwox 105 --hostname www.example.net --hostnameip 1.2.3.4 --authns "b.iana-servers.net" --authnsip 199.43.133.53 --ttl 2000 --filter "src host 10.0.2.4" --spoofip "raw"
```

4. As an attacker framed the response using netwox and waiting for the DNS query to initiate from the User machine and once the user initiated the DNS query, the attacker sent the spoofed DNS reply packet to the user and the user machine accepted as shown in Fig 3 and Fig 4.
5. The real DNS response was also received by the local DNS and sent to User machine, but it was rejected as shown in Fig5.

## Task 6: DNS Cache Poisoning Attack

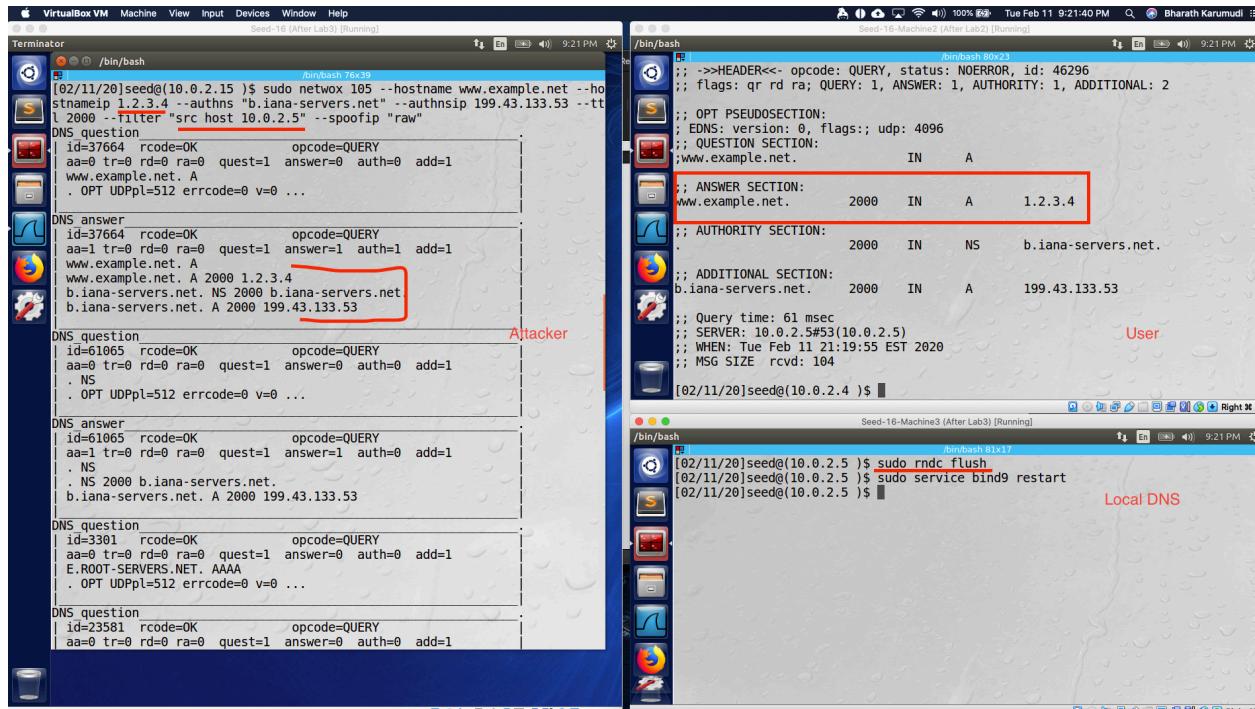


Fig1: Attacker spoofing the DNS Server and Poisoned the DNS Cache

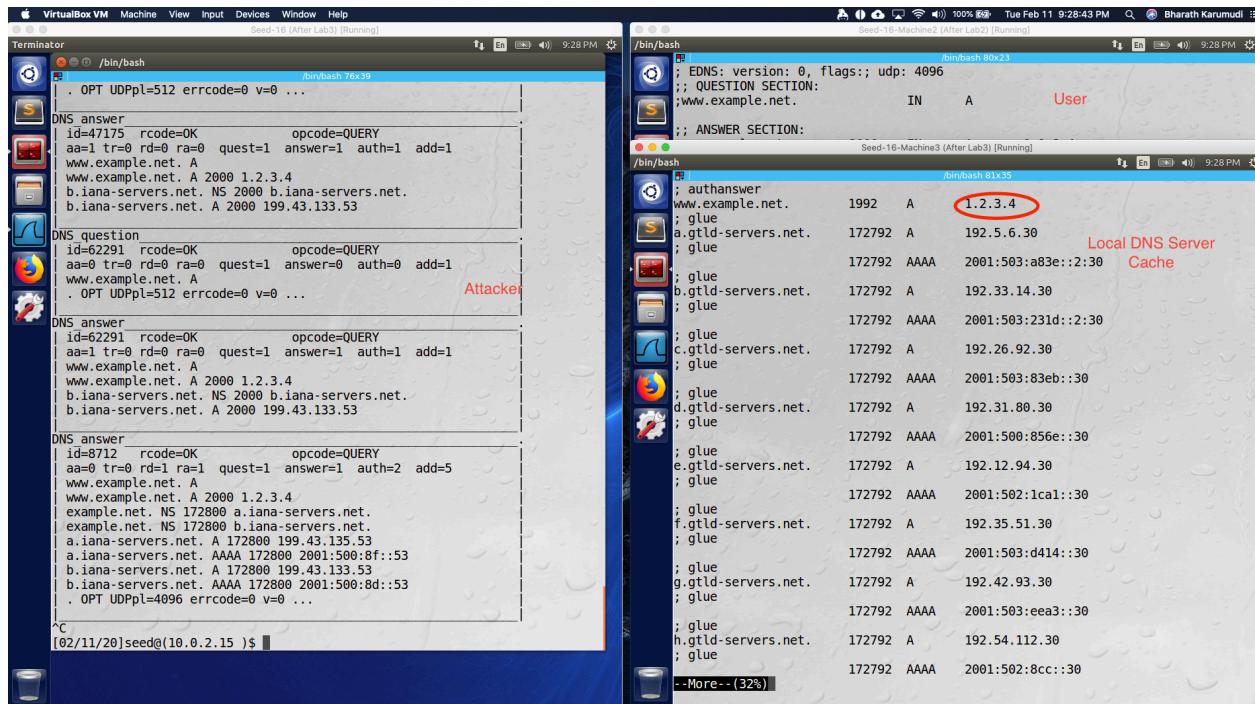


Fig2: Local DNS Server Cache was Poisoned

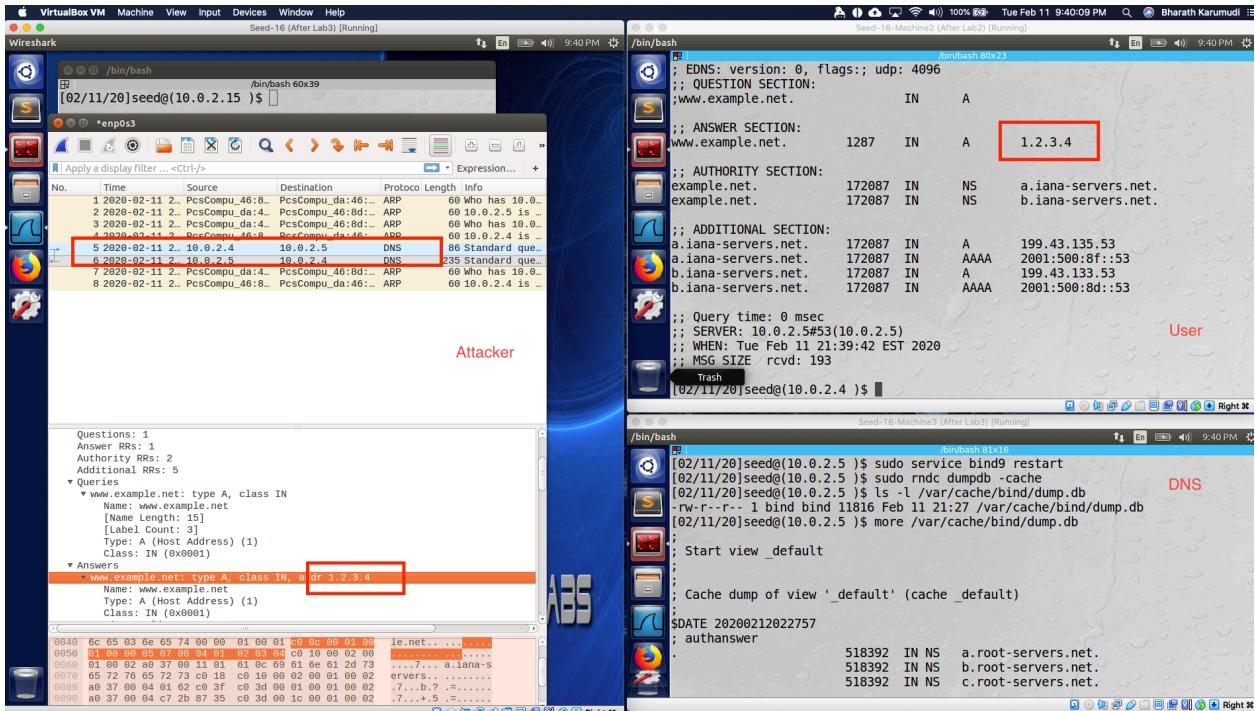


Fig3: After the attack, due to Cache Poisoned the user getting the fake answer though attack was stopped and Wireshark observation

**Observation:** As an attacker poisoned the DNS cache and after the attack the user still got the fake DNS response because of answer saved in DNS server cache.

**Explanation:** When the user sends out the DNS query for [www.example.net](http://www.example.net); as an attacker sent a spoofed DNS reply to the local DNS server. The DNS server accepted the response and provided the Answer to the user as shown in Fig1. Also, the DNS server cached (Fig 2) the Answer provided and next time when user asked for same domain, the DNS server rather than asking the root servers, it checked in its cache and provided the fake answer that attacker provided initially, as shown in Fig 3. This is called DNS Cache Poisoning.

```
sudo netwox 105 --hostname www.example.net --hostnameip 1.2.3.4 --authns "b.iana-servers.net" --authnsip 199.43.133.53 --ttl 2000 --filter "src host 10.0.2.5" --spoofip "raw"
```

Once the DNS cache was poisoned until the TTL expires, the DNS server provides the answers from cache.