Assignment 3

Pharming Different Types of DNS Attacks

MOHIT MEWARA
84132114
OCTOBER 17, 2016

Contents

1 Introduction

This report contains the results for NSF-sponsored DNS Pharming Attack SEED lab. All the tasks specified in the lab has been performed successfully and screenshots of the same have been attached to the report. The report also contains a folder “CNS Assignment 3”, which contains all the files used to perform tasks. The files of each task are available in sub-folders inside the “CNS Assignment 2” folder.

2 Setting up Lab Environment

Firstly, we need to set up a lab environment, which contains 4 virtual machines - User VM, Attacker VM, DNS, DNS2.

- I have downloaded the VmWare Workstation for my windows machine.
- Then the 12.04 Ubuntu image is downloaded and executed successfully in VmWare Workstation.

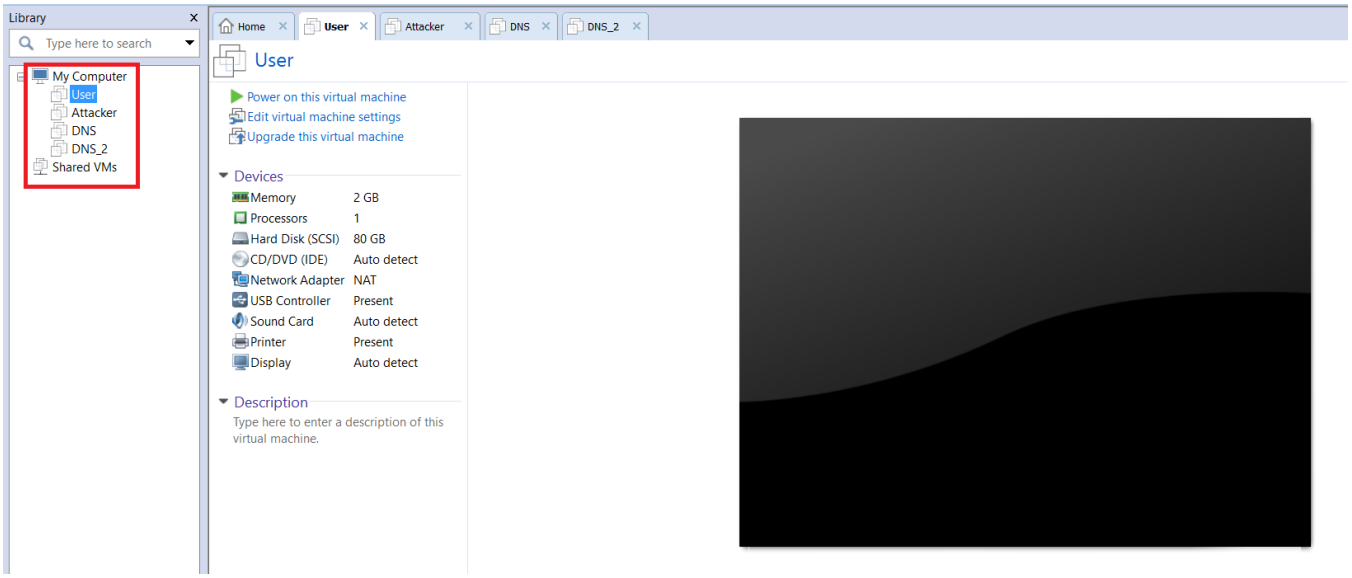

Figure 1: Four Virtual Machines

- All the VM’s are present in the same LAN environment. This closed environment is set in the same LAN because we don’t want the spoofed packets to reach the internet and make DNS queries to legitimate DNS servers. The below figure shows the IP address of different VM’s which are part of the lab.

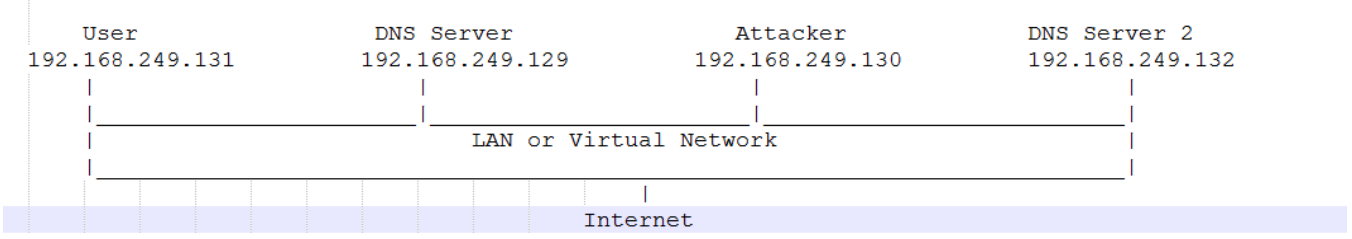

Figure 2: Lab Environment

2.1 Install and configure the DNS server

- **Step 1: Install the DNS server**
  We need to install BIND9 DNS server for starting the DNS server. But, the ubuntu image which we have downloaded already contains the BIND9 service. So, there is no need to download it.

- **Step 2: Edit named.conf.options file**
  At path(/etc/bind/) 2 files are present - named.conf and named.conf.options. The named.conf file is used by DNS to start and named.conf.options file is included by named.conf file. We added an entry for DNS dump - "/var/cache/bind/dump.db" into named.conf.options file.

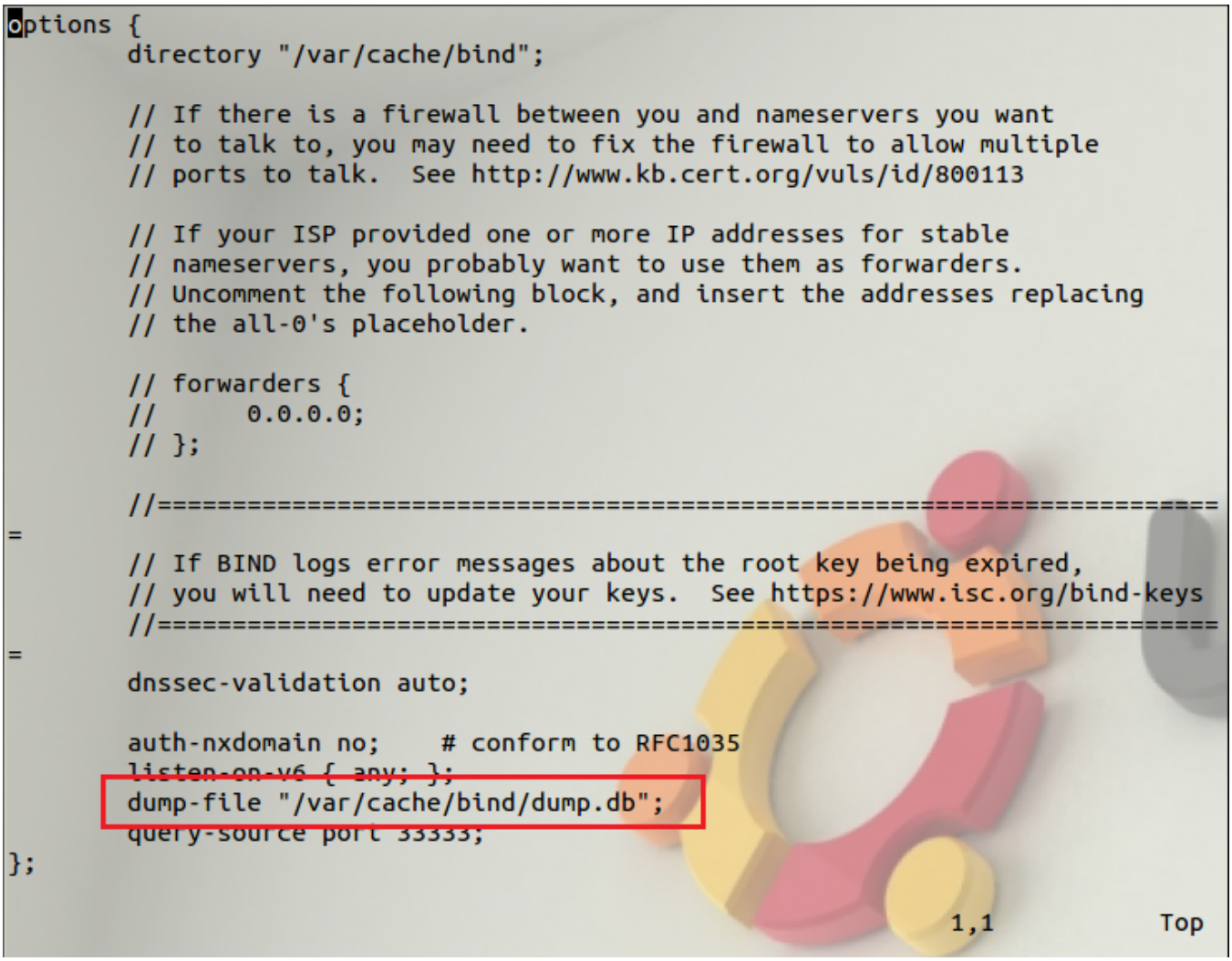

Figure 3: DNS dump file(dump.db) entry in named.conf.options file

- **Step 3: Create Zones**
  A zone is created for domain "example.com". The entry of zone is added to file - /etc/bind/named.conf. The IP 249.168.192 is added as zone because the IP of all Virtual Machines are in the subnet 192.168.249.x.

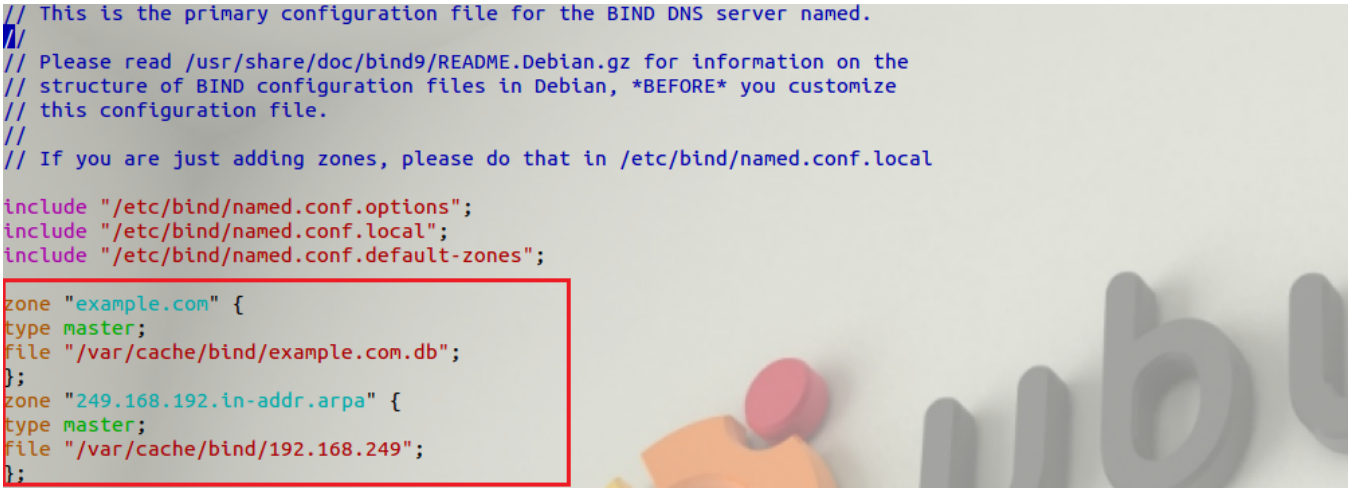

Figure 4: Zone of example.com added in named.conf file

- **Step 4: Setup zone files**
  The DNS lookup is put in the zone files. The zone file of "example.com" is composed at "/var/cache/bind/example.com.db". The Domain Name - IP Mapping is as follows:
  www.example.com = 192.188.249.101 (Ip address of domain)
  mail.example.com = 192.188.249.102 (Ip address of mail server)
  ns.example.com = 192.188.249.10 (Ip address of Name Server)

  A reverse DNS lookup file called 192.168.249 for example.com domain is composed at /var/cache/bind/

Figure 5: Zone file "example.com.db"

```
$TTL 3D
@       IN      SOA     ns.example.com. admin.example.com. (
                2008111001
                8H
                2H
                4W
                1D)
@       IN      NS      ns.example.com.

101     IN      PTR     www.example.com.
102     IN      PTR     mail.example.com.
10      IN      PTR     ns.example.com.

~
```

Figure 6: Reverse DNS Lookup file “192.168.249

- **Step 5: Start the DNS server**
  The DNS server is started by running the command.

$$sudo \quad /etc/init.d/bind9 \quad restart$$

```
[10/15/2016 17:26] root@ubuntu:/home/seed# sudo /etc/init.d/bind9 restart
 * Stopping domain name service... bind9
waiting for pid 3791 to die
                                                                    [ OK ]
 * Starting domain name service... bind9                           [ OK ]
[10/15/2016 17:26] root@ubuntu:/home/seed#
```

Figure 7: DNS Server Restart

2.2 Configure the User Machine

The DNS server of the User machine(192.168.249.131) is set as 192.168.249.129 in DNS setting file /etc/resolv.conf.
Also, the DHCP server is deactivated in Ubuntu to avoid DHCP to overwrite DNS setting file.

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.249.129
~
```

Figure 8: DNS Server Modified in resolv.conf file of the User machine

The dig command can be used to confirm that the DNS server has been changed to 192.168.249.129.

```
[10/15/2016 17:51] root@ubuntu:/home/seed# dig www.example.com

; <<>> DiG 9.8.1-P1 <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42165
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       192.168.249.101

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

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.249.10

;; Query time: 10 msec
;; SERVER: 192.168.249.129#53(192.168.249.129)
;; WHEN: Sat Oct 15 17:52:06 2016
;; MSG SIZE  rcvd: 82
```

Figure 9: Output of dig command

2.3 Configure the DNS2 Machine

The DNS2 machine has been configured in the same way as the first DNS machine has been configured. The zone of "dnsphishinglab.com" is defined in named.conf file. The dumps file entry is also included in named.conf.options file.

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "dnsphishinglab.com" {
type master;
file "/var/cache/bind/dnsphishinglab.com.db";
};
zone "249.168.192.in-addr.arpa" {
type master;
file "/var/cache/bind/192.168.249";
};
```

Figure 10: Zone entry of "dnsphishinglab.com" in named.conf

The zone file of "dnsphishinglab.com" is created at "/var/cache/bind/dnsphishinglab.com.db".

Figure 11: Zone file of "dnsphishinglab.com"

The reverse DNS lookup file called 192.168.249 for dnsphishinglab.com domain is composed at /var/cache/bind/

Figure 12: Reverse DNS lookup file

3 HOSTS file attack

(A) Attack Description:
Whenever the user enters domain name in browser or tries to perform nslookup for a particular domain, the User Machine first checks the local host file for Domain Name - Ip Mapping. And if the mapping is not present in the hosts file, the machine queries DNS server for Ip address of the domain. The "/etc/hosts" file has more preference over the remote DNS lookup. The host file attack can be successful if the User Machine is compromised by the adversary. In this attack, the adversary modifies the Domain Name- Ip Address mapping in "/etc/hosts" file present in User Machine to an Ip address with phished website.

(B) Steps:

(i) Open the "/etc/hosts" present in User machine.

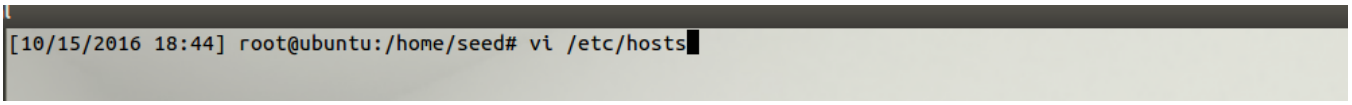

Figure 13: Command to open hosts file

(ii) Add an entry of www.microsoft.com in the hosts file and map it with some Ip. For this attack, I have mapped www.microsoft.com with 127.0.0.1 (localhost). This Ip can be changed to some malicious Ip by the adversary.

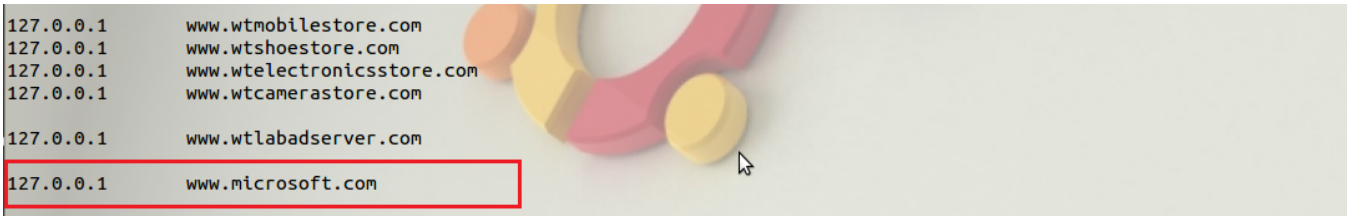

Figure 14: User's Hosts file modified

(iii) Now, if the user tries to open www.microsoft.com in browser then the website of localhost opens because the machine is reading Hosts file.

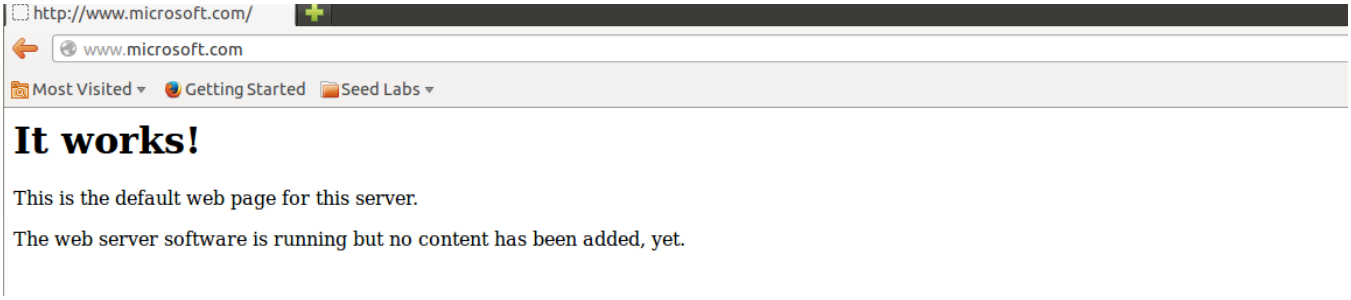

Figure 15: User's /etc/Hosts file modified

(C) How a real-world attacker could leverage this sort of attack?
A real-world attacker could make a rogue entry in hosts file which will redirect user to a phished website, which would look legitimate to the user. Then from the phished website, the attacker can ask for user credentials and as user thinks that he is talking to legitimate website, he passes on his credentials to the attacker. The attacker can later use the user credentials for malicious activities.

(D) Discuss whether you feel this is a viable attack in the real world. What factors contribute to your answer?
In real world most users don't checks hosts file frequently. So, if the adversary can get access to user's machine and change the hosts file then the user would not notice that the hosts file is compromised which makes this attack possible.
However, to make this attack successful, the adversary should have access to user's machine. In most cases, the adversary does not have access to user's machine and nowadays user's are also aware of protecting the password. With technology advancement, hacking is becoming harder, therefore it is hard for the adversary to make this attack viable.

4 Host-Level Response Spoofing

(A) Attack Description:
In this attack, the attacker sends a spoof DNS response to user, when the user sends a DNS request to DNS server. If the spoofed response and the original DNS response has same sequence number and

spoofed response reached the user machine earlier than the original DNS response, then the user machine will accept the spoofed response and the attack would be successful.

(B) Steps:

(i) For this attack, 3 Machines - User, DNS Server, Attacker are required. We have already set up the 3 machines, so we can use them for this attack. Also, this attack is successful, if the attacker and user lie over the same network. As all the machines are over same LAN, this attack is possible.

(ii) I then checked that whether I can sniff the DNS packets sent by the user to DNS server. I used Wireshark for packet sniffing. The following screenshot contains the DNS request packet sent by the user to server. The user(192.168.249.131) sends a request to the server(192.168.249.129) and then the server responds back to user. All these packets are visible to attacker(192.168.249.130)

Figure 16: Nslookup for “www.example.com”

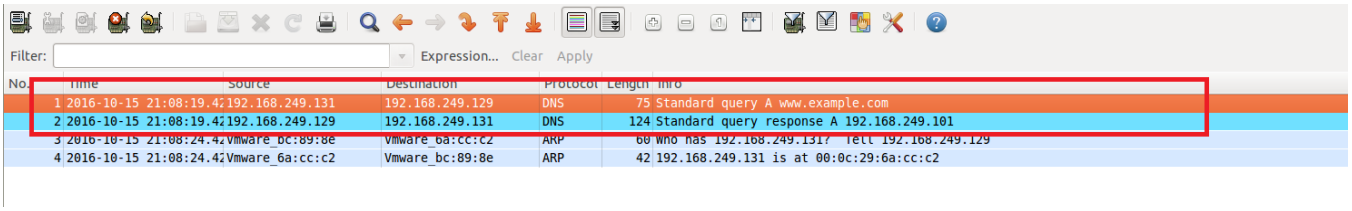

Figure 17: Packet Sniffing using Wireshark

(iii) Using Netwag tool No 105 (Sniff and send DND answers), I filled the form where I specified:
hostname = www.example.com
hostnameip = 1.2.3.4
authns = ns.example.com
authnsip = 1.2.3.5
device = Lo0
ttl = 100s
spoofif = raw

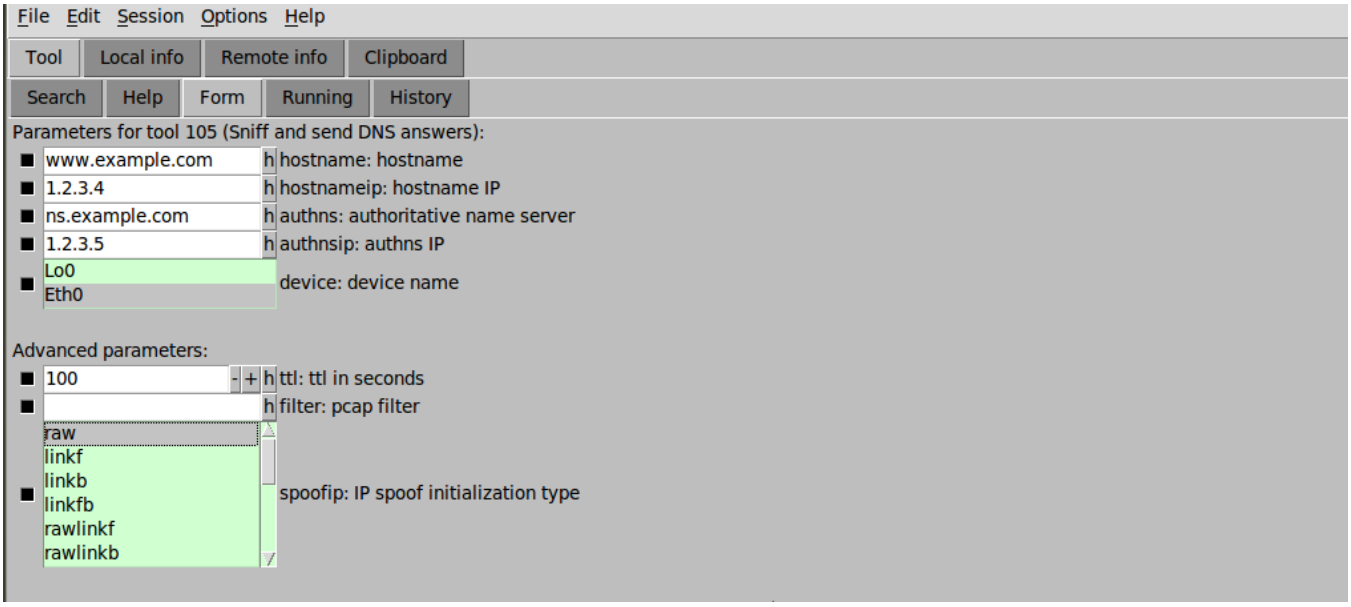

Figure 18: Form of Netwag tool

(iv) Then using user's machine, I run the command nslookup www.example.com. The Netwag tool detects the request from the user and sends the spoofed DNS response to user. The user machine got the response 1.2.3.4 instead of 192.168.249.131.

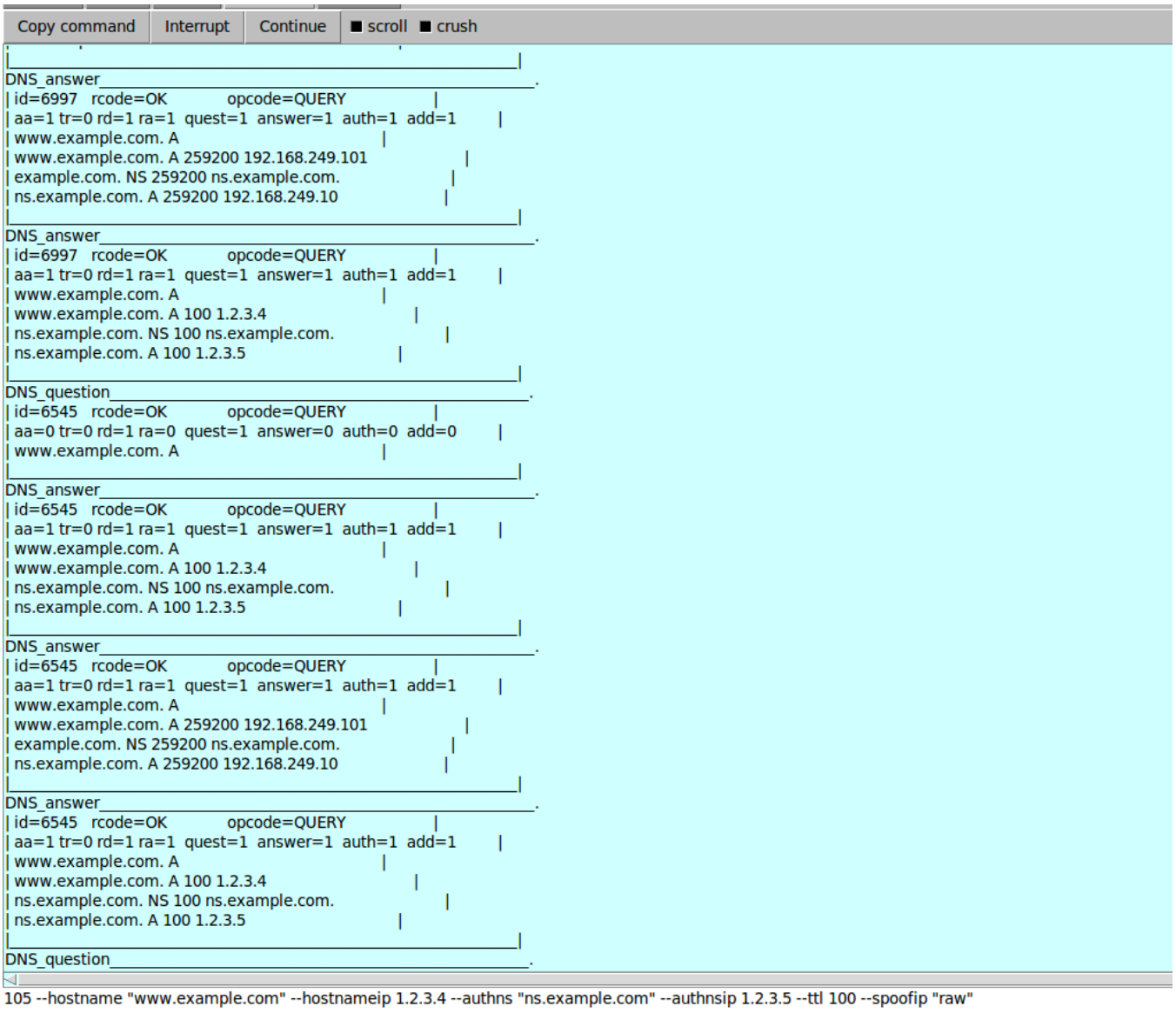

Figure 19: Spoofed Packets sent from Netwag to User

```
[10/15/2016 21:49] root@ubuntu:/home/seed# nslookup www.example.com
Server:         192.168.249.129
Address:        192.168.249.129#53

Name:   www.example.com
Address: 1.2.3.4
```

Figure 20: Nslookup response to User

(v) While checking the packets in Wireshark, it can be seen that the forged Netwag response reaches user prior legitimate DNS response. Also, the sequence number of all the packets are

same.

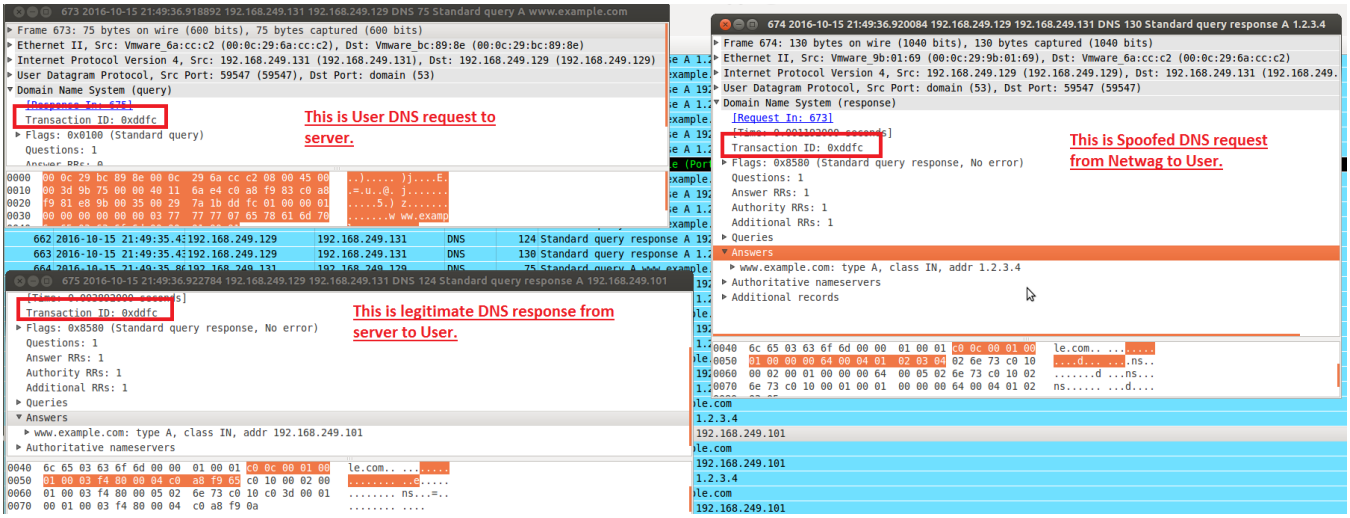

Figure 21: DNS request-response packets sniffed by Wireshark

(vi) The response of dig command is:

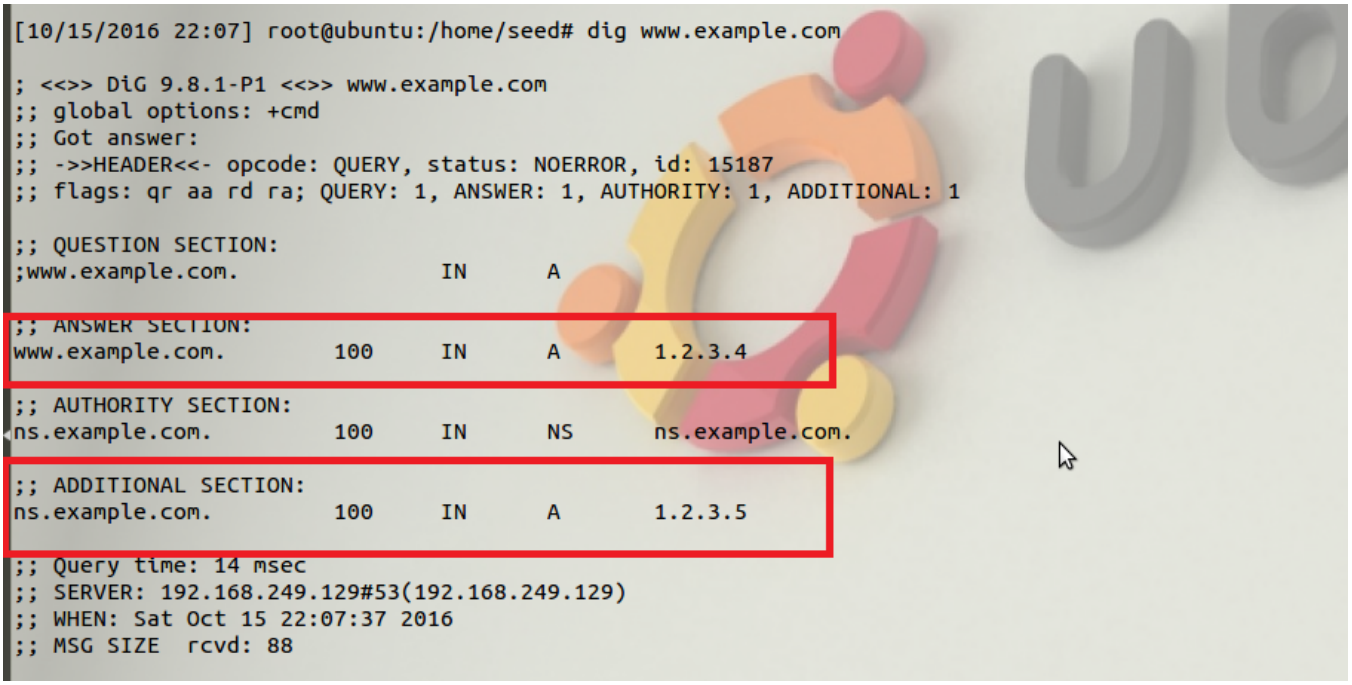

Figure 22: Dig Command Response showing Spoofed Domain Ip and Name server Ip

(C) Explain how a real-world attacker could leverage this sort of attack.
The attacker has to sniff the packets of User and then send the spoofed packets to user using Netwag tool. This means that the Attacker should have access to the user network. However, if the attacker has access to network and has successfully carried out the attack then the attacker can make user to visit a phished site. So, phishing attacks are possible using Host-Level Response Spoofing.

(D) Discuss whether you feel this is a viable attack in the real world. What limitations about this attack contribute to your answer?
Analyzing the following points, it seems that this attack is less viable in the real world.

- To make this attack successful, the attacker must have access to the user network.
- The attacker has to run Netwag tool for every domain which user visits frequently. This means that attacker must know all the sites which user visits which seem to be a tough process.
- The success rate is very low, as most of the times DNS response comes earlier than the spoofed response.
- The attacker has to perform this attack for all the users present in the network. So, DNS server cache poisoning attack is a better option for attacker than this attack.

5 Server-Level Response Spoofing

(A) Attack Description:
In this attack, the DNS server cache is replaced with the spoofed Ip address. When the user queries DNS for a domain and that domain is not present in the DNS server also, then the DNS queries root DNS for the domain name. So, if the response of root DNS server can be spoofed by the attacker then the entry in local DNS can be poisoned (replaced with spoofed one). Once the DNS is compromised, every user querying the DNS will get spoofed response.

(B) Steps:

(i) First, I checked that the User machine (192.168.249.131) is having a working internet connection because the DNS made by me during set up would be querying the Domain Name Server presents over the internet.

(ii) When I run the command “dig www.google.com” from the user, I got the following response:

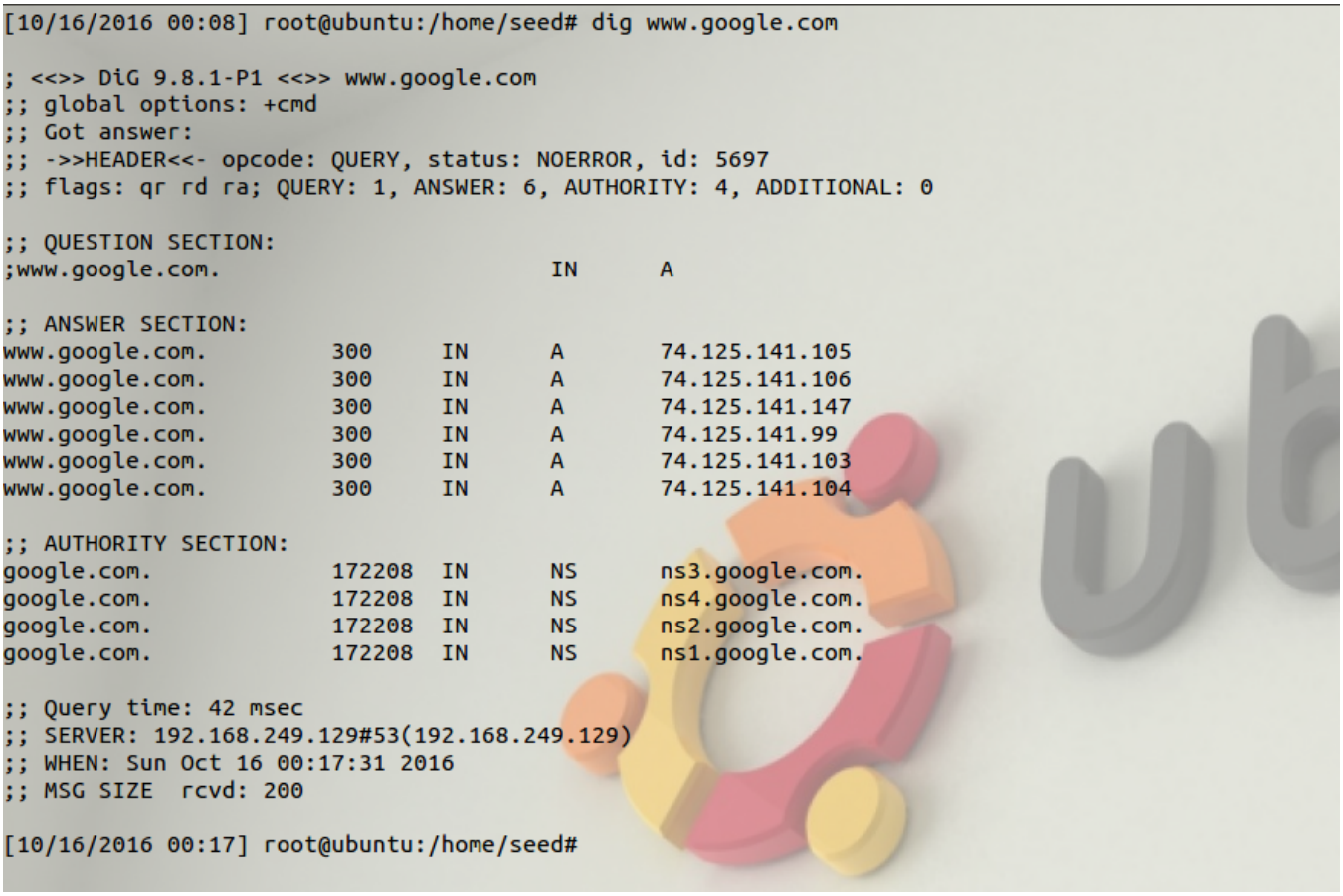

Figure 23: Legitimate dig response for www.google.com

(iii) This attack is only possible when the local DNS server (192.168.249.129) does not have the entry of the domain which the user will query. For this task, I will be querying "www.google.com" and thus I removed the cache of DNS by the command:

$$sudo \quad rndc \quad flush$$

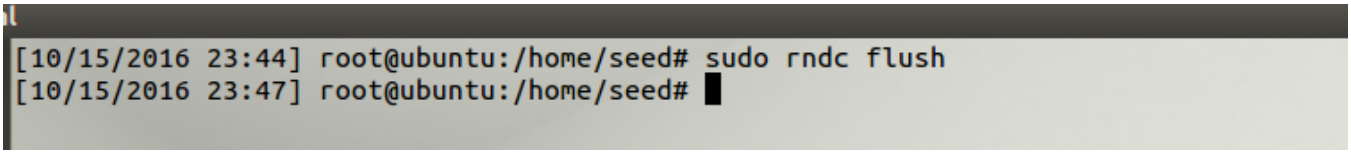

Figure 24: DNS Server (192.168.249.129) cache removed

(iv) Using Netwag tool No 105 (Sniff and send DNS answers), I filled the form where I specified:
hostname = www.google.com
hostnameip = 1.2.3.4
authns = ns.google.com
authnsip = 1.2.3.5
device = Lo0
ttl = 100s

filter = src host 192.168.249.129
spoofif = raw

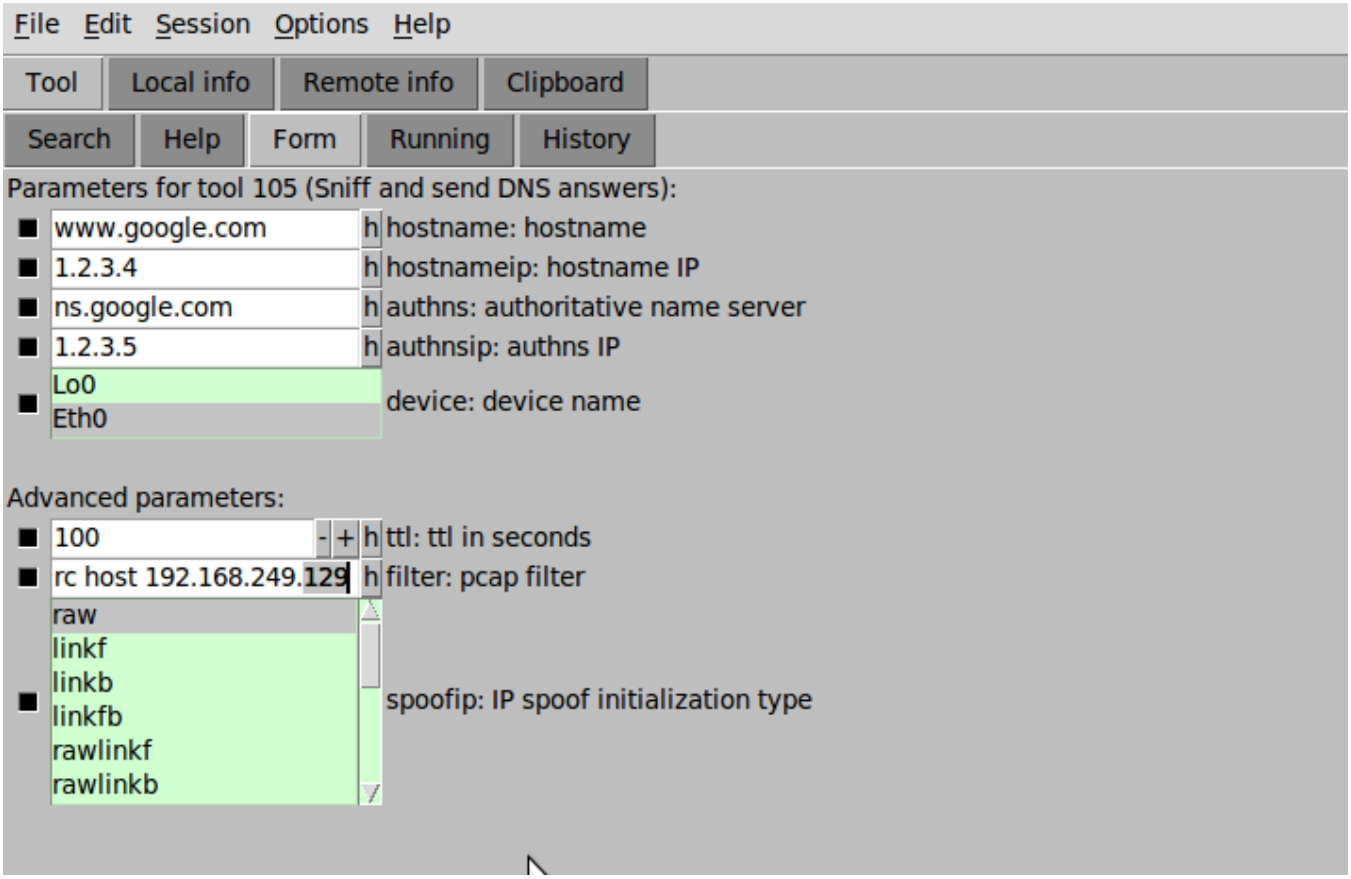

Figure 25: Netwag tool form

(v) Then I started the Netwag tool and ran dig command on user machine. The response of dig command has spoofed Ip address of www.google.com, which means that that the attack was successful and the DNS server cache has been poisoned.

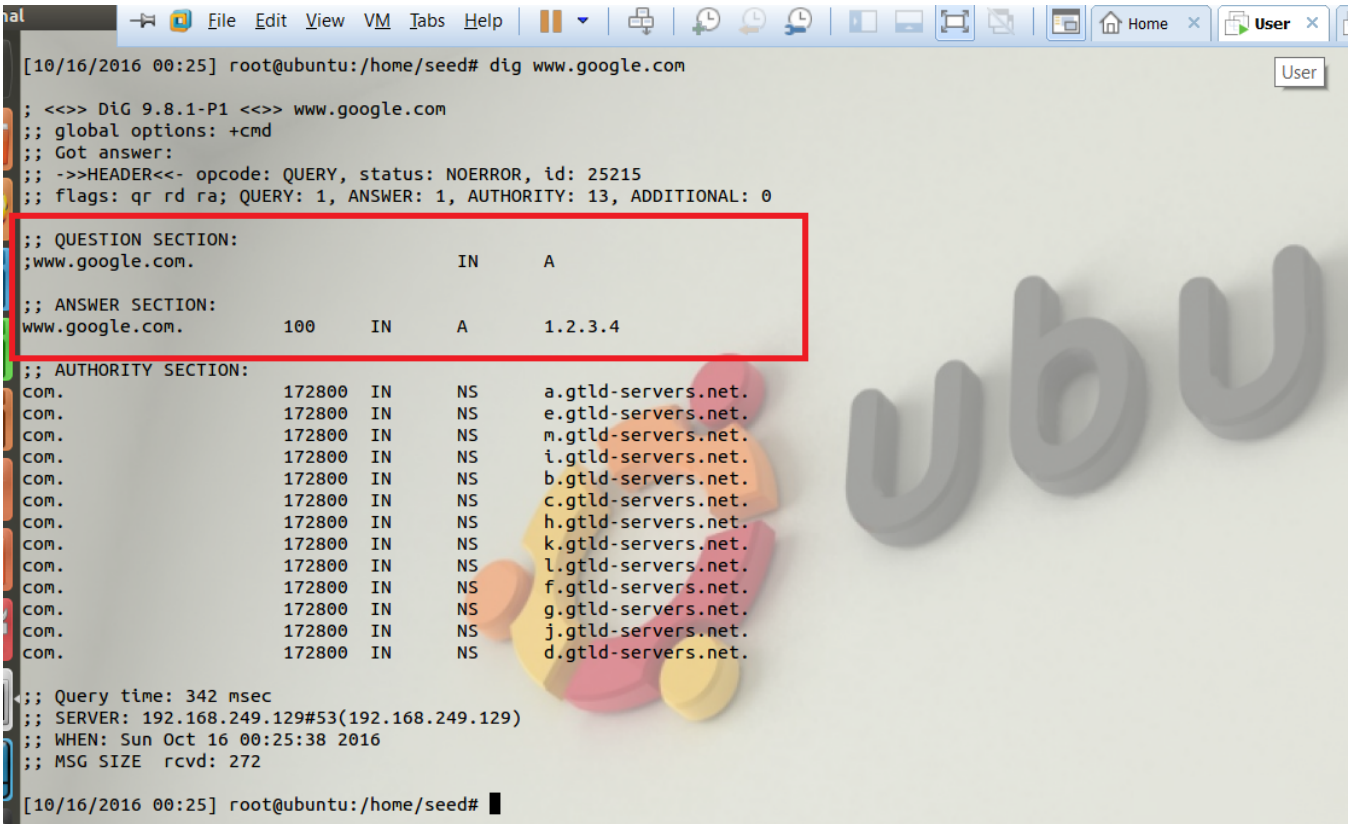

Figure 26: Spoofed Ip of www.google.com

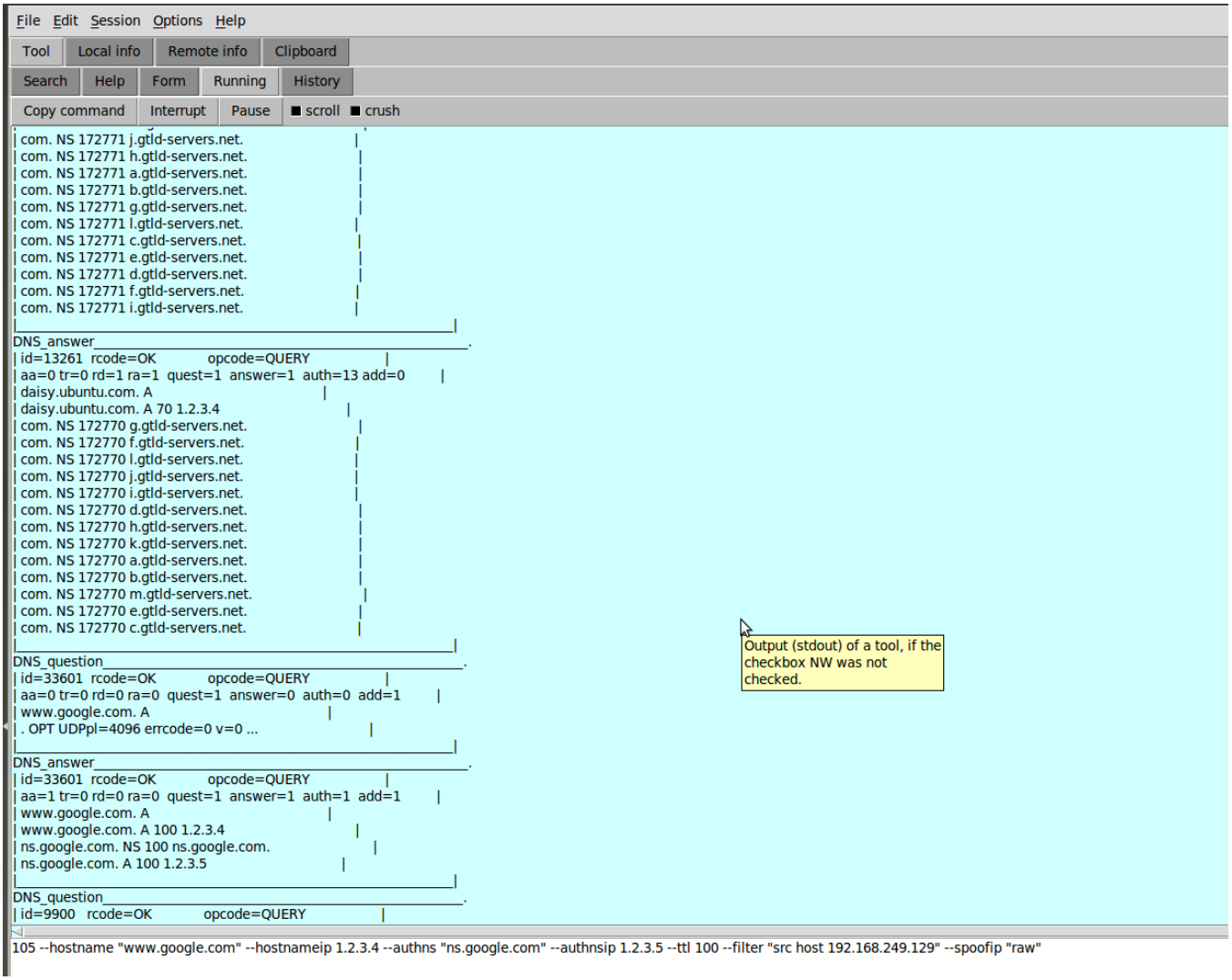

Figure 27: Spoofed Packet sent by Netwag tool

(vi) Wireshark running at the Attacker captures the request-response packets and it can be seen that the Transaction Id of Request and Spoofed Response are same. And this packet reached

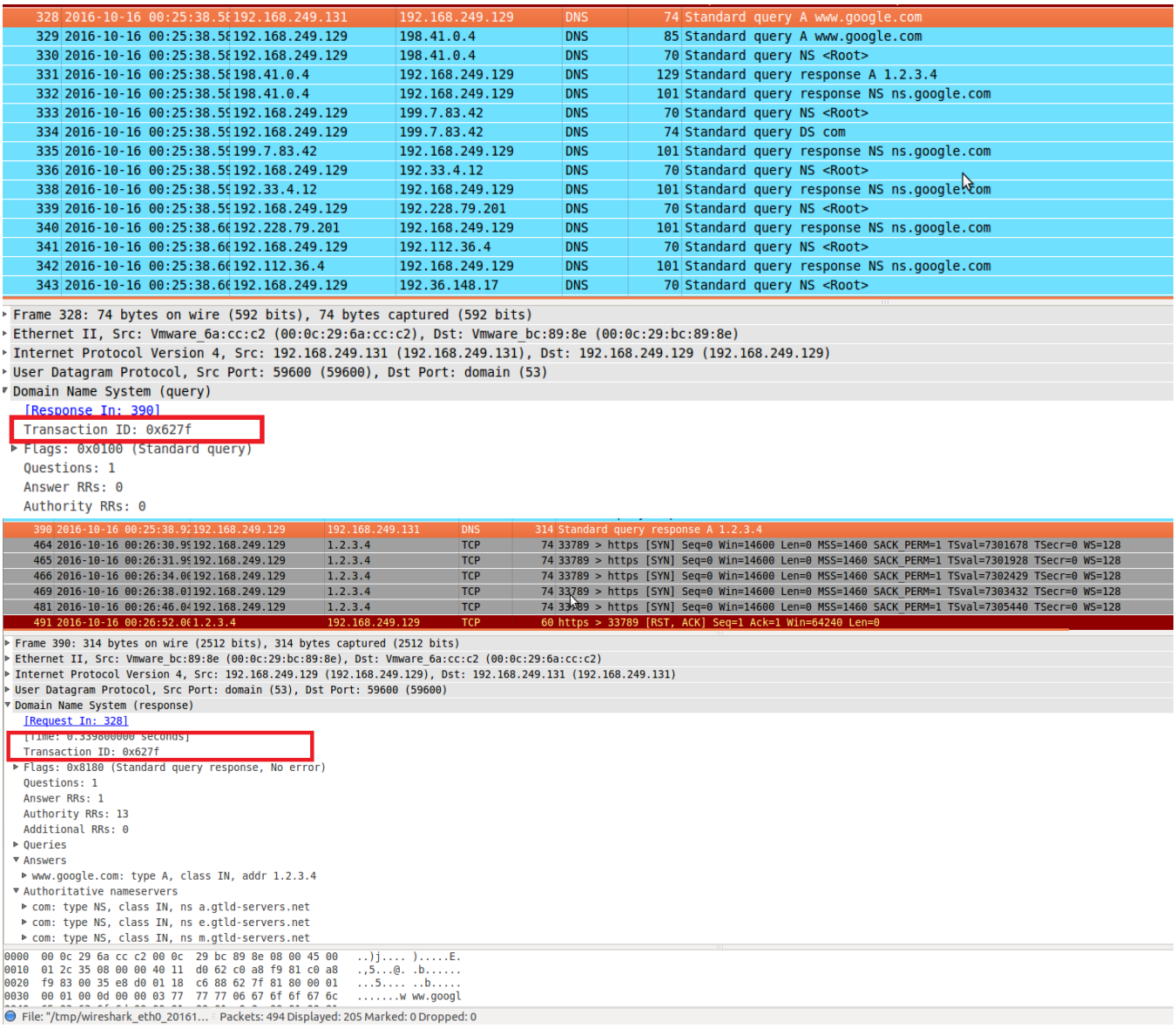

Figure 28: Spoofed packet Transaction Id is same as legitimate response

(vii) The DNS cache can be dumped in dump.db file using the command ”sudo rndc dumpdb -cache”. The dump.db file contains the entry ”www.google.com 1.2.3.4” and the ”ns.google.com 1.2.3.5”. These are the rogue entries which I have included in the file using Netwag tool.

Figure 29: Dump.db file content contains the rogue entries of google.com

(viii) I run the dig command again in DNS server and the output is 1.2.3.4 for www.google.com. This confirms that the spoofed Ip(1.2.3.4) for www.google.com is stored in the cache of server and it will remain in the cache till TTL expires.

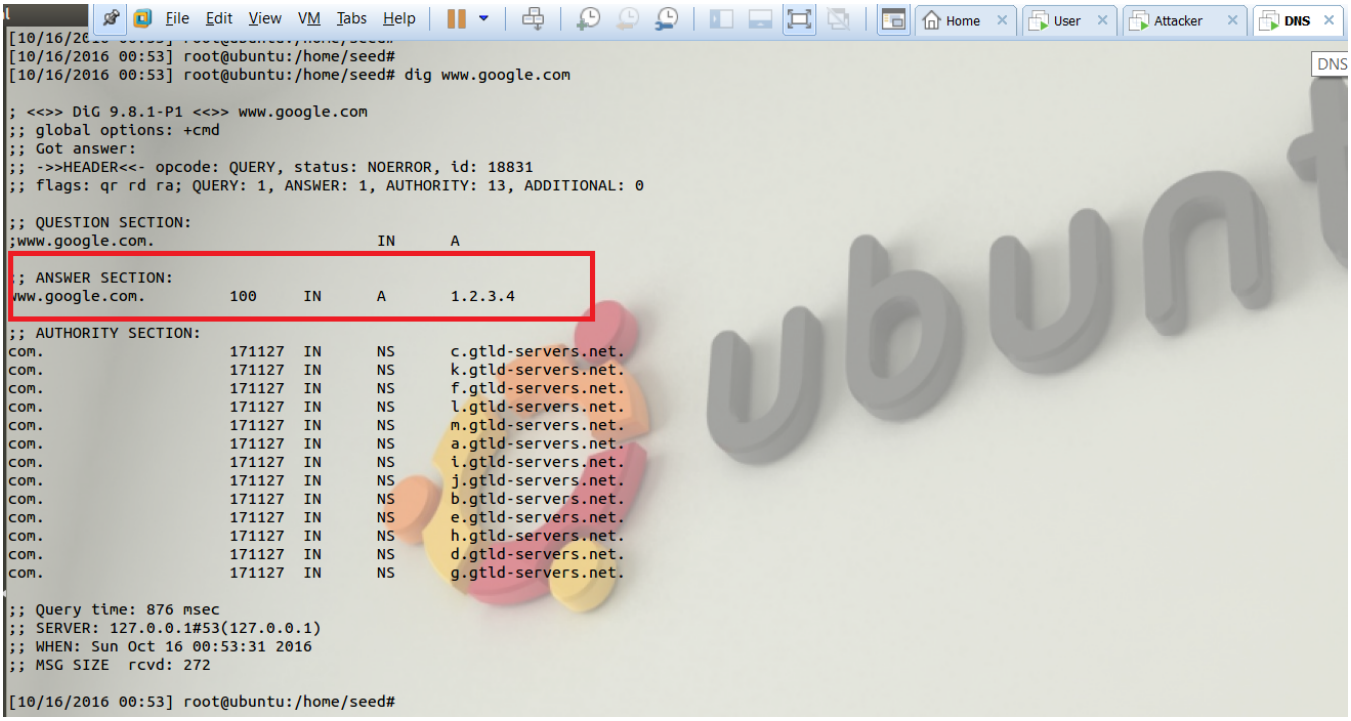

Figure 30: Cache Poisoning of DNS Server

(C) Explain how a real-world attacker could leverage this sort of attack.
A real-world attacker can use this attack to make a rogue Ip entry in DNS which would redirect user

unknowingly to a spoofed website. And then the attacker could extract sensitive information like bank passwords, credit card number, SSN number etc. from user to breach the security.

(D) Discuss whether you feel this is a viable attack in the real world. What limitations about this attack contribute to your answer?
Once a legitimate DNS record is saved in the DNS server, it stays there for a long time. The maximum TTL(Time to live) of a record is 2 days. So, DNS caching acts like a friend for making this attack harder. Suppose the user queries for a domain name from server. The server will query root server only when there is no record in its directory. If there is an entry of record, the local DNS will simply reply back the existing record to the user. So, to make this attack successful, the attacker must know the TTL of the legitimate record in the DNS, which is mostly unknown to the attacker. Also, if the local DNS is in a VPN or closed network, then either attacker must be an insider or it should first compromise an insider to make this attack work.
The consequences of this attack are very serious as many clients will be affected by this attack. However, the above points made it clear that the odds for a successful Server-Level Response Spoofing is very low, thus this attack is less viable in the real-world.

6 Kaminsky Attack

(A) Attack Description:
Kaminsky attack poisons the cache of DNS server even when a legitimate entry of a record is present in DNS. In Kaminsky attack, the attacker sends a query to DNS with a domain name which is not valid(like 1234.facebook.com). As the local DNS will not have any entry for this bogus domain name, it queries root DNS to resolve the name. The attacker then sends the spoofed packets to local DNS pretending to be root server with Ip address of 1234.facebook.com and also the updated ns.facebook.com and www.facebook.com Ip. The local DNS server takes these responses as legitimate and updates the Ip address of domain www.facebook.com.
In this attack, the DNS cache which helped us to thwart server-level response spoofing has been made void. Thus, this attack can be used for cache poisoning.

(B) Steps:

(i) For this attack, I have created an additional DNS2 which act as a response server by DNS1 for queries of ”www.dnsphishinglab.com”. The zone is created in second DNS server to avoid leaking of packets over the internet. To perform this attack in the closed environment, I have also disconnected my machine with the internet.

(ii) The Ip for ”www.dnsphishinglab.com” is 192.168.249.151 which has to be spoofed into another Ip address. The user machine requests local DNS for Ip of the domain, but as we have applied forwarder in local DNS, the local DNS asks the second DNS to resolve the query. The second DNS then responds back to local DNS which in turn responds back to user machine with Ip address.

```
[10/16/2016 21:07] root@ubuntu:/home/seed/Desktop/Edited File# nslookup www.dnsphishinglab.com
Server:         192.168.249.129
Address:        192.168.249.129#53

Name:   www.dnsphishinglab.com
Address: 192.168.249.151

[10/16/2016 21:11] root@ubuntu:/home/seed/Desktop/Edited File#
```

Figure 31: Nslookup from User Machine for ”www.dnsphishing.com”

98	2016-10-16 21:20:23.3…	192.168.249.131	192.168.249.129	DNS	82	Standard query A www.dnsphishinglab.com
101	2016-10-16 21:20:23.3…	192.168.249.129	192.168.249.132	DNS	93	Standard query A www.dnsphishinglab.com
102	2016-10-16 21:20:23.3…	192.168.249.132	192.168.249.129	DNS	142	Standard query response A 192.168.249.151
103	2016-10-16 21:20:23.3…	192.168.249.129	192.168.249.131	DNS	131	Standard query response A 192.168.249.151

Figure 32: DNS request-response packet sniffed using Wireshark

(iii) The pacgen tool is used to generate packets. There are 3 header files in pacgen - Ip configuration, Ethernet configuration, and UDP configuration. These files are used to design the structure of packet. The Ip address and MAC address of all the machines are included in the packet.

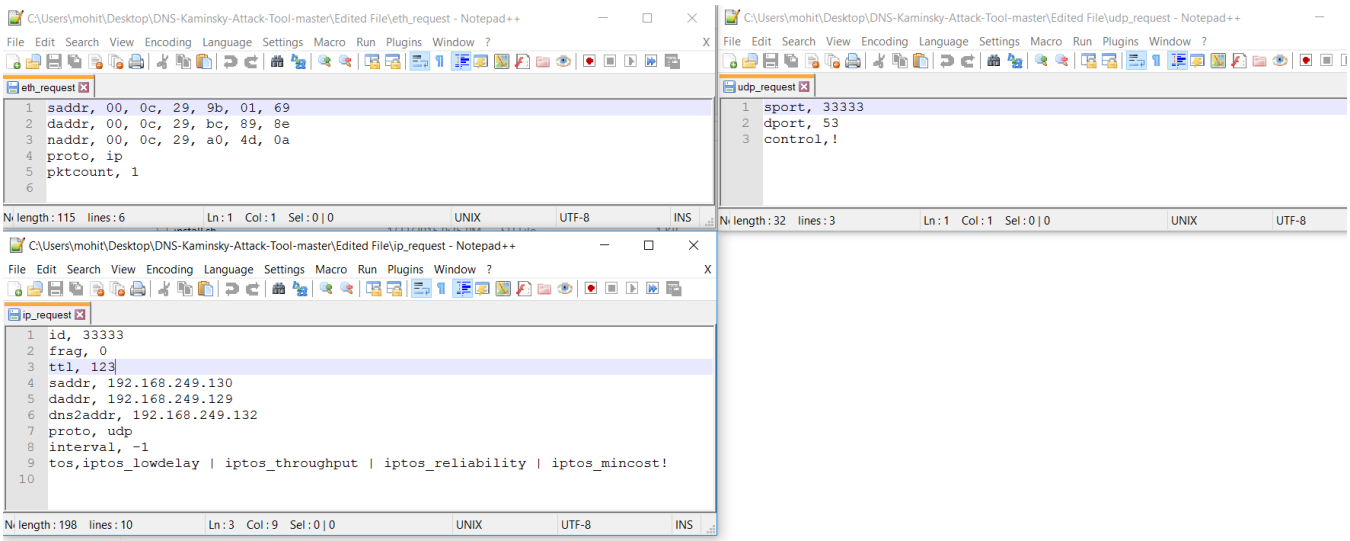

Figure 33: Pacgen Header Files

(iv) The pacgen code is modified to accomplish Kaminsky attack. Also, the Payload file used in by the pacgen is generated according to our need. Two payloads files are generated - Request Payload(payload_ query) and Response Payload(payload_ attack). The request payload contains the DNS request which is sent by a client to the local DNS server and the Response payload contains the spoofed DNS response which is sent by the attacker to the local DNS prior the legitimate response of second DNS reaches the local DNS. The payload is created by capturing the genuine requests and response from Wireshark. By using the Follow UDP stream, we can save the packet in the raw format. Then using the Hex Editor, the payload is modified. The spoofed Ip address of “www.dnsphishinglab.com”, “ns.dnsphishinglab.com” and the fake domain is added in the payload.

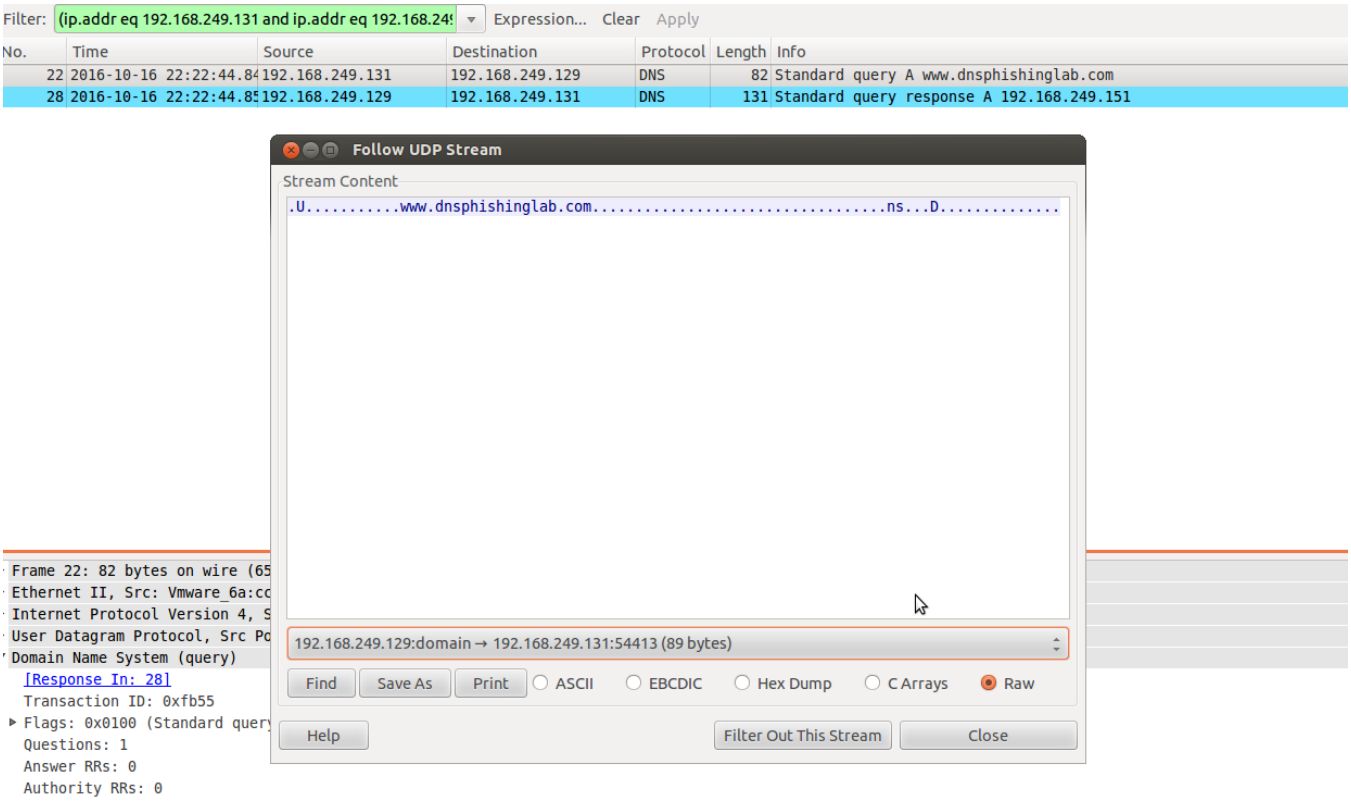

Figure 34: Follow UDP stream used to save DNS payload

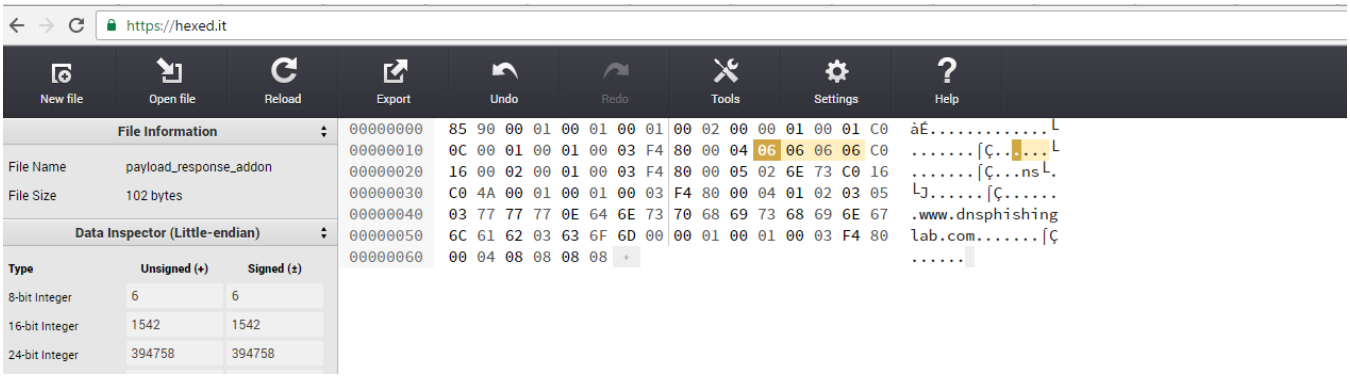

Figure 35: Hex Editor used to modify payload

(v) There are 2 sections which are implemented in pacgen. The first section generates spoofed domains whose RR does not exist in local DNS, thus ensuring that the local DNS will query the second DNS.
The second section sends spoofed DNS responses to local DNS server. The source host of these responses is the second DNS server and the destination is local DNS.
The response sent to local DNS contains the rogue Ip of fake subdomain and additional records containing Ip address of name server and original website. In my case, I have sent 6.6.6.6 Ip for Spoof-740.dnsphishinglab.com and in additional records I have set Ip 16.17.18.19 for ns.dnsphishinglab.com and 33.34.35.36 for www.dnsphishinglab.com. For successful attacks,

these entries replace the entries already present in the cache.

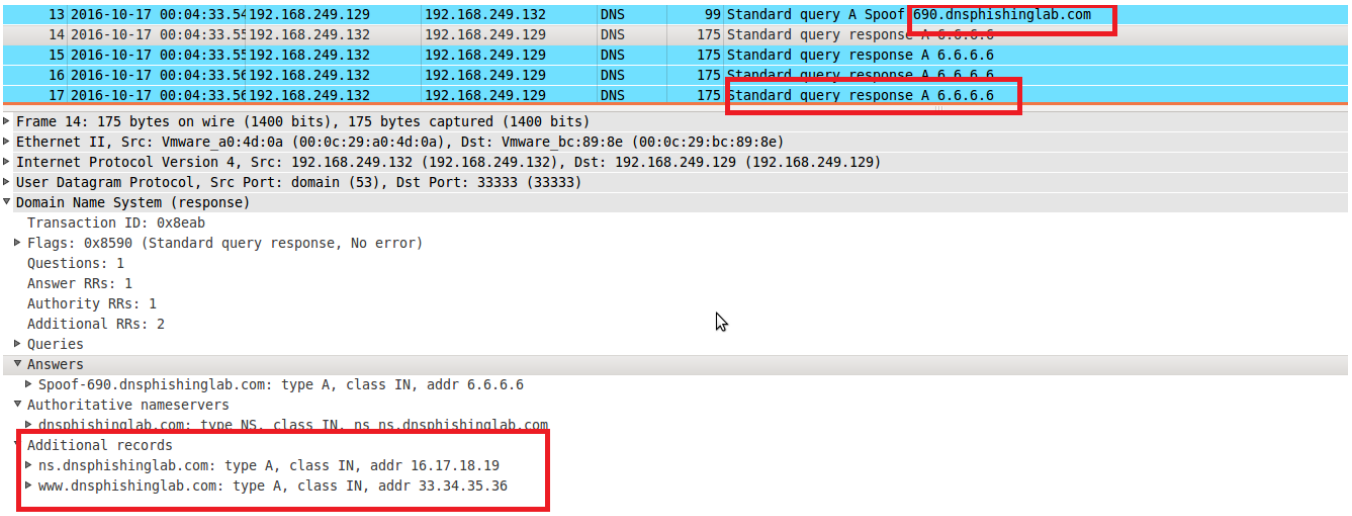

Figure 36: Spoofed Domain Request sent by user to local DNS. The spoofed DNS response to local DNS contains rogue IP - 6.6.6.4 and additional records

(vi) Thousands of spoofed packets are sent as response to local DNS for every subdomain query initiated by user. And when the Transaction Id of spoofed packet is same as the Transaction Id of legitimate response and the spoofed packet reaches prior to legitimate response, the spoofed packet is accepted by local DNS server and the entry in cache is updated. An entry of Spoof-982.phishinglabattack.com (subdomain randomly generated by the code) has an entry in dump.db of local DNS.

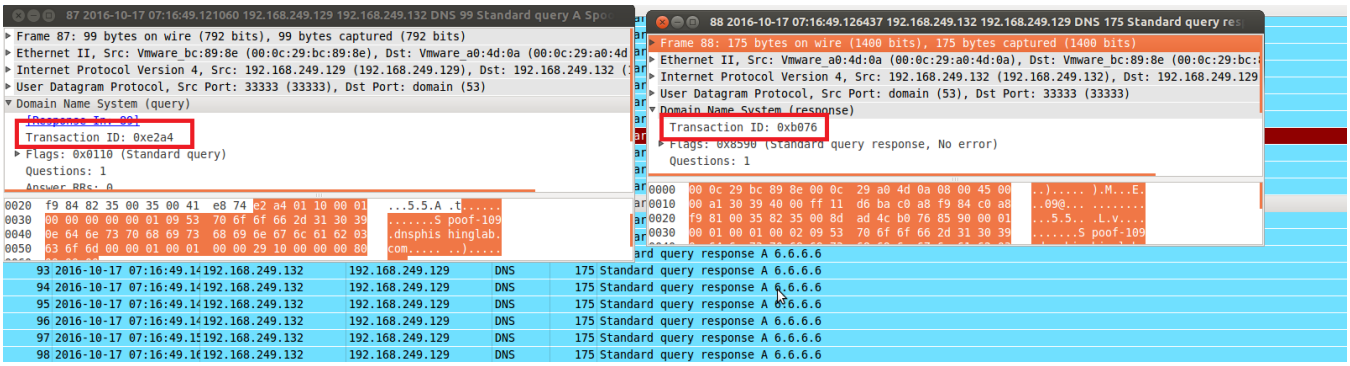

Figure 37: The left window contains the packet sent by local DNS to the second DNS with Transaction Id 0xe2a4 and the right window contains the spoofed packet sent by attacker to local DNS with Transaction Id 0xb076. When both transaction Id is same, the attack is successful

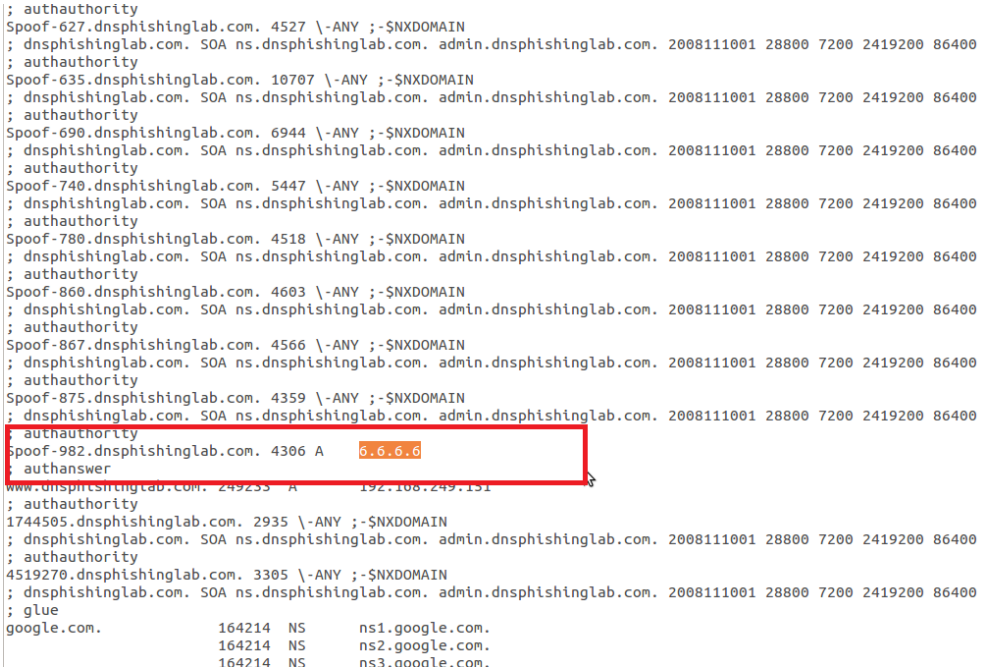

Figure 38: Successful entry of spoofed domain in cache of local DNS

(C) How successful was your attack tool? Did you have to run it several times? Why? What could you do to increase its level of success?
I tried this attack many times. The success rate of attack is low as the legitimate response from server comes prior the spoofed packets reached DNS. However, the attack successfully made an entry in cache of server after running various iterations. The result of cache is seen from dump.db file. To increase the level of success, I ran a loop which quickly sent several spoofed packets to DNS with different Transaction Id. As every machine is in local LAN, the response it swift. So, to make this attack possible, I included artificial latency in response of second DNS server by running the command - " sudo tc qdisc add dev eth6 root delay 100ms".
The DNS query port is also set to 33333 for ease of attack. In real world this query port is also random.

(D) Prevention from leakage to packets to the internet.
For prevention of packets to the internet I have included Second DNS server as the forwarder for First DNS server, All the requests for "www.dnsphishinglab.com" have been forwarded to this Server. Also, I have disconnected the internet connection of my machine to ensure that no packet should be leaked to internet.

(E) Explain how a real-world attacker could leverage this sort of attack.
This is a very serious attack as it can make cache entry of DNS void. The real world attacker can use this attack to replace DNS Resource Records with the faked RR which would send rouge IP address to client. These rogue addresses are of phished websites, from which the attacker can ask the user to provide sensitive information like bank passwords, credit card number, social networking sites password and much more. So, this attack can be used to compromise user credentials.

(F) Discuss whether you feel this is a viable attack in the real world.
Yes, Kaminsky attack is viable and a very dangerous attack. As I have demonstrated that the cache of server had been modified using this attack, the same can be accomplished by attacker by sending

bogus responses to local DNS server. And, once the server cache has been poisoned, it will remain in cache for a long time and could affect a lot of clients. Although, it takes a lot of time for a successful attack, but it is achievable bu using multiple bot-nets and using multiple subdomains.
This was the reason that when Kaminsky attack was discovered, it was not made public immediately. The attack was made public after patching maximum routers for six months. Even today, a lot of attackers use this attack to compromise security.