# Assignment 3 - DNS Attacks

Soumya Sen

10/16/2016

## 1 Machine Details:

MacBook Pro (Retina, 13-inch,Early 2015)

**Processor:** 2.7 Ghz Intel Core i5

**Memory:** 8 GB

**Serial Number:** C02S2XC3FVH5

In the first three tasks, 'Client' was the user, 'Server' was the DNS Server and the 'Attacker' was the attacker in my Virtual machine trying to sniff the packets. In the fourth task, Server2 was the forwarding resolver server.

## 2 Task 1: HOSTS File Attack

1. Exact steps taken: In this task, the client, server and the attacker are all on the same network. Here the attacker modifies the user's 'hosts file' and compromises the user's machine. In the host file, I added the IP address of UF news (ie 128.227.9.48) for www.example.com. So, whenever a user types www.example.com, it gets redirected to the UF news website.

2. As demonstrated in the screenshots, I edited the HOSTS file by mapping the IP of the UF news website (128.227.9.48) against the www.example.com by entering sudo vi hosts. Even the dig command shows that it's getting redirected to the UF news website.

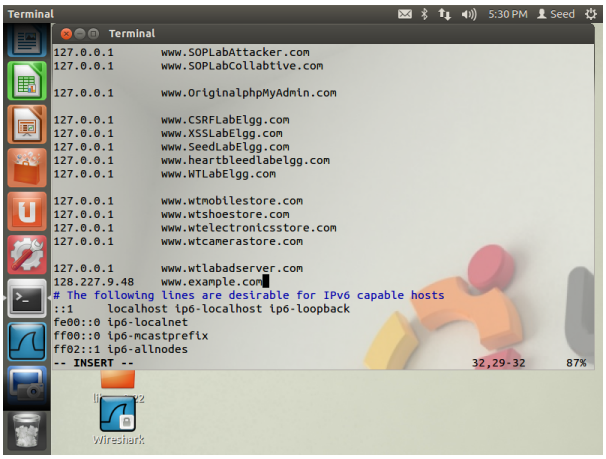


Figure 1: Changed hosts file

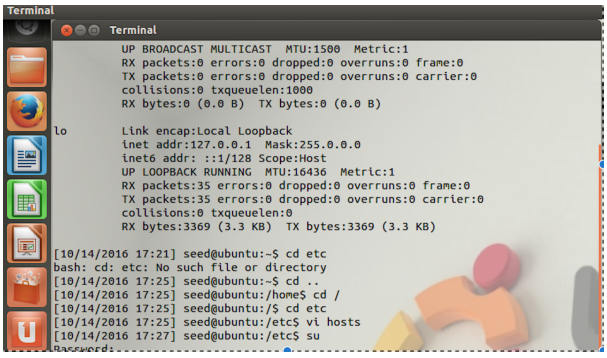


Figure 2: Command for changing the hosts file

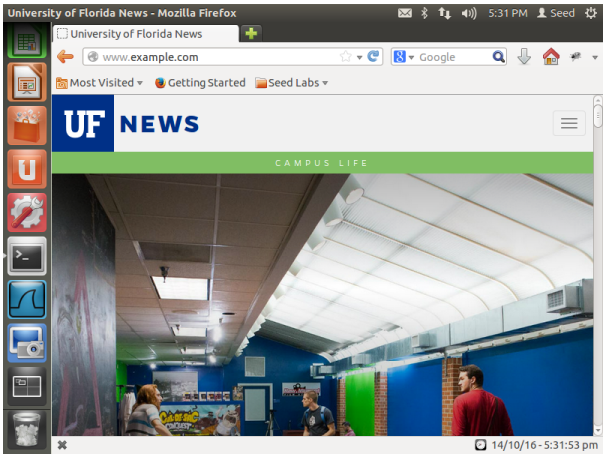


Figure 3: Redirection to the UF news website

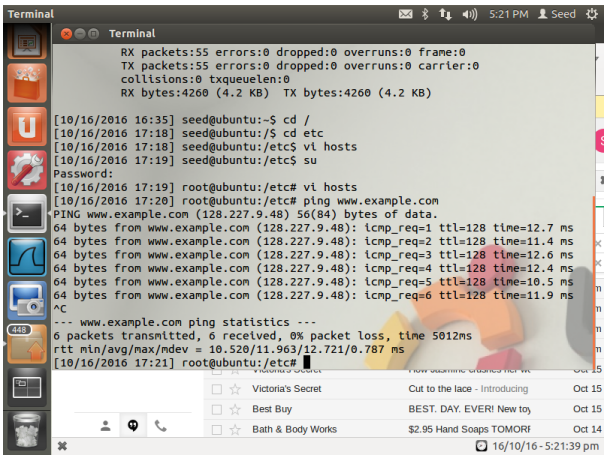


Figure 4: Digging the example.com and checking the ip configuration

3. How can a real life attacker leverage this attack: This attack is very basic. A naive attacker can also change the entries in the host file. If an attacker gets hold of victim's machine for even a short period of time, he can easily launch this attack by redirecting the user to malicious websites. If the malicious websites contain virus, the user's machine might be compromised. A real life attacker can also create a spoofed website which looks like the original website of The Bank of America, Wells Fargo etc. It can result in loss of money.

4. Viable attack: This attack is not very viable in a real world scenario. The attacker has to get hold of the user's machine which is not very practical. This attack also is of local nature since the attack is on the victim's machine. This can happen in real life if the user's password is not very strong and the attacker can either guess it easily or has found the user's credentials by some means.

# 3 Task 2: Host-Level Response Spoofing

## 3.1 DNS Server setup

For this attack, I set up a DNS server (IP Configuration: 192.168.183.146). I first installed and configured bind9 server. This server has the example.com.db domain and the zone files, reverse lookup to answer all queries about the domain. The dump file for dump.db was also created.

## 3.2 User Machine Configuration

The user's DNS server's IP Configuration is 192.168.183.141. I changed the resolv.conf file of the user's machine to direct the nameserver to the DNS machine which is at 192.168.183.146. I also changed the network connection by disabling the DHCP lookup for this machine. In this attack, the attacker spoofs the DNS response directly to the user. When the user queries for example.com, the attacker sniffs the packets and answers before the actual DNS server does.

## 3.3 Attacker Machine Details

The attacker's IP Configuration is 192.168.183.145. I used the NetWag tool to spoof the response of the DNS server with the port 105. In the tool, I entered the malicious IP (ie 128.227.9.48) . For the server's

name and address, I entered the correct entries so that the user thinks that the response has come from the original DNS server.

1. Steps taken: I set up three VMS on NAT and Host-only (ie Client, Server, Attacker) . From the attacker's VM, I sniffed the packets from Netwag which the client sends to the server simultaneously. The attacker selects Netwag tool 105 (Sniff and send DNS). The attacker makes the netwag run simultaneously when the client is sending a dig command to the client. A spoofed IP configuration is sent successfully to the client via the Netwag tool.

2. As we can see from the screenshots, the website is getting redirected and the attack was successful. The response id from the dig and netwag is same ie 40471.

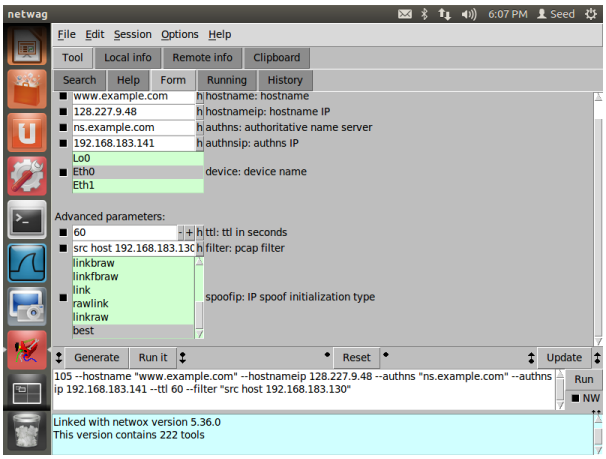


Figure 5: The configuration of the Attacker's Netwag

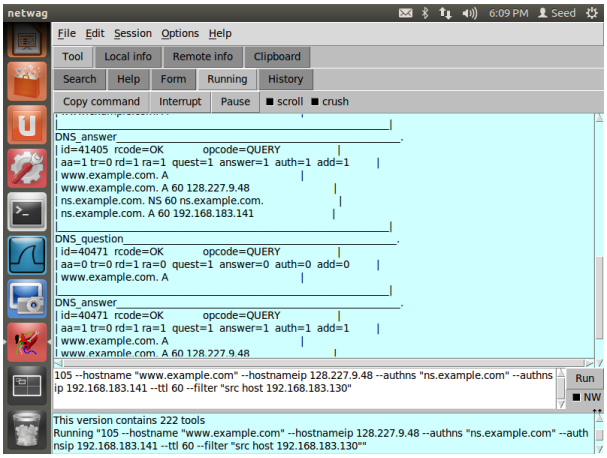


Figure 6: Netwag capturing packets for ID 40471

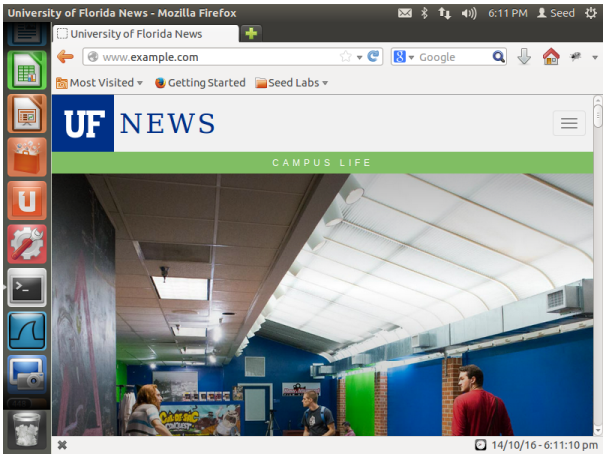


Figure 7: Redirection to the UF news website

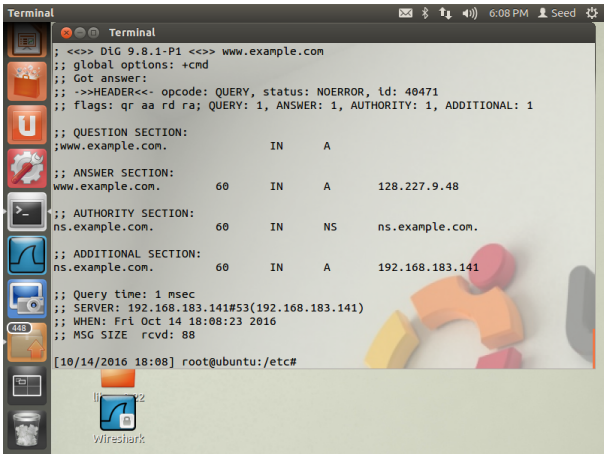


Figure 8: Dig www.example.com for ID 40471

3. How can a real life attacker leverage this attack: For this attack to be successful, the attacker needs to be in the same network as the client. If the attacker is on the same network as the victim, he can spoof the IP and send it to the client before the server can respond. Generally such attackers are insiders who know the

user's whereabouts or they are people who are trying to stalk the user. Similar to the previous attack, this can also result in loss of money or a user downloading virus on his computer.

4. Viable attack: In this task, the attacker needs to keep sniffing the packets which the client sends to the DNS Server and also spoof the packets before the server can reply. Inspite of these limitations, this can happen in real life. This can happen in crowded places like a park or restaurant where the attacker is in the same network and can see what the user is doing. The attack is highly improbable though because sniffing packets continuously is not feasible.

# 4 Task 3: Server-Level Response Spoofing

In the previous attack, the attacker continuously sniffed the packets of the user and spoof the response each time the client sends a request to the server. In this task the attacker poisons the user's query. This happens via Server-Level response spoofing.

1. Steps taken: In this attack, attacker, client and the server need to be in the same network. The attacker will try poisoning the cache of the DNS server in this task. Every time the client asks for the name resolution, the DNS server answers from its cache and the user gets a spoofed ip response. Here the target is the DNS server and not the client, unlike the second attack. The attacker flushes the cache and starts the attack. After that, I ran the netwag tool 105 from the attacker's machine. In the address field, the attacker enters the server's ip address (ie 192.168.183.141). After the configuration is done, the client sends a request for 'www.google.com' to the server and then the attacker sends a spoofed response for google website. The server's cache successfully gets poisoned and a spoofed ip response (1.2.3.4) is sent to the client.

2. These screenshots show the netwag configuration and we see that the netwag and the dig runs for the same id.

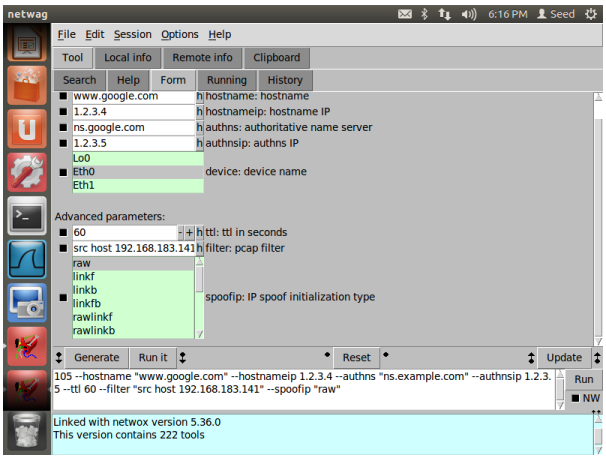


Figure 9: Attacker's Netwag configuration

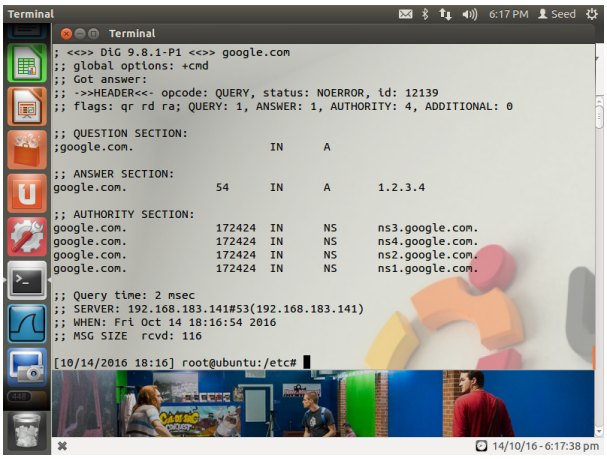


Figure 10: Victim's machine gets redirected to 1.2.3.4 for ID 12139

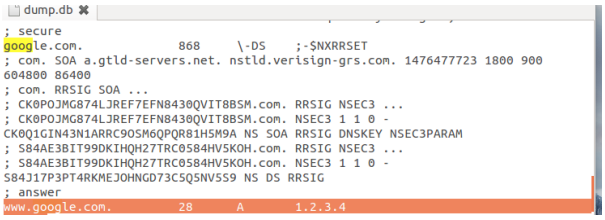


Figure 11: Poisoned cache of the server

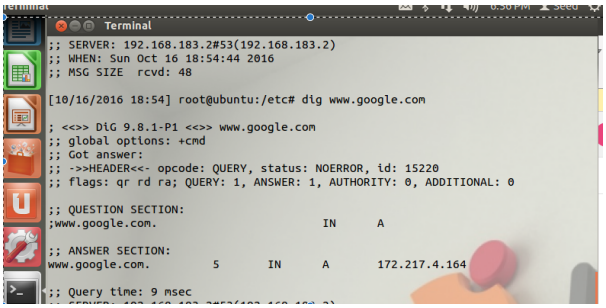


Figure 12: Original dig for google.com before the cache gets poisoned

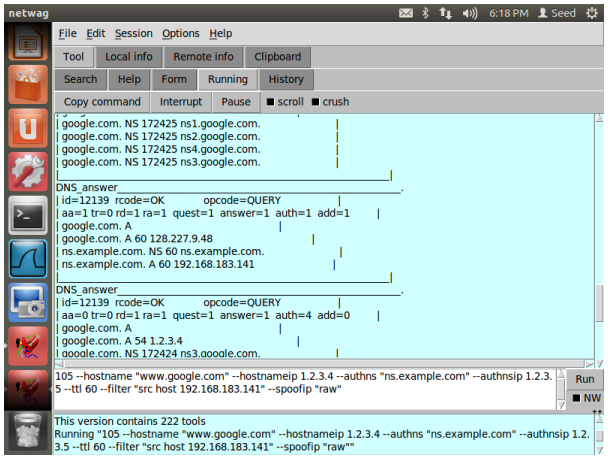


Figure 13: We see that for ID packets are getting spoofed for ID 12139 in the netwag and dig command

3. How can a attacker leverage this attack: This attack can be very dangerous because this can affect the user's machine permanently. It can result in large scale Denial of Service by dumping all the traffic from the users onto an unsuspecting server. It can result in massive phishing scams as well. This is a strong attacking method compared to the attacks performed before. The same results of the previous attacks can be repeated but it will be on a larger scale.

4. Viable attack: This attack can happen but only when the attacker is in the same network as server. This attack can be done once to poison the cache completely so the attacker can be an insider. There is a timing problem. The attacker wouldn't come to know when the server flushes it's own cache. The attack needs to be launched before the address is stored in the cache. Hence, poisoning the cache like this can be difficult. The next attack (ie Kaminsky attack) is more logical and dangerous.

# 5 Task 4: Kaminsky Attack

Kaminsky Attack is a special type of DNS attack in which an attacker targets the DNS server cache and in this attack, the attacker doesn't even have to be in the same network. In this attack, the attacker prompts the server to ask other DNS servers by querying a non existing name in the same zone. The attacker then spoofs the DNS response packets by replying to the query and sending the spoofed response packets at the same time in the same domain. This fake address will be then stored in the original DNS server and the cache will then be poisoned.

## 5.1 The original DNS Server configuration

The IP configuration of the first DNS server is 192.168.183.146. After that, I added the query-source port 333333 in the named.conf.options file so that we know that the query was sent from the 33333 port. Then the DNS forwarder's IP address was added in the named.conf. I added 'query-source port 33333' in */etc/bind/named.conf.options* to ensure that the query was sent from 33333 port.

## 5.2 Forwarding Resolver DNS server configuration

1. Steps taken: Initially a dig is run to check if the first DNS server is forwarding it to the next DNS server. I configured a new DNS server for www.dnsphishinglab.com.db in the same way as for www.example.com.db. Then to delay the packets from the original DNS server, I executed the 'tc qdisc add dev eth0 root netem delay 10 ms' so that the forwarding DNS server can send the replies before the original DNS server. I set up the forwarding server in such a way that it contains the records with the actual address of dnsphishinglab.com. Then after that, I made changes in the pacgen program to send packets to the server followed by 5000 spoofed responses to increase the possibility of cache getting poisoned.

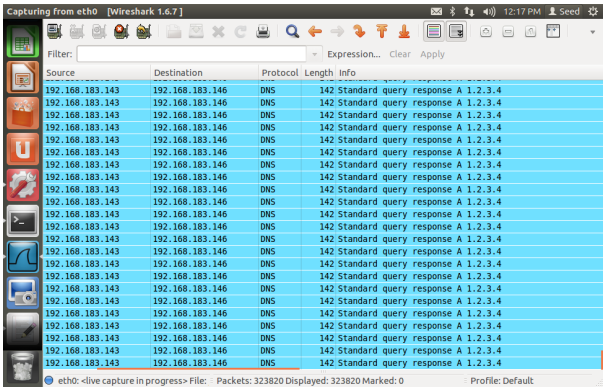


Figure 14: Requests between both DNS servers which shows that the spoofed response is getting sent

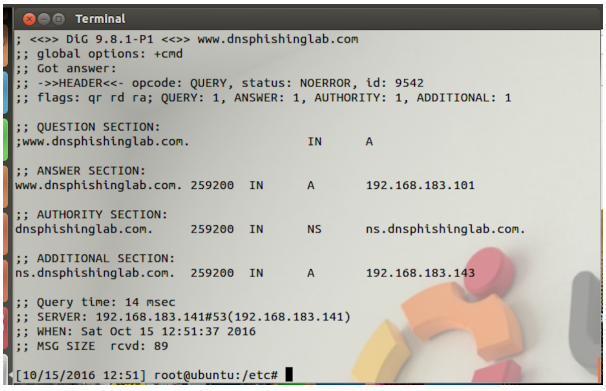


Figure 15: This dig shows that the dnsphishinglab is getting forwarded to resolver (192.168.183.143) but the local server is 192.168.183.141.

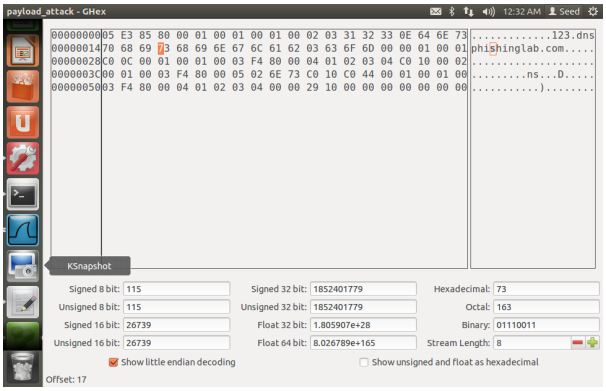


Figure 16: This is the payload request opened by Ghex. Out here, the spoofed response which has to be changed is 1.2.3.4

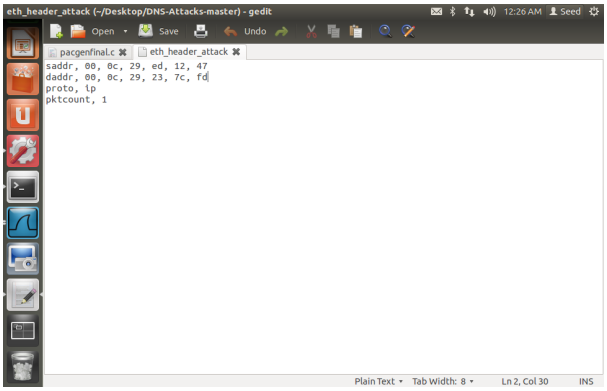


Figure 17: This is out configuration of the ethernet header request which contains the MAC address of the VM's

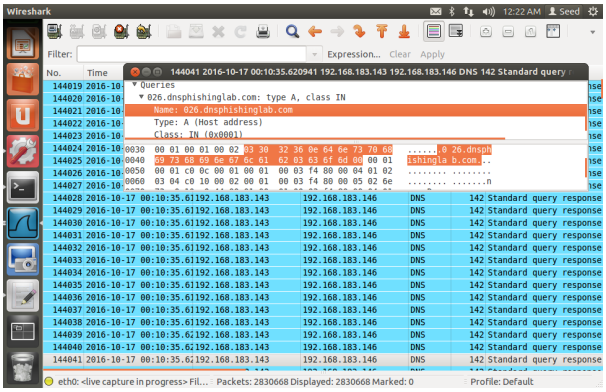

Figure 18: A random query of dnsphishinglab.com generated in Wireshark with the additional payload responses

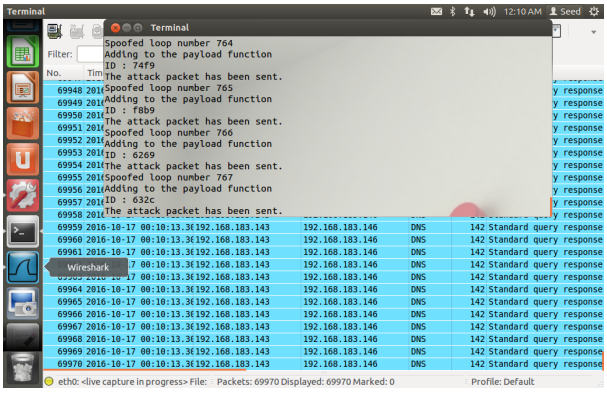

Figure 19: The pacgen.c program running and generating various spoofed response packets with a random id

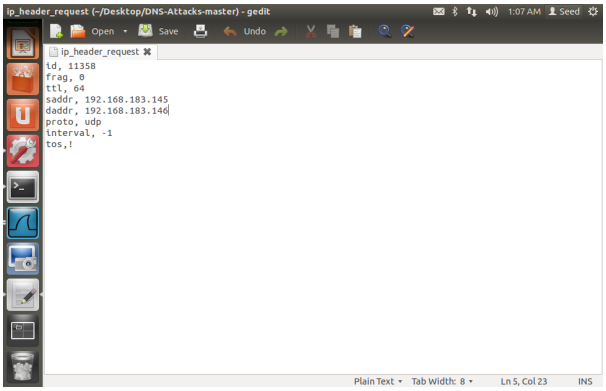

Figure 20: IP configuration of the forward resolver and the original server in the IP header request

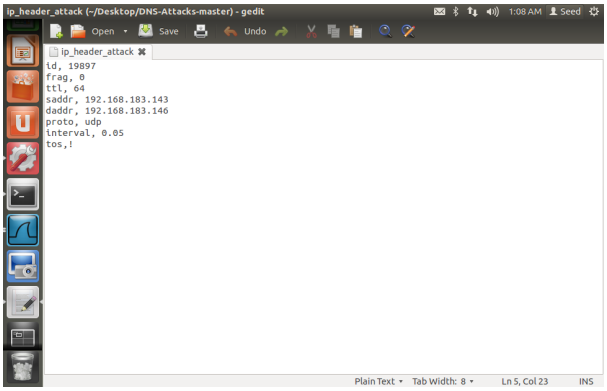

Figure 21: IP configuration of the attacker and the original server for IP header attack

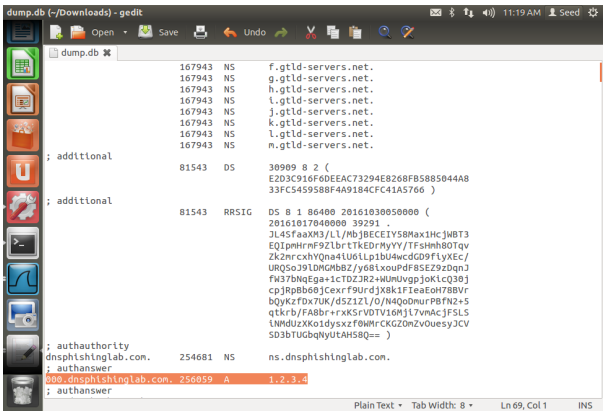

Figure 22: We see that the cache's ip configuration for a subdomain of dnsphishinglab gets changed to 1.2.3.4

2. Attack tool: Firstly, I changed the IP headers, ethernet headers and payload request according to the IP configuration of our machines. Then, I modified the pacgen.c in such a way that everytime a packet was being generated, there were 5000 spoofed responses for it. I created some random host names and ids by the rand() function in my program. The spoofed response is 1.2.3.4 which we can see in Wireshark. Once this was done, huge spoofed DNS responses were targeted to the original server.

3. My program was able to generate correct DNS packets with spoofed responses of 1.2.3.4. Initially I sent very few spoofed packets to match the transaction id. Later I created a bigger loop with more spoofed responses (In my case, it was 5000). Also, I had to delay the packets from the forwarding server by 10 ms to avoid the race condition.

4. To ensure that the packets were not sent to the network, I created the new DNS server which was the forward resolver for www.dnsphishinglab.com. The original server was then configured to send the request to the new DNS server. This was done by adding the zone files of dnsphishinglab.com. I also made these VMS into a Host only network so that it's not connected to the Internet. The NAT network was switched off in these VMs. Similarly, I switched off my laptop's wifi to run this attack.

5. How can a attacker leverage this task: This attack can easily be done on any DNS server. Here, the attacker doesn't even have to be on the same network as the client. The attacker queries the DNS Server with a non existing host name and before the actual server can reply, the attacker spoofs the IP and poison's the actual server's cache.

6. Viable attack: This attack is very viable and practical in the real world scenario. These type of attacks can be launched very fast and since the attacker is not in the same network as the client, the attacker can easily poison the DNS cache by including his malicious domain. Transaction IDs are also not that very difficult to guess (65,536 values) and generally the servers use the same port for resolving the queries. The chances can reduce by using source port randomization, DNS cookies or using DNSSEC but very few machines use these protocols. One limitation of this attack is that the attacker can not poison the cache which is already present in the DNS server. He has to somehow make the DNS server flush the cache or spoof domains which do not exist in the cache.