

计算机网络安全实验二

1 DNS协议漏洞利用实验

1.1 docker使用

1.1.1 建立实验环境

1.1.2 docker常用指令

1.2 设置本地 DNS 服务器

1.2.1 任务1 配置用户计算机

1.2.2 任务2 设置本地DNS服务器

1.2.3 任务 3: 在本地 DNS 服务器中建一个区域

1.3 本地DNS攻击

1.3.1 任务 4: 修改主机文件

1.3.2 任务 5: 直接欺骗用户响应

1.3.3 任务 6: DNS 缓存中毒攻击

1.3.4 任务 7: DNS 缓存中毒: 针对授权区域部分

1.3.5 任务 8: 针对另一个域

1.3.6 任务 9: 针对附加部分

1.4 远程 DNS 攻击实验

1.4.1 配置本地 DNS 服务器 Apollo

1.4.2 远程缓存中毒

1.4.3 结果验证

计算机网络安全实验二

1 DNS协议漏洞利用实验

1.1 docker使用

1.1.1 建立实验环境

普通用户: *seed* 密码:*dees* 超级用户: *root* 密码: *seedubuntu*

Network(bridge): 172.17.0.0/16:

*server*是已经创建好的, 如果没有, 就按照创建*dns*的方式创建。

创建*dns*:

```
sudo docker run -it --name=dns --hostname=dns "seedubuntu" /bin/bash
```

我的ip:

```
Attacker: 172.17.0.1 # 也就是虚拟机seed@VM  
dns: 172.17.0.4  
user: 172.17.0.2 # 就是实验一创建的用户 docker
```

1.1.2 docker常用指令

打开或停止*HostM*:

```
sudo docker start/stop HostM
```

把*HostM*映射到*bash*中:

```
sudo docker exec -it HostM /bin/bash
```

查看当前*docker*有哪些:

```
sudo docker ps -a
```

关闭防火墙:

```
sudo iptables -F
```

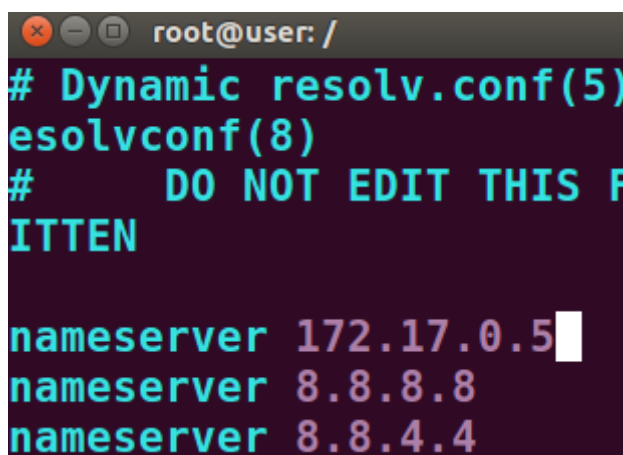
主机和容器之间拷贝数据:

```
sudo docker cp 容器名称:路径 主机路径  
sudo docker cp 主机路径 容器名称:路径
```

1.2 设置本地 DNS 服务器

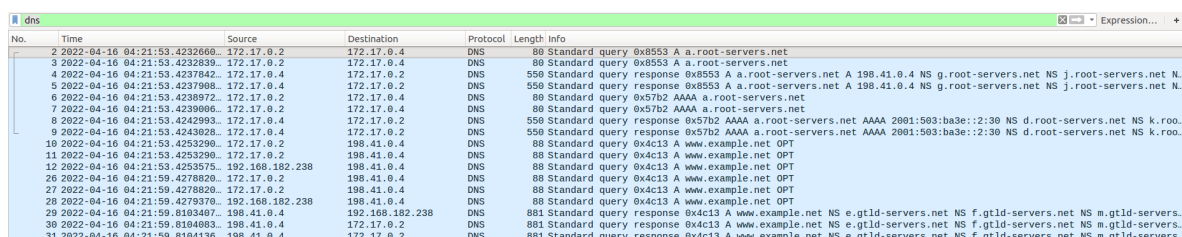
1.2.1 任务1 配置用户计算机

在用户机上，我们需要使用172.17.0.5作为本地 DNS 服务器（默认情况下，DNS 服务器程序已在 SEED VM 中运行）。这是通过更改用户计算机的解析程序配置文件（`/etc/resolv.conf`）来实现的，因此将服务器172.17.0.5添加为文件中的第一个 `nameserver` 条目，即此服务器将用作主 DNS 服务器，如下图所示。



```
root@user: /  
# Dynamic resolv.conf(5)  
resolvconf(8)  
# DO NOT EDIT THIS FILE - IT IS AUTOMATICALLY  
GENERATED  
nameserver 172.17.0.5  
nameserver 8.8.8.8  
nameserver 8.8.4.4
```

完成配置用户计算机之后，使用 `dig` 命令获取 IP 地址。如下图所示（此时已经配置完成本地 DNS 服务器如任务2，并且已经运行过一次 `dig` 指令）。



No.	Time	Source	Destination	Protocol	Length	Info
2	2022-04-16 04:21:53.4232660	172.17.0.2	172.17.0.4	DNS	80	Standard query 0x8553 A a.root-servers.net
3	2022-04-16 04:21:53.4232839	172.17.0.2	172.17.0.4	DNS	80	Standard query 0x8553 A a.root-servers.net
4	2022-04-16 04:21:53.4237842	172.17.0.4	172.17.0.2	DNS	550	Standard query response 0x8553 A a.root-servers.net A 198.41.0.4 NS g.root-servers.net NS j.root-servers.net NS k.root-servers.net
5	2022-04-16 04:21:53.4237988	172.17.0.4	172.17.0.2	DNS	550	Standard query response 0x8553 A a.root-servers.net A 198.41.0.4 NS g.root-servers.net NS j.root-servers.net NS k.root-servers.net
6	2022-04-16 04:21:53.4238972	172.17.0.2	172.17.0.4	DNS	80	Standard query 0x57b2 AAAA a.root-servers.net
7	2022-04-16 04:21:53.4239066	172.17.0.2	172.17.0.4	DNS	80	Standard query 0x57b2 AAAA a.root-servers.net
8	2022-04-16 04:21:53.4242993	172.17.0.4	172.17.0.2	DNS	550	Standard query response 0x57b2 AAAA a.root-servers.net AAAA 2001:503:ba3e::2:30 NS d.root-servers.net NS k.root-servers.net
9	2022-04-16 04:21:53.4243028	172.17.0.4	172.17.0.2	DNS	550	Standard query response 0x57b2 AAAA a.root-servers.net AAAA 2001:503:ba3e::2:30 NS d.root-servers.net NS k.root-servers.net
10	2022-04-16 04:21:53.4253298	172.17.0.2	198.41.0.4	DNS	80	Standard query 0x4c13 A www.example.net OPT
11	2022-04-16 04:21:53.4253390	172.17.0.2	198.41.0.4	DNS	80	Standard query 0x4c13 A www.example.net OPT
12	2022-04-16 04:21:53.4253575	192.168.182.238	198.41.0.4	DNS	80	Standard query 0x4c13 A www.example.net OPT
26	2022-04-16 04:21:59.4278820	172.17.0.2	198.41.0.4	DNS	80	Standard query 0x4c13 A www.example.net OPT
27	2022-04-16 04:21:59.4278820	172.17.0.2	198.41.0.4	DNS	80	Standard query 0x4c13 A www.example.net OPT
28	2022-04-16 04:21:59.4279370	192.168.182.238	198.41.0.4	DNS	80	Standard query 0x4c13 A www.example.net OPT
29	2022-04-16 04:21:59.8103407	198.41.0.4	192.168.182.238	DNS	881	Standard query response 0x4c13 A www.example.net NS e.gtld-servers.net NS f.gtld-servers.net NS m.gtld-servers.net NS p.gtld-servers.net NS s.gtld-servers.net NS t.gtld-servers.net NS u.gtld-servers.net NS v.gtld-servers.net NS w.gtld-servers.net NS x.gtld-servers.net NS y.gtld-servers.net NS z.gtld-servers.net
30	2022-04-16 04:21:59.8104893	198.41.0.4	172.17.0.2	DNS	881	Standard query response 0x4c13 A www.example.net NS e.gtld-servers.net NS f.gtld-servers.net NS m.gtld-servers.net NS p.gtld-servers.net NS s.gtld-servers.net NS t.gtld-servers.net NS u.gtld-servers.net NS v.gtld-servers.net NS w.gtld-servers.net NS x.gtld-servers.net NS y.gtld-servers.net NS z.gtld-servers.net
31	2022-04-16 04:21:59.8104136	198.41.0.4	172.17.0.2	DNS	881	Standard query response 0x4c13 A www.example.net NS e.gtld-servers.net NS f.gtld-servers.net NS m.gtld-servers.net NS p.gtld-servers.net NS s.gtld-servers.net NS t.gtld-servers.net NS u.gtld-servers.net NS v.gtld-servers.net NS w.gtld-servers.net NS x.gtld-servers.net NS y.gtld-servers.net NS z.gtld-servers.net

可见，用户机向 DNS 服务器发送请求，然后收到回应。

1.2.2 任务2 设置本地 DNS 服务器

将缓存的内容转储到上面指定的文件，在 `dns` 中运行：

```
sudo rndc dumpdb -cache
```

显示报错：`rndc: connect failed: 127.0.0.1#953: connection refused`，搜索发现本条报错意思是 `rndc` 在当前是不可用的。因此，需要先启动 DNS 服务器：

```
sudo service bind9 restart
```

编辑 `/etc/bind/named.conf.options`，向选项块添加 `dump-file "/var/cache/bind/dump.db";`，并注释掉 `dnssec-validation auto;`（关闭 DNSSEC），如下图所示，打开的时候已经配置好了。

```
// dnssec-validation auto;  
dnssec-enable no;  
dump-file "/var/cache/bind/dump.db";  
auth-nxdomain no;          # conform to RFC1035  
query-source port          33333;
```

然后再运行:

```
sudo rndc dumpdb -cache # Dump the cache to the sepcified file  
sudo rndc flush # Flush the DNS cache
```

再运行:

```
sudo named -d 3 -f -g # 启动DNS服务器
```

在用户机上运行:

```
ping www.baidu.com
```

在Wireshark上查看ping命令触发的DNS查询。

我运行了两次ping命令，第一次解析时产生大量DNS查询和回应报文，第二次数量明显减少，说明第二次有一部分是从DNS缓存中得到的。

The image shows a Wireshark packet capture of DNS traffic. The top pane displays a list of packets, with the first ping showing many DNS queries and responses, and the second ping showing fewer, indicating cache hits. The middle pane shows the details of the selected packet, including the domain name system protocol. The bottom pane shows the raw packet data in hexadecimal and ASCII. A terminal window in the bottom right shows the output of the ping command, indicating successful connectivity to www.baidu.com.

1.2.3 任务3: 在本地DNS服务器中建一个区域

1. 创建区域: 在dns中编辑 /etc/bind/named.conf :

```
zone "example.com" {  
    type master;  
    file "/etc/bind/example.com.db";  
};  
  
zone "0.168.192.in-addr.arpa" {  
    type master;  
    file "/etc/bind/192.168.0.db";  
};
```

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "example.com" {
    type master;
    file "/etc/bind/example.com.db";
};
zone "0.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/192.168.0.db";
};
```

19,1

Bot

把文件从主机中移动到docker中：

```
sudo docker cp 192.168.0.db dns:/etc/bind/
sudo docker cp example.com.db dns:/etc/bind/
```

2. 设置正向查找区域文件(即example.com.db)。

3. 设置反向查找区域文件(即192.168.0.db)。

```
root@dns: /
root@dns:/# cat /etc/bind/192.168.0.db
$TTL 3D
@      IN      SOA      ns.example.com. admin.example.com. (
                        2008111001
                        8H
                        2H
                        4W
                        1D)
@      IN      NS       ns.example.com.

101    IN      PTR      www.example.com.
102    IN      PTR      mail.example.com.
10     IN      PTR      ns.example.com.
```

4. 重新启动BIND服务器并进行测试：

重启BIND 9服务，然后观察Wireshark：

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-04-16 04:44:42.8658930	172.17.0.2	172.17.0.4	ICMP	60	Standard query 0x0e6f A www.example.com OPT
2	2022-04-16 04:44:42.8659158	172.17.0.2	172.17.0.4	DNS	88	Standard query 0x0e6f A www.example.com OPT
3	2022-04-16 04:44:42.8659177	172.17.0.2	172.17.0.4	DNS	88	Standard query 0x0e6f A www.example.com OPT
4	2022-04-16 04:44:42.8662010	172.17.0.4	172.17.0.2	DNS	137	Standard query response 0x0e6f A www.example.com A 192.168.0.101 NS ns.example.com A 192.168.0.10 OPT
5	2022-04-16 04:44:42.8662061	172.17.0.4	172.17.0.2	DNS	137	Standard query response 0x0e6f A www.example.com A 192.168.0.101 NS ns.example.com A 192.168.0.10 OPT

在Wireshark中可以看到，来回的报文减少了很多。并且返回www.example.com的地址是192.168.0.101，和之前的不同。这是因为该域名已经在我们本地的DNS服务器中设置托管，不会为该域中的主机名发送DNS查询。

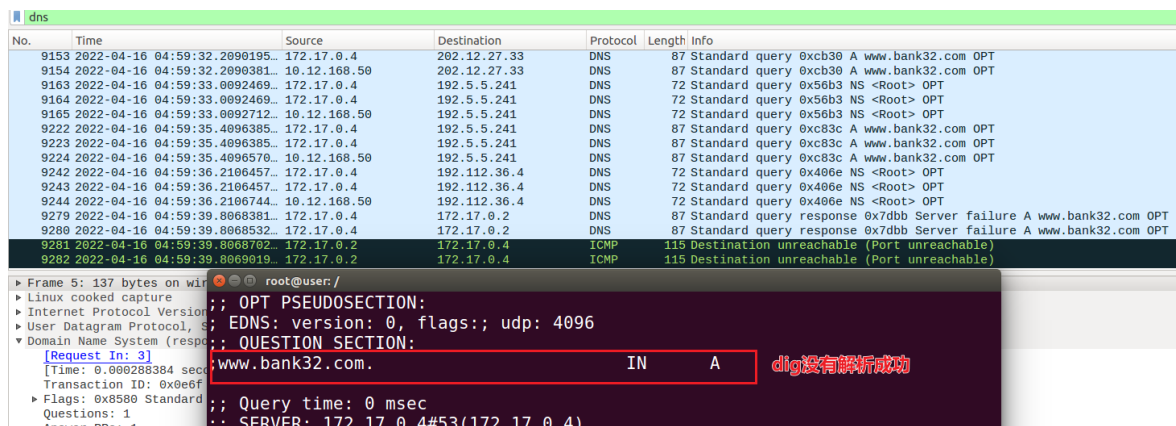
1.3 本地DNS攻击

1.3.1 任务 4：修改主机文件

修改/etc/hosts文件，添加：

1.2.3.4 www.bank32.com

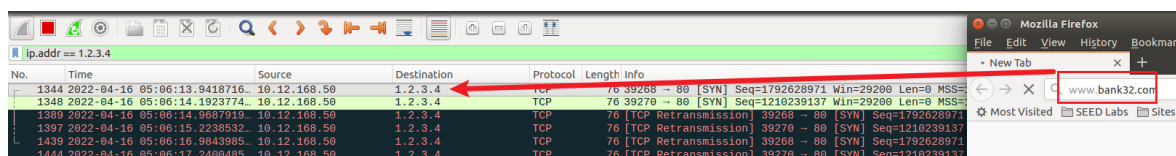
用dig命令测试结果，发现修改主机文件确实不影响对www.bank32.com文件解析，如下图所示：



用ping命令测试修改结果，确实影响了，如下图所示：

```
root@user:/# ping www.bank32.com
PING www.bank32.com (1.2.3.4) 56(84) bytes of data.
```

用Web浏览器测试结果，这个需要到seed@VM中检验。因此把seed@VM的/etc/hosts也修改一下，测试结果如下。



如上图所示，解析的DesIP被修改成1.2.3.4。

1.3.2 任务 5：直接欺骗用户响应

1.3.3 任务 6：DNS 缓存中毒攻击

1.3.4 任务 7：DNS 缓存中毒：针对授权区域部分

1.3.5 任务 8：针对另一个域

1.3.6 任务 9：针对附加部分

1.4 远程 DNS 攻击实验

1.4.1 配置本地 DNS 服务器 Apollo

1.4.2 远程缓存中毒

1.4.3 结果验证