DNS Cache Poisoning

CSE 406

Computer Security Sessional

Sanjay Malakar

1505057

Group: 12

1 Attack Overview

DNS (Domain Name System) is the Internet's phonebook; it translates hostnames to IP address and vice-versa. This is done via DNS resolution. DNS attacks manipulates this resolution process in various ways. One of them is DNS Cache Poisoning Attack. There are two main ways to perform this attack, local (where the attacker and victim DNS server are on the same network, where packet sniffing is possible) and remote (where packet sniffing is not possible). I implemented the remote DNS cache poisoning attack.

2 Lab Environment

To demonstrate this attack, I have used three virtual machines, which runs on one single physical machine.

1. A DNS server
2. Victim user
3. Attacker which also hosts fake DNS server

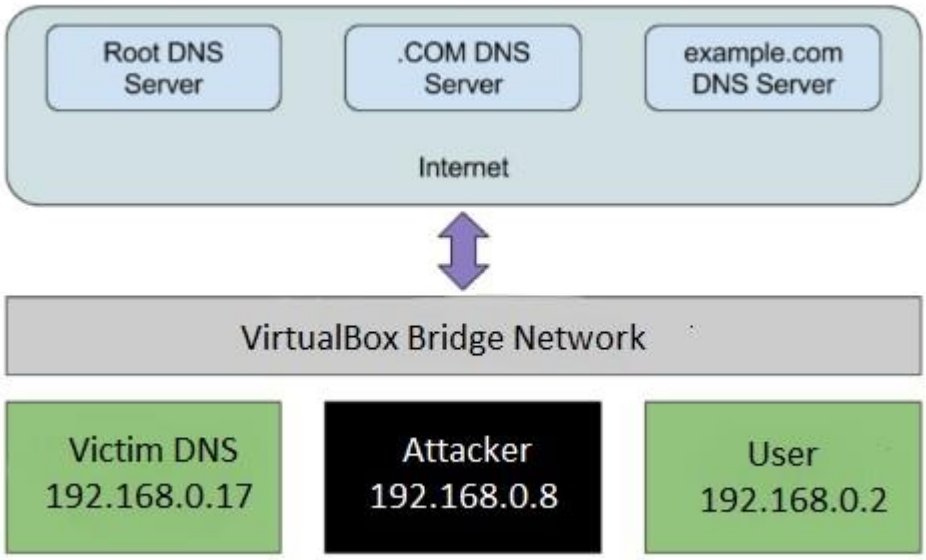


Fig 1: Lab Environment Setup

2.1 Local DNS server configuration

I have used Bind9 to setup my local DNS server which I will attack to poison.

To dump the cache to a file I have added the following contents in /etc/bind/named.conf.options

```
options {
        dump-file               "/var/cache/bind/dump.db";
        query-source port 33333;
}
```

1. The dump-file defines the file where the DNS cache will be dumped.
2. Source *Ports.* Some DNS servers now randomize the source port number in the DNS queries; this makes the attacks much more difficult. For the sake of simplicity, I have assumed that the source port number is a fixed number and set to 33333.

2.2 User machine configuration

On the 192.168.0.2 user machine I have set victim DNS server 192.168.0.17 as the default DNS server.

This is done by changing the DNS setting file /etc/resolve.conf in user machine:

```
nameserver 192.168.0.17
```

2.3 Attacker machine configuration

I used domain name www.example.com as our attacking target. The authentic IP address of www.example.com is 93.184.216.34, and its name server is managed by the Internet Corporation for Assigned Names and Numbers (ICANN).

The goal of this attack is to launch a DNS cache poisoning attack on the local DNS server, such that when the user runs the dig command to find out www.example.com's IP address, the local DNS server will end up going to the attacker's name server ns.dnslabattacker.net to get the IP address, so the IP address returned can be any number that is decided by the attacker. As a result, the user will be led to the attacker's web site, instead of the authentic www.example.com.

The main difficulty is the fact that the transaction ID in the DNS response packet must match with that in the query packet. Because the transaction ID in the query is usually randomly generated, without seeing the query packet, it is not easy for the attacker to know the correct ID.

Usually a request is cached in DNS server for days, this makes the attacker to wait until the DNS cache to time out which can take hours or days.

The Kaminsky Attack. Dan Kaminsky came up with an elegant technique to defeat the caching effect is to query a bogus subdomain of the hostname so that it's not in the DNS cache making the DNS server to query again.

The following is done in attacker machine when attack is run:

1. The attacker queries the DNS Server for a non-existing name in example.com, such as abcde.example.com, where abcde is a random name
2. Since the mapping is unavailable in DNS cache, it sends a DNS query to the nameserver of the example.com domain
3. While DNS server waits for the reply, the attacker floods server with a stream of spoofed DNS response, each trying a different transaction ID, hoping one is correct. In the response, not only does the attacker provide an IP resolution for abcde.example.com, the attacker also provides an "Authoritative Nameservers" record, indicating

ns.dnslabattacker.net as the name server for the example.com domain. If the spoofed response beats the actual responses and the transaction ID matches with that in the query, DNS server will accept and cache the spoofed answer, and its DNS cache is poisoned

4. Even if the spoofed DNS response fails (e.g. the transaction ID does not match or it comes too late), it does not matter, because the next time, the attacker will query a different name, so DNS server has to send out another query, giving the attack another chance to do the spoofing attack. This effectively defeats the caching effect

For the attack I have sent 1000 spoofed response of that request and tried the attack 1000 times. As the transaction ID is 16 bits this makes the probability of the attack being successful:

1 - (1-1000/2^16)^1000 (though many packets will be ignored)

If the attack succeeds, in DNS cache, the nameserver for example.com will be replaced by the attacker's name server ns.dnslabattacker.net.

So, when a resolve request is made for example.com is made it will be redirected to ns.dnslabattacker.net, but actually ns.dnslabattacker.net does not exist.

So, I have created a fake DNS nameserver named as ns.dnslabattacker.net on attacker's machine using Bind9 such that when it receives queries for example.com it returns the IP of attacker's machine (192.168.0.8) where I have hosted a website to demonstrate the attack.

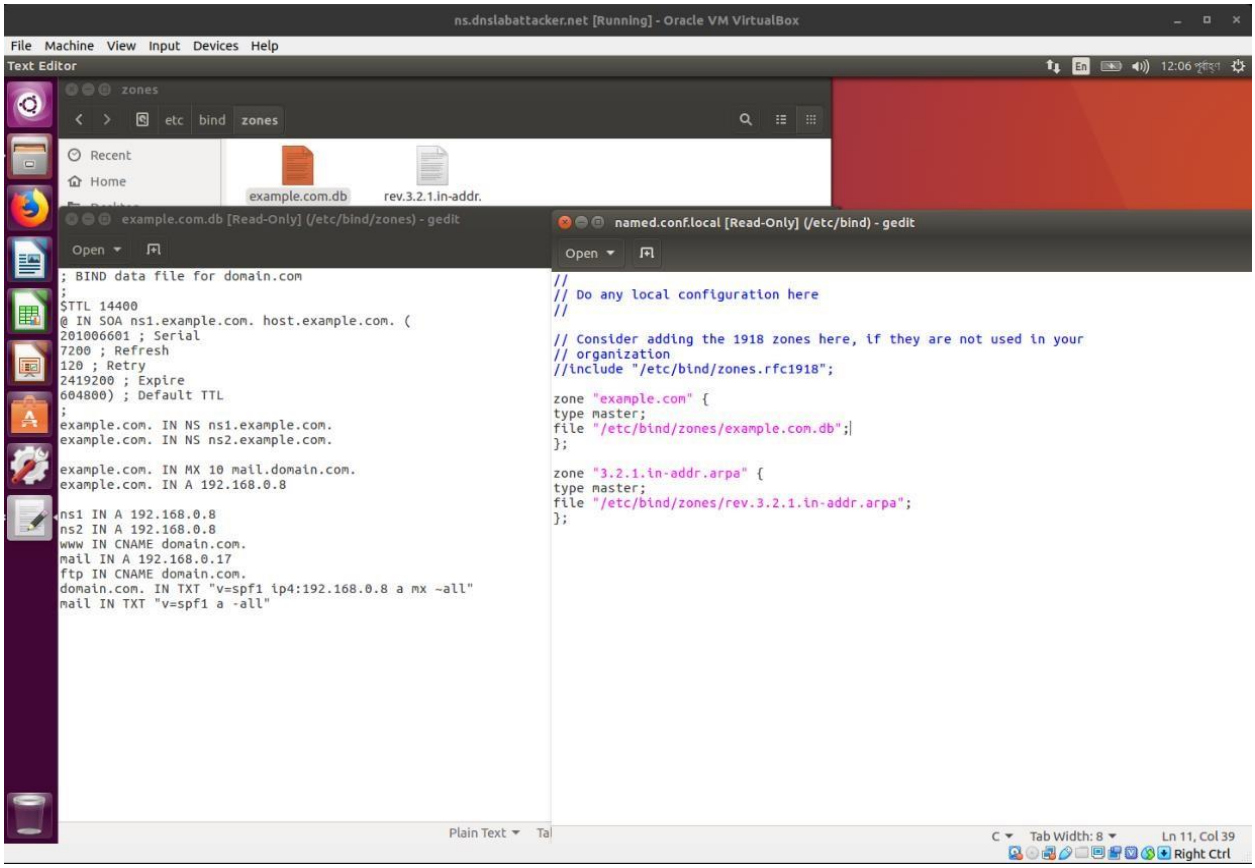


Fig 2: ns.dnslabattacker.net DNS server

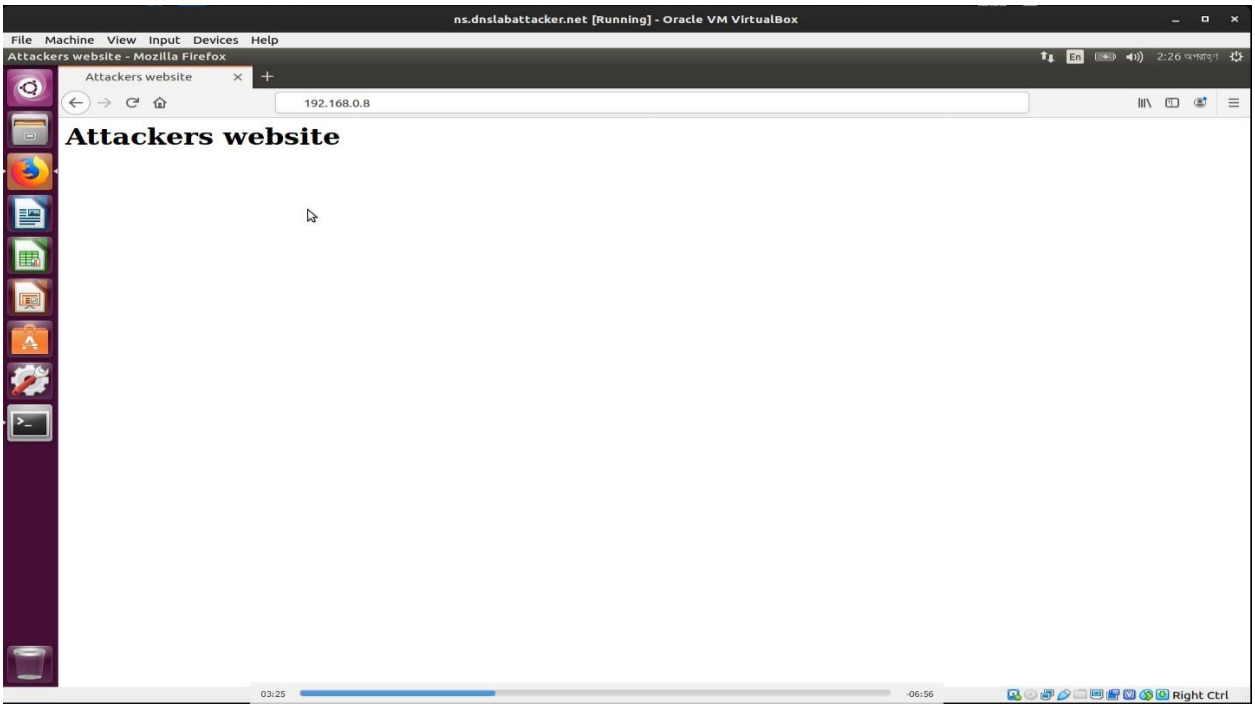


Fig 3: Attacker's website on 192.168.0.8

3 Steps of attack

Steps of my DNS cache poisoning attack:

1. I ran the attack on attacker's machine

```
sanjay@sanjay-GL552VW:~/Desktop$ sudo ./dns --ip=192.168.0.17 --domain=example.com --orig-ns=199.43.133.53 --attacker-ns=ns.dnslabattacker.net --attacker-ip=192.168.0.4 --n-requests=100 --n-responses=1000 --n-tries=1000
```

   ip=Victim DNS server's IP
   domain=example.com
   orig-ns=Original nameserver of example.com (199.43.133.53)
   attacker-ns=the fake ns of attacker (ns.dnslabattacker.net)

   n-requests=Total duplicate requests per subdomain

   n-response=Total spoofed response per subdomain

   n-tries=Total number attacks

2. Upon successful attack the nameserver of example.com will be set as ns.dnslabattacker.net on victim DNS server's cache.

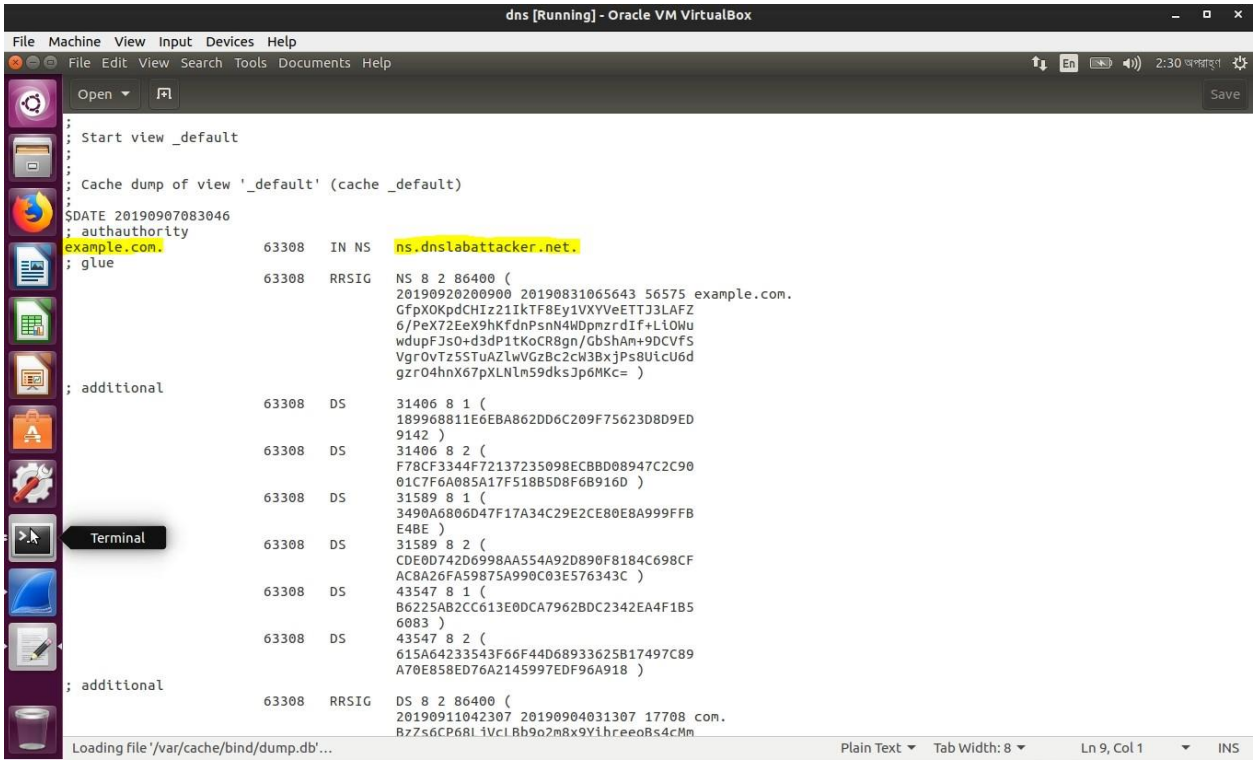


Fig 4: Victim DNS server's cache after attack

3. Upon checking the packets on Wireshark I found the request and response where the DNS server is poisoned.

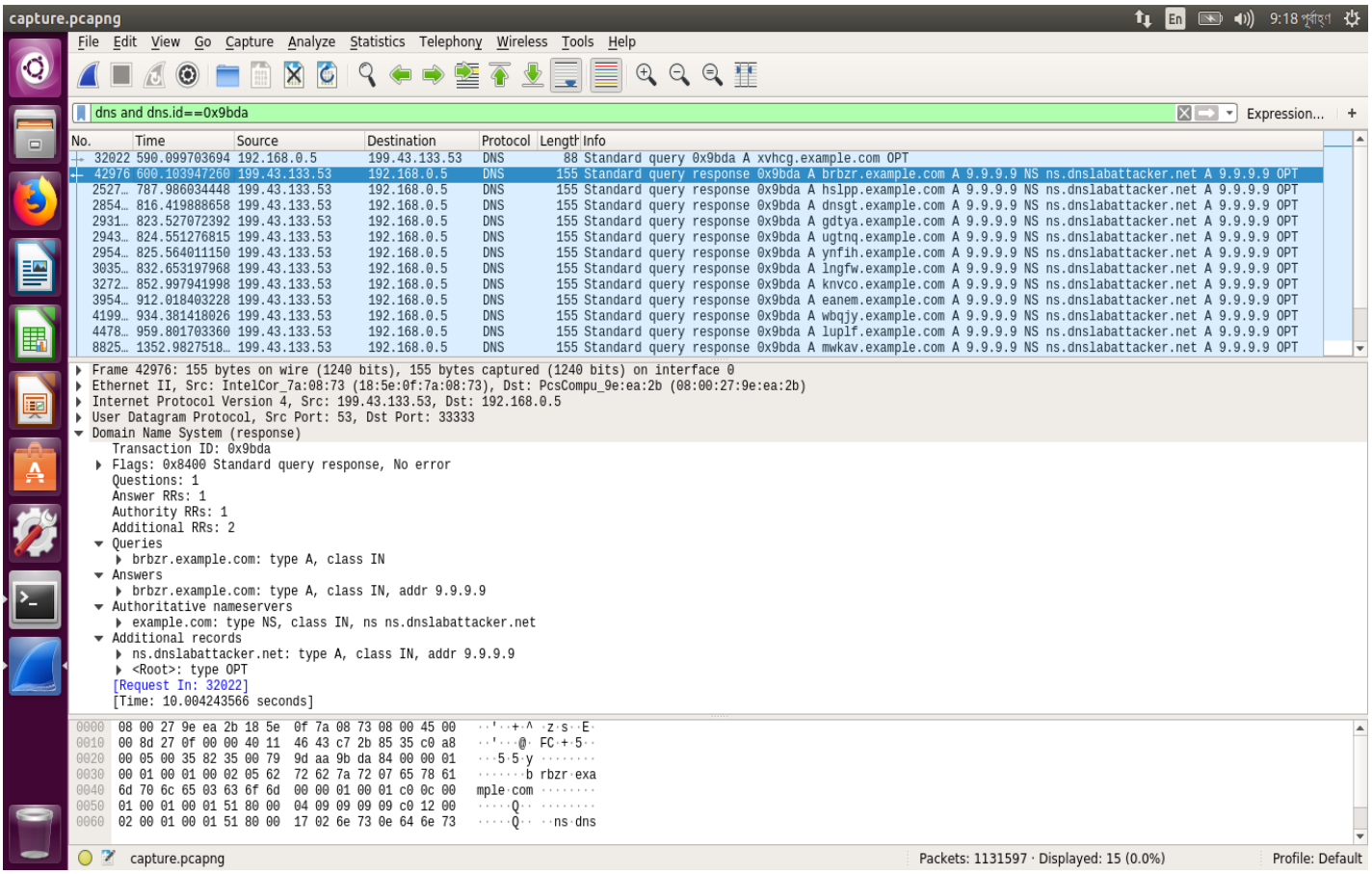


Fig 5: Wireshark log of a successful attack

4. On victims' machine (192.168.0.2) where default DNS server is set as poisoned DNS server (192.168.0.17), I ran dig example.com and it returned 192.168.0.8 which was set by the attacker on ns.dnslabattacker.net's DNS server.

5. Then I opened example.com on browser and it returned attacker's website as expected.

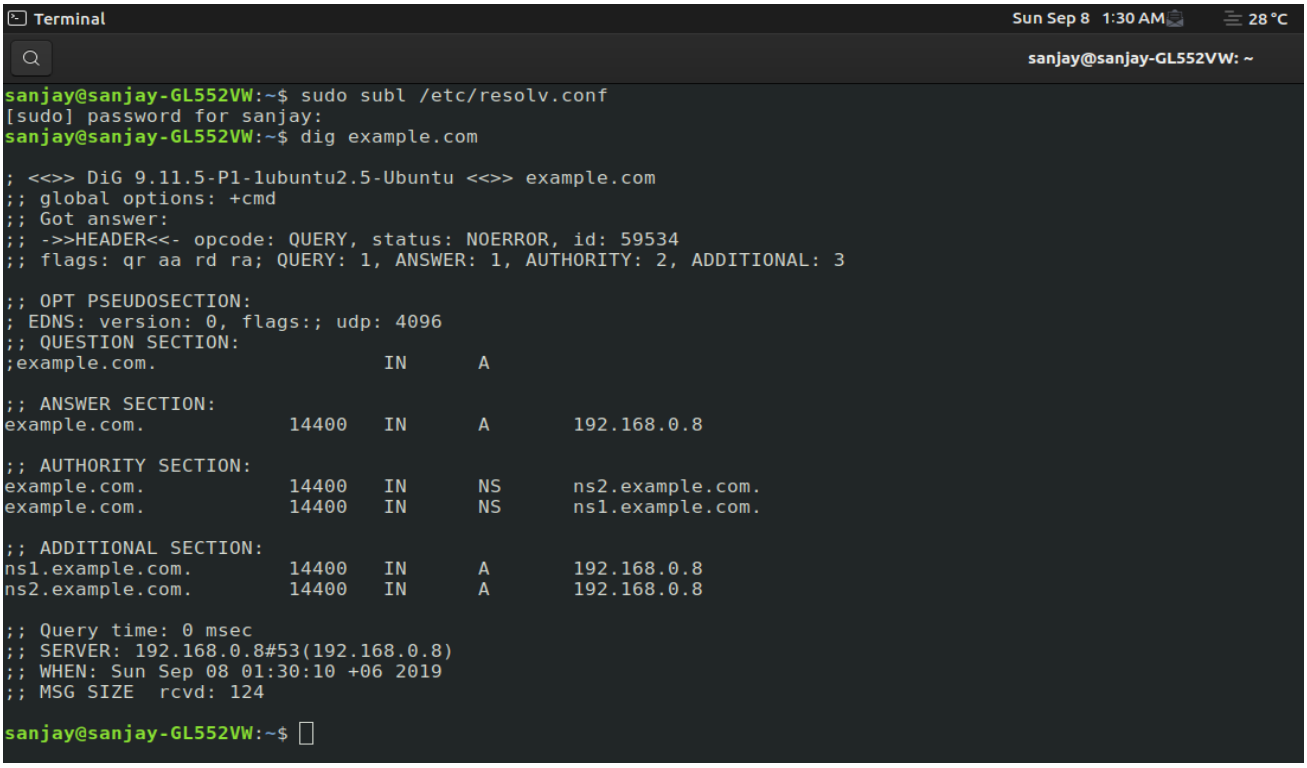


Fig 6: Result of dig example.com on victim user (192.168.0.2)

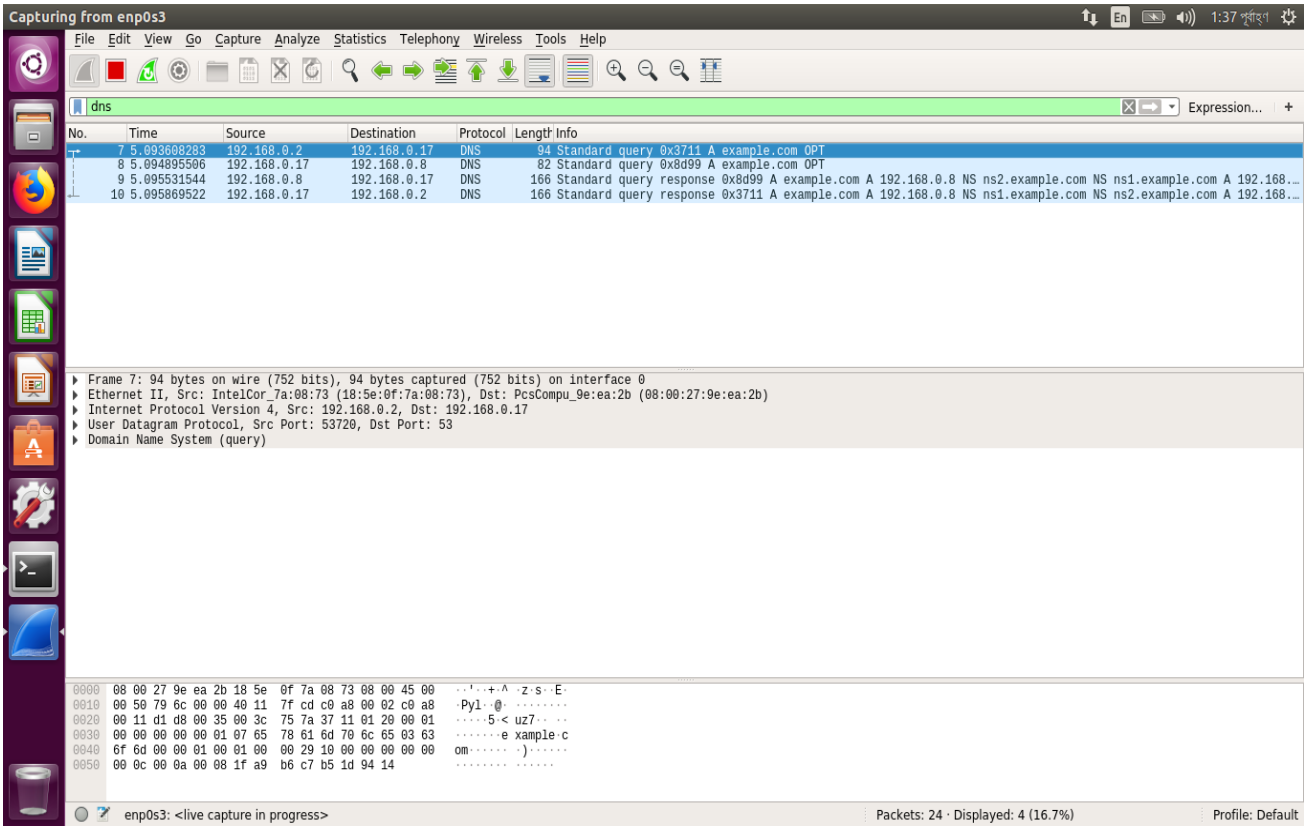
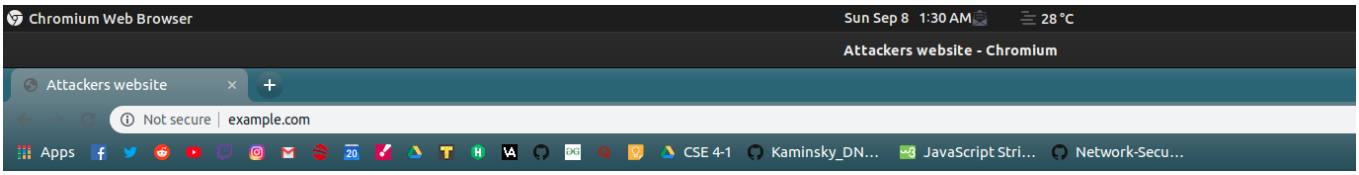


Fig 7: example.com resolve in DNS server (192.168.0.17)



Attackers website

Fig 8: example.com in browser of victim user (192.168.0.2)

4 Attack results

So, the attack was successful as it was able to poison the DNS cache with fake nameserver ns.dnslabattacker.net for example.com.

I send 1000 fake response and tried the attack for 1000 times, this makes the probability of the attack to succeed:

1 – (1-1000/2^16)^1000 = 0.99999979005124 (which is close to 1, so the attack was successful every time)

The spoofed response was faster than the actual response and the transaction id was the same of the requested id. So, the DNS server accepted the spoofed response. In the additional field of the response the nameserver of example.com was set to ns.dnslabattacker.net.

So, the DNS server cached the nameserver ns.dnslabattacker.net for example.com.

5 Countermeasure

There are several measures that enterprises should take to prevent DNS cache poisoning attacks. For starters, we should configure DNS servers to rely as little as possible on trust relationships with other DNS servers. Doing so will make it more difficult for attackers to use their own DNS servers to corrupt their targets' servers. Beyond limiting trust relationships on the DNS, we should ensure that we are using the most recent version of DNS. Domain Name Systems that use BIND 9.5.0 or higher include features such as port randomization and cryptographically secure Transaction IDs, both of which help prevent cache poisoning attacks.

In order to further prevent cache poisoning attacks, we should configure DNS name servers to:

1. Limit recursive queries.

2. Store only data related to the requested domain.

3.Restrict query responses to only provide information about the requested domain.

There are also cache poisoning tools available to help organizations prevent cache poisoning attacks. The most popular cache poisoning prevention tool is probably DNSSEC (Domain Name System Security Extension). DNSSEC is a cache poisoning tool developed by the Internet Engineering Task Force that provides secure DNS data authentication.

We can enable DNSSEC in our Bind9 DNS server by editing the

/etc/bind/named.conf.options file:

```
options {
        dnssec-validation auto;
        dnssec-enable yes;
        dnssec-lookaside auto;
}
```

We can also enable port randomization by removing query-source port 33333; from our /etc/bind/named.conf.options file.

This will prevent the DNS cache poisoning attack.