COLLEGE OF ENGINEERING

NETWORK SECURITY(EE/CMPE-209)

FALL-2017

REPORT
(Remote DNS Attack)

Submitted to:

Prof. Juzi Zhao

Submitted by:

Rahul Rao Kodati (011419148)

DNS (Domain Name System)
(2 points)

Table of Contents:

Introduction:

The **Domain Name System** (**DNS**) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System is an essential component of the functionality on the Internet, that has been in use since 1985.

The Domain Name System delegates the responsibility of assigning domain names and mapping those names to Internet resources by designating authoritative name servers for each domain. Network administrators may delegate authority over sub-domains of their allocated namespace to other name servers. This mechanism provides distributed and fault tolerant service and was designed to avoid a single large central database.

Lab Setup :

We set up the lab environment like what is shown below. To simplify the lab environment, we used our computer, DNS server, and attacker's computer be on one physical machine, but using different virtual machines. Our configuration is based on Ubuntu, which is the operating system we use in our pre-built virtual machine.

As you can see from Figure 1, we set up the DNS server, the user machine and the attacker machine in the same LAN. We assume that the DNS Server's IP is 192.168.0.152 and the attacker machine's IP is 192.168.0.148.

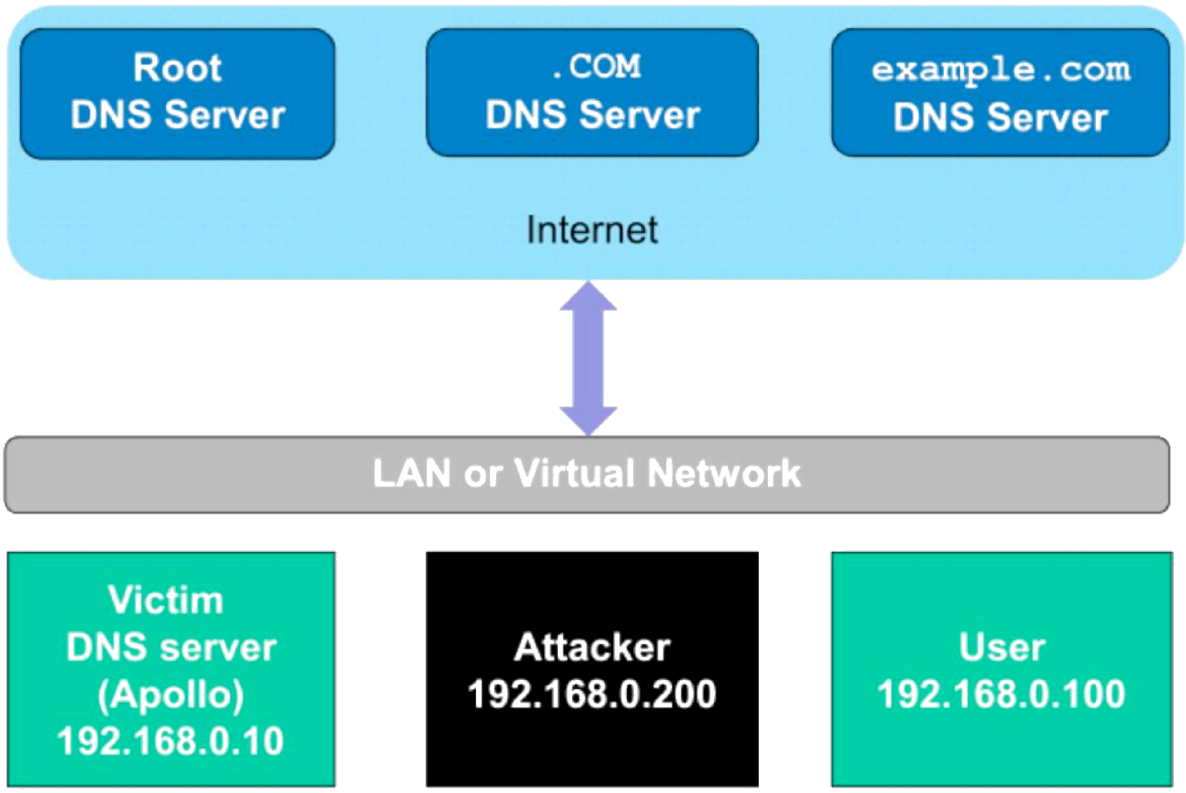


Fig : Lab Environment Setup

1.Configure the Local DNS server Apollo

Step 1:
Installed the BIND 9 DNS server using below commands

```
# sudo apt-get install bind9
```

Step 2:
Created the named.conf.options file. The DNS server needs to read a configuration file /etc/bind/named.conf to start. This configuration file usually includes an option file, which is called /etc/bind/named.conf.options.
we added the following contents to the option file:

```
options {
        dump-file "/var/cache/bind/dump.db";
        };
```

The file /var/cache/bind/dump.db is used to dump DNS server's cache.
Here are some useful commands:

```
% sudo rndc flush // Flush the DNS cache
% sudo rndc dumpdb -cache // Dump the cache to dump.db
```

Step 3: Restart DNS server

We used the following command to restart DNS Server:

```
% sudo service bind9 restart
```

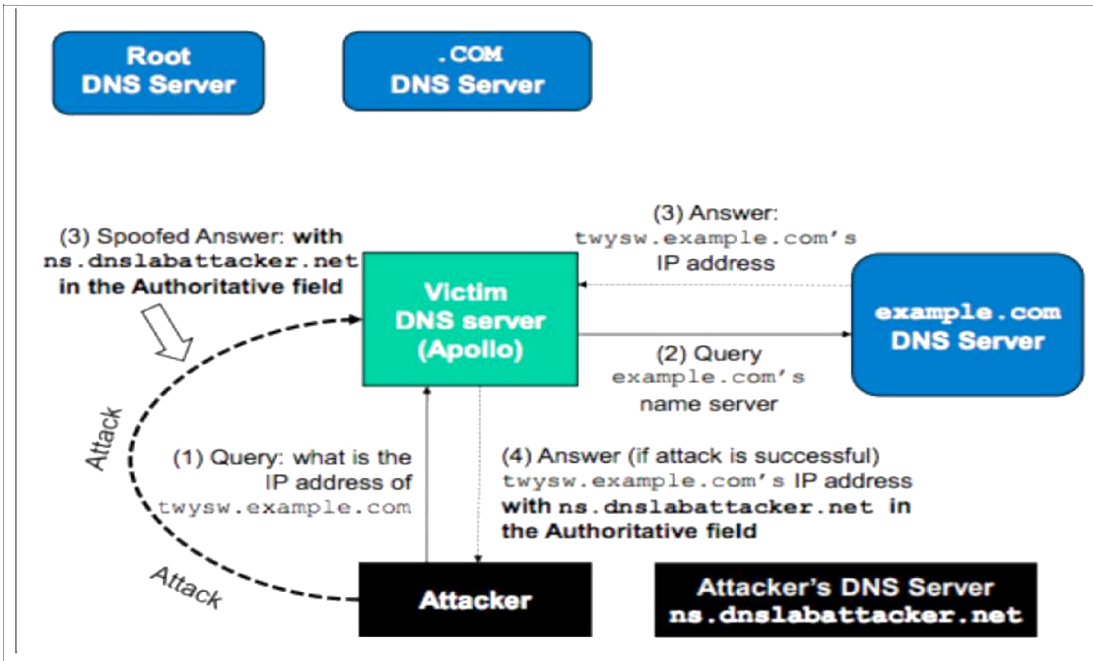
II. Configure the User Machine

Click "System Settings" -> "Network", Click "Options" in "Wired" Tab, Select "IPv4 Settings" -> "Method" ->"Automatic(DHCP) Addresses Only" and update only "DNS Servers" entry with IP address of BIND DNS Server.

Click the "Network Icon" on the top right corner and Select "Auto eth0". This will refresh the wired network connection and updates the changes. We restarted the Ubuntu machine for the modified settings to take effect.

III. Task 1: Remote Cache Poisoning

In this task, the attacker sends a DNS query request to the victim DNS server (Apollo), triggering a DNS query from Apollo. The query may go through one of the root DNS servers, the .COM DNS server, and the final result will come back from example.com's DNS server. In case that example.com's name server information is already cached by Apollo, the query will not go through the root or the .COM server. In this lab, the situation depicted is more common, so we will use this figure as the basis to describe the attack mechanism. While Apollo waits for the DNS reply from example.com's name server, the attacker can send forged replies to Apollo, pretending that the replies are from example.com's name server. If the forged replies arrive first, it will be accepted by Apollo. The attack will be successful.



Before doing task-1 we checked the example.com cache in the DNS Server by using dig command.
The following screenshot is the output of the DNS server cache of example.com.

```
                                        b/lRjhdmJ1EBT7bUue0Oblj18Xc/bfF/44P0
                                        v/74dRujGASH53bqbhAgsEm81cw= )
; authauthority
aaaba.example.com.      3492    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
ababa.example.com.      3494    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
bbaba.example.com.      3496    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
bbbba.example.com.      3498    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
bbcba.example.com.      3499    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
bbdba.example.com.      3501    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
cbdba.example.com.      3503    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; example.com. RRSIG NSEC ...
; example.com. NSEC www.example.com. A NS SOA TXT AAAA RRSIG NSEC DNSKEY
; authauthority
dbdba.example.com.      3505    \-ANY   ;-$NXDOMAIN
; example.com. SOA sns.dns.icann.org. noc.dns.icann.org. 2017102403 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
```

Attack Configuration:

We need to make the following configuration for this task:

1. Configure the Attack Machine. We configured the attack machine, so it uses the targeted DNS server (i.e., Apollo) as its default DNS server.

Make sure that the network configuration for this VM is "NAT Network".

2. Source Ports. Some DNS servers now randomize the source port number in the DNS queries; this makes the attacks much more difficult. Unfortunately, many DNS servers still use predictable source port number. For the sake of simplicity in this lab, we assume that the source port number is a fixed number. We can set the source port for all DNS queries to 33333. This can be done by adding the following option to the file /etc/bind/named.conf.options on Apollo:

query-source port 33333

3. DNSSEC. Most DNS servers now adopt a protection scheme called ”DNSSEC”, which is designed to defeat the DNS cache poisoning attack.

If you do not turn it off, your attack would be extremely difficult, if possible at all. In this lab, we will turn it off. This can be done by changing the file /etc/bind/named.conf.options on Apollo. Please find the line "dnssec-validation auto", comment it out, and then add a new line. See the following:

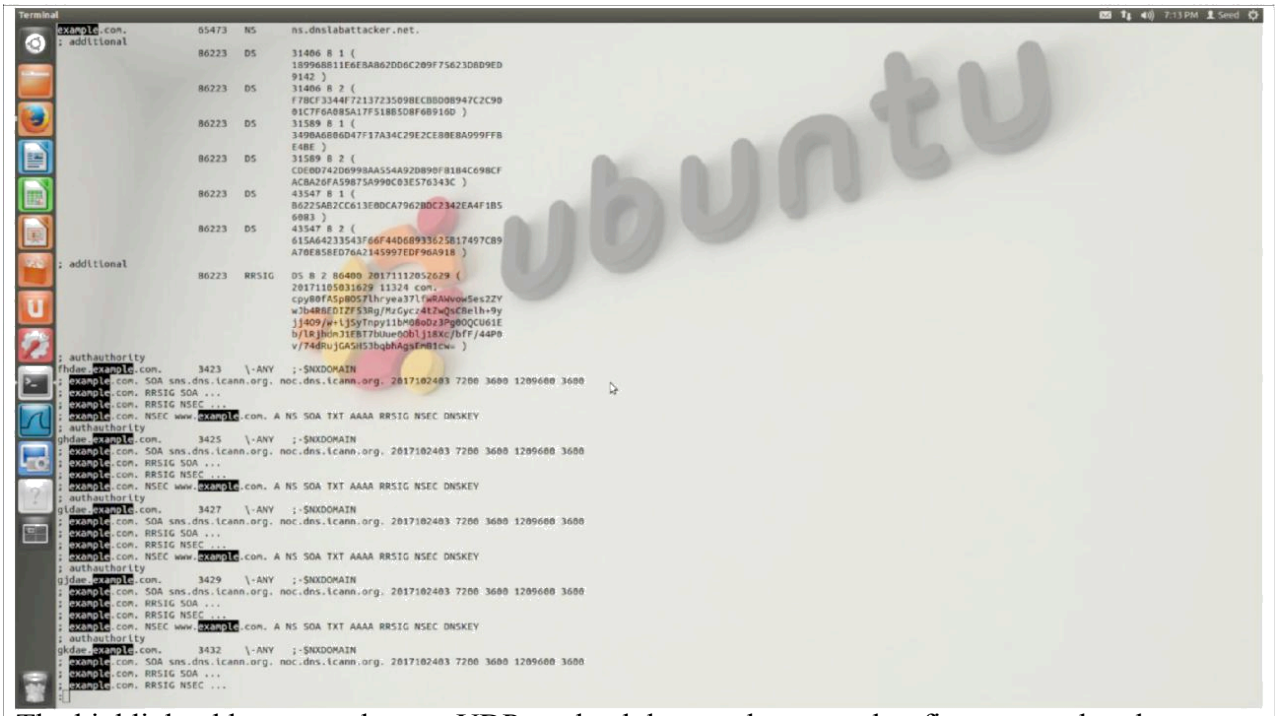
//dnssec-validation auto; dnssec-
enable no; 4. Flush the Cache.

Flush Apollo’s DNS cache, and restart its DNS server.

Forge DNS Response Packets.

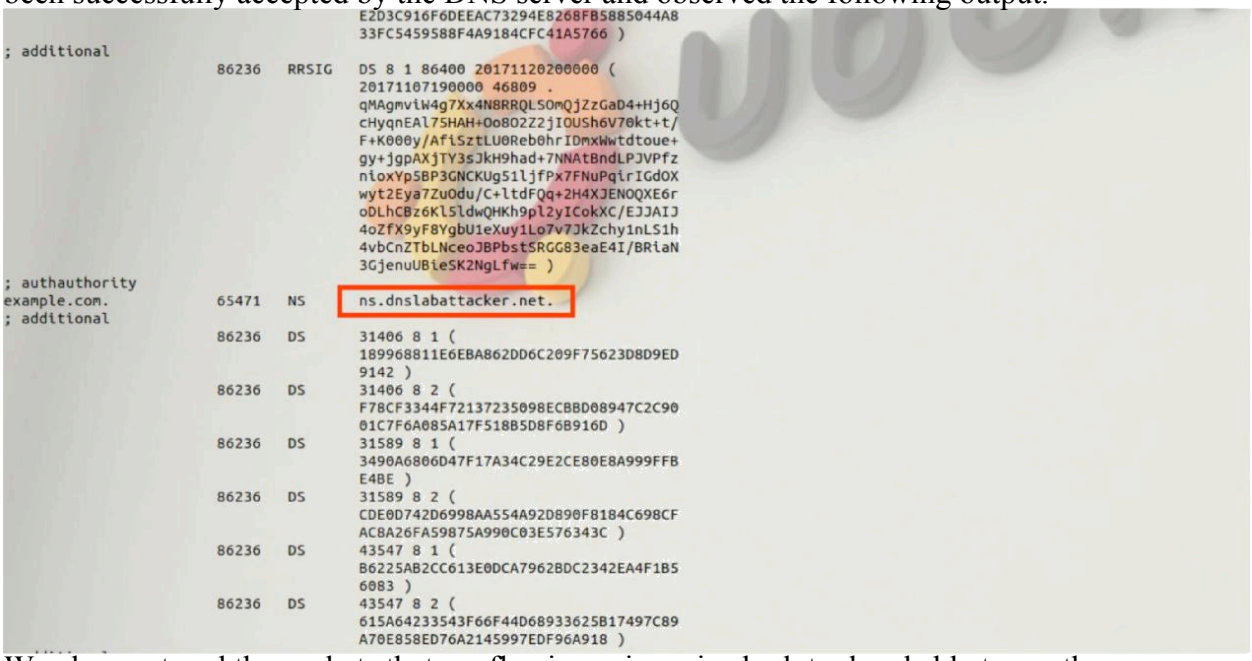
In order to complete the attack, the attacker first needs to send DNS queries to Apollo for some random hostnames in the example.com domain. Right after each query is sent out, the attacker needs to forge a large number of DNS response packets in a very short time window, hoping that one of them has the correct transaction ID and it reaches the target before the authentic response does. It is better to write C code to achieve this. The seed labs provided a sample code called udp.c. This program can send a large number of DNS packets.

DNS response packet details: it is not easy to construct a correct DNS response packet.

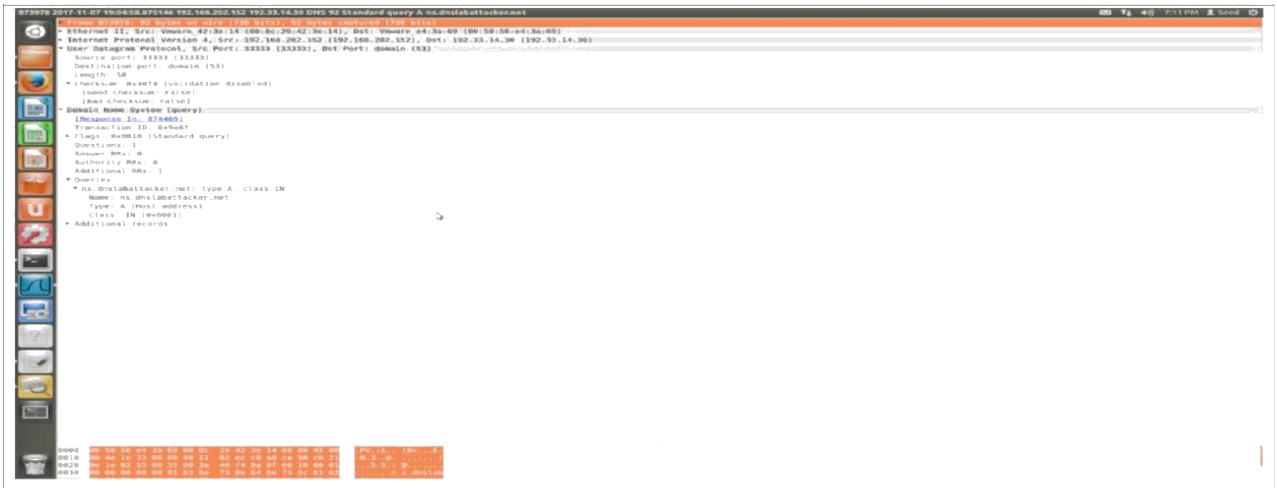


The highlighted bytes are the raw UDP payload data, and you need to figure out what they are. The details about how each byte works are explained clearly. There are several techniques used in the response packet, such as the string pointer offset to shorten the packet length.

We Checked the dump.db file to see whether your spoofed DNS response has been successfully accepted by the DNS server and observed the following output.

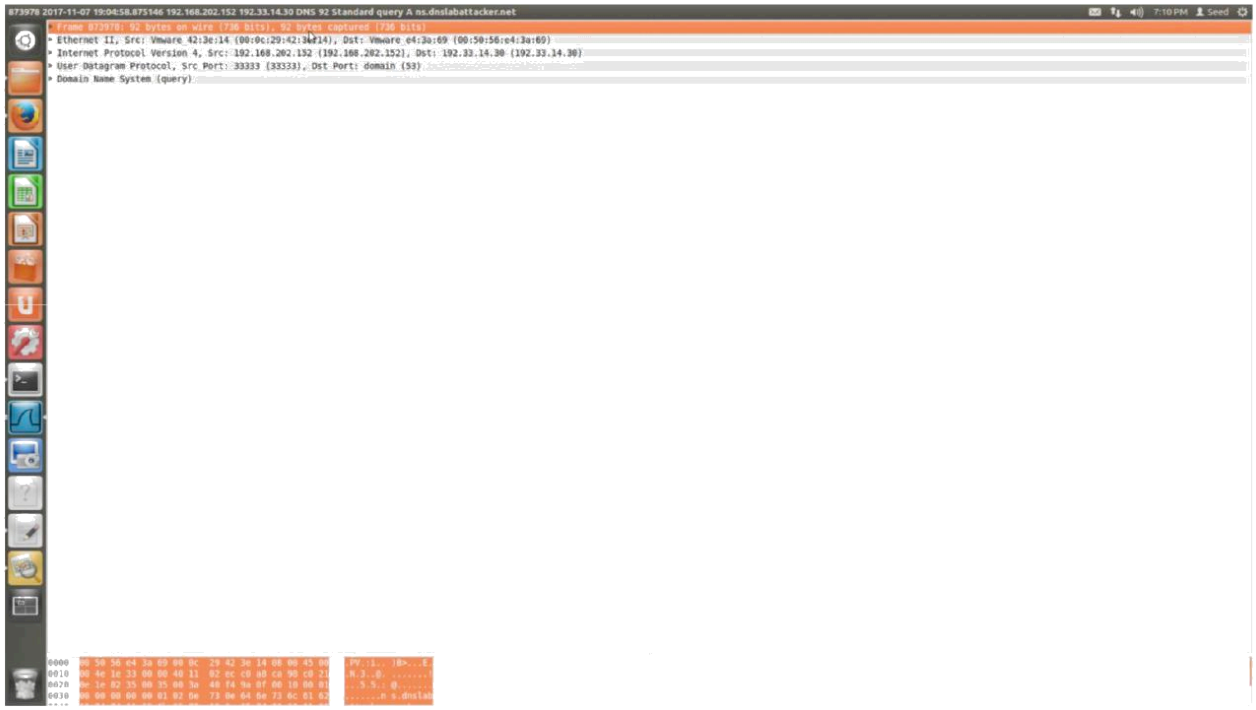


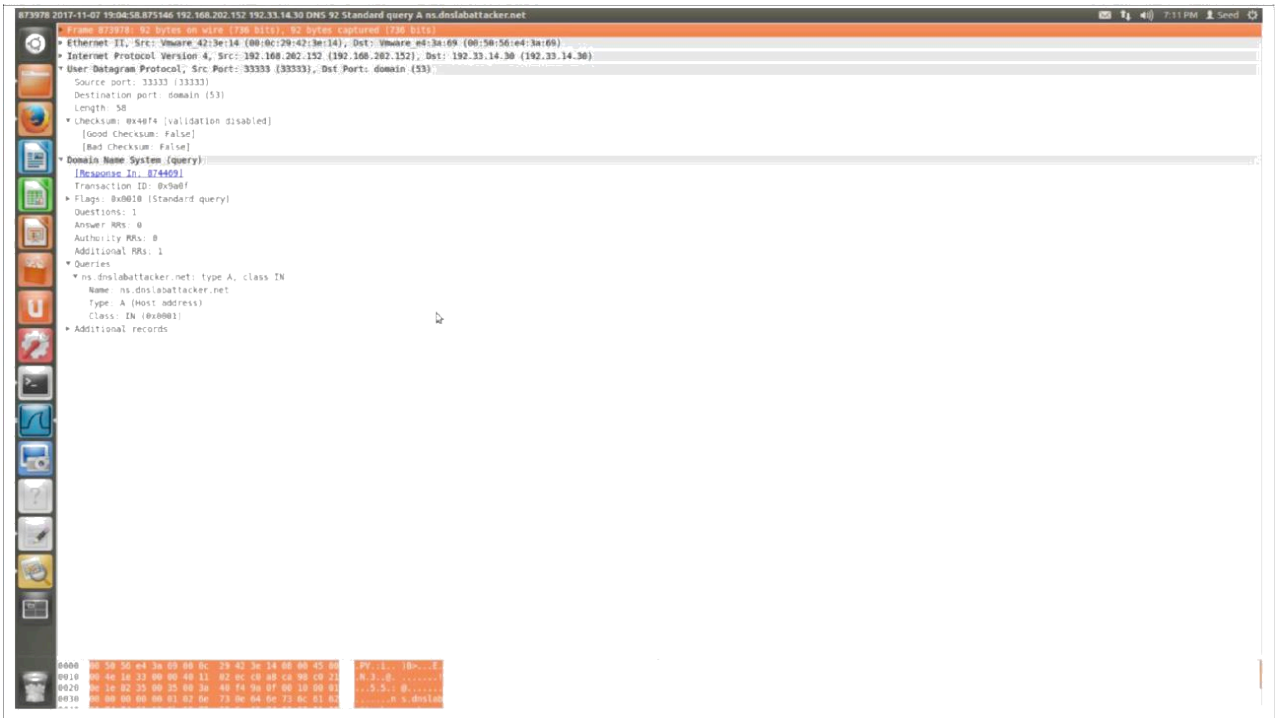
We also captured the packets that are flowing using wireshark tool and able to see the name in query section as ns.dnslabattacker.net.



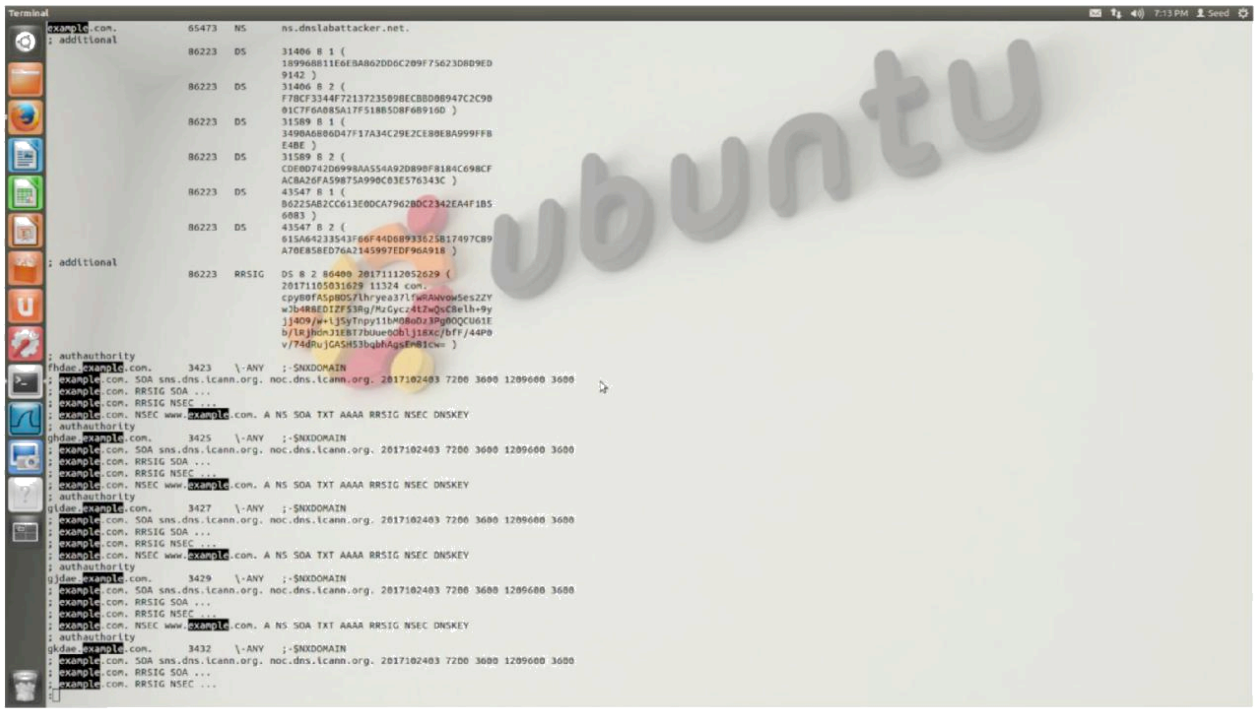
Task 2: Result Verification

Some of the scrrenshots in the attacker machine are:

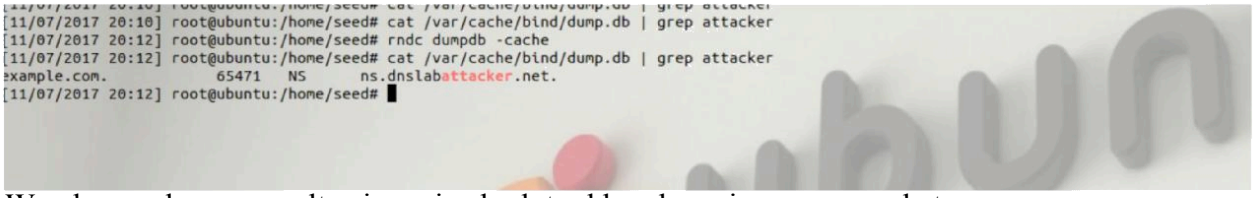




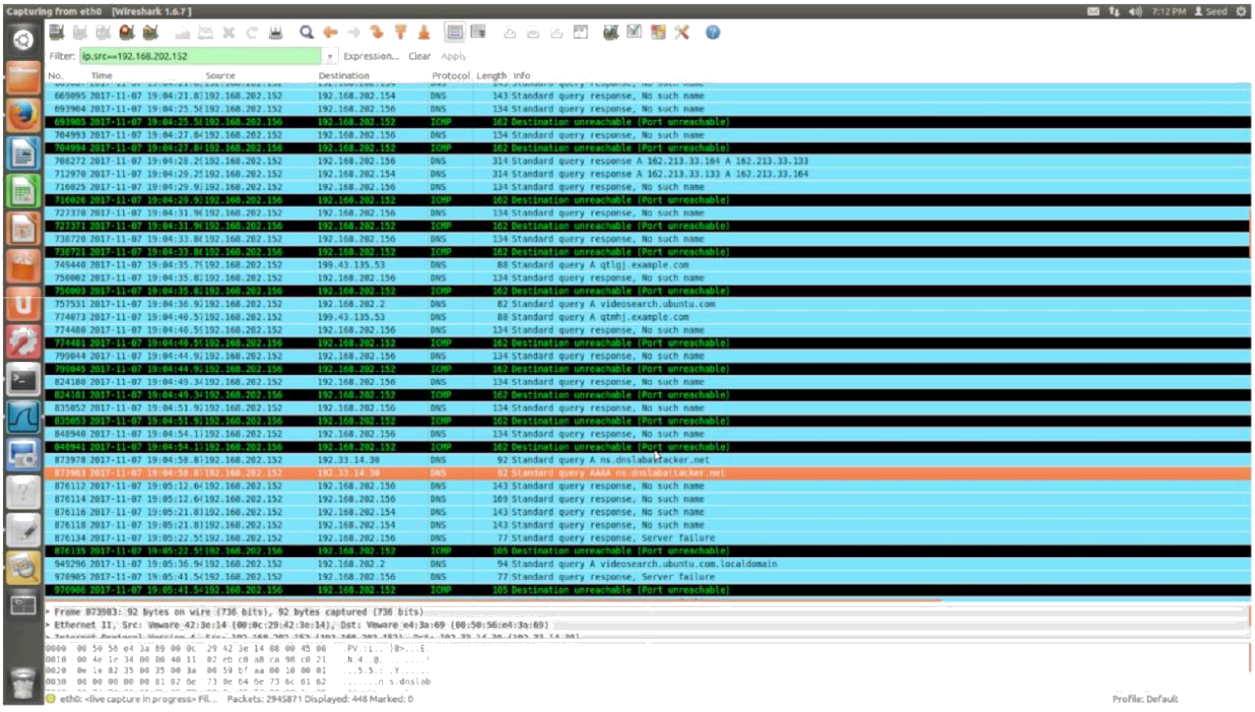
We also used the dig command to ask for example.com IP address. When Apollo receives the DNS query, it searches for example.com's NS record in its cache, and finds ns.dnslabattacker.net. It will therefore send a DNS query to ns.dnslabattacker.net.



verifying whether attack was successful in server machine:



We observed same result using wireshark tool by observing query packets.



QUESTION:

Note that although it's possible to successfully poison the name server of (nameserver of example.com is now ns.dnslabattacker.net) we can't provide an IP address for that nameserver (ns.dnslabattacker.net) in our spoofed response. If we try to include it as an additional RR, this may be ignored by if it uses the bailiwick rule. This just means that any records that aren't in the same domain of the question are ignored. So, if we ask for information about foo.example.com, then we only accept information in the additional section that is about example.com. Thus ns.dnslabattacker.net can't be answered in the additional section.

Thus we need another way to tell the resolver what's the ip address of ns.dnslabattacker.net. In our lab we simply store this ip address within resolver's name zone.