南開大學

网络技术与应用课程实验报告

实验 3：通过编程获取 IP 地址与 MAC 地址的对应关系

学院：网络空间安全学院

专业：信息安全-法学

学号：2111954

姓名：许积君

目录

一、 实验要求

通过编程获取 IP 地址与 MAC 地址的对应关系实验，要求如下:

1. 在 IP 数据报捕获与分析编程实验的基础上，学习 NPcap 的数据包发送方法。
2. 通过 NPcap 编程，获取 IP 地址与 MAC 地址的映射关系。
3. 程序要具有输入 IP 地址，显示输入 IP 地址与获取的 MAC 地址对应关系界面。界面可以是命令行界面，也可以是图形界面，但应以简单明了的方式在屏幕上显示。
4. 编写的程序应结构清晰，具有较好的可读性。

二、 设计思路

(一) 伪造 ARP 获取 MAC 地址的原理

1. ARP 协议工作原理：ARP 用于将 IP 地址映射到 MAC 地址，以便在局域网中正确路由数据包。当设备想要与局域网中的其他设备通信时，它会向目标设备发送 ARP 请求，询问目标设备的 MAC 地址。目标设备会回应一个包含其 MAC 地址的 ARP 响应。
2. ARP 缓存：设备通常会维护一个 ARP 缓存表，记录 IP 地址到 MAC 地址的映射关系。这样，当设备再次需要与同一设备通信时，就不必再次发送 ARP 请求，而可以直接使用缓存中的映射关系。
3. ARP 欺骗的原理：攻击者发送虚假的 ARP 响应或请求，欺骗其他设备，使它们相信攻击者的 MAC 地址是与某个特定 IP 地址相关联的正确地址。攻击者可以通过这种方式截取、篡改或监视网络流量。
   (a) ARP 响应攻击：攻击者发送虚假的 ARP 响应，将自己的 MAC 地址伪装成目标 IP 地址的 MAC 地址。其他设备将此虚假信息存储在其 ARP 缓存中，将流量发送到攻击者而不是真正的目标设备。
   (b) ARP 请求攻击：攻击者发送虚假的 ARP 请求，询问目标 IP 地址对应的 MAC 地址。其他设备收到请求后，会回复并将其 ARP 缓存中的信息更新为攻击者的 MAC 地址。
4. 获取目标 IP 的 MAC 地址：通过成功进行 ARP 欺骗，攻击者就能够获取目标 IP 地址的 MAC 地址。这使得攻击者能够中间截取通信，进行数据嗅探，或者进行其他恶意活动。

(二) 实验设计步骤

由于本次实验是要获取本机或者相同网段下其他 IP 对应的 MAC 地址, 所以采用伪造 ARP 请求包的方法

1. 获取网络接口列表，选择需要捕获 MAC 地址的网卡
2. 伪造 ARP 请求包，设置目标 IP，在请求包中我们分为两部分
   (a) 第一个是获取本机的 MAC 地址，此时我们随意构造合法的 IP 和 MAC 地址

(b) 第二个是获取远程的 MAC 地址，为了正确接收到 ARP 响应包，我们将第一步获得的本机 IP 和 MAC 地址赋值给 ARP 请求包的相关数据。当然，我们也可以选择伪造 IP 和 MAC 地址，网关接收到组建的 ARP 请求后会由网关发出一个 ARP 请求，找到本机发送网卡的真实 IP 和 MAC 地址，从而进一步获取远程主机的 MAC。

3. 用打开的网卡广播 ARP 请求包

4. 监听打开的网卡的流量，并筛选 ARP 保温，捕获对应的 ARP 响应报文，在相应报文中就可以找到 MAC 地址

三、 关键代码

(一) 获取设备列表，打印网卡信息和 IP

这部分代码与实验 2 没有区别

```
pcap_if_t* allAdapters;// 所有网卡设备保存
    char errbuf[PCAP_ERRBUF_SIZE];//错误缓冲区，大小为256
    int index = 1;
    // 获取本地机器设备列表，并打印
    //pcap.h抓包库中的函数
    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &allAdapters, errbuf)
        != -1)
    {   /* 打印网卡信息列表 */
        pcap_if_t* ptr;
        for (ptr = allAdapters; ptr != NULL; ptr = ptr->next)
        {
            if (ptr->description)
                printf("ID %d  Name: %s \n", index, ptr->description);
            index++;

            for (pcap_addr_t* addr = ptr->addresses; addr != nullptr; addr =
                addr->next)
            {
                if (addr->addr->sa_family == AF_INET)
                {

                    cout << "  IP地址: \t\t" << inet_ntoa(((struct
                        sockaddr_in*)(addr->addr))->sin_addr) << endl;
                    cout << "  网络掩码: \t\t" << inet_ntoa(((struct
                        sockaddr_in*)(addr->netmask))->sin_addr) << endl;
                    cout << "  广播地址: \t\t" << inet_ntoa(((struct
                        sockaddr_in*)(addr->broadaddr))->sin_addr) << endl;
                }
            }
            cout << "==========================================================" << endl;
        }
        index--;
    }
```

```
29      else
30      {
31          printf("Error in pcap_findalldevs_ex: %s\n", errbuf);
32      }
33
34      if (index == 0)
35      {
36          cout << "没有找到接口" << endl;
37      }
38        while (1)
39      {
40          //打开想要监控的网卡
41          cout << "请输入想要监控的网卡的ID" << endl;
42          int num;
43          cin >> num;
44          if (num == 0)
45          {
46              cout << "结束获取, 关闭进程" << endl;
47              pcap_freealldevs(allAdapters);
48              return 0;
49          }
50          while (num < 1 || num > index)
51          {
52              cout << "不存在该设备, 请重新输入合适的ID" << endl;
53              cin >> num;
54          }
55          int i = 0;
56          pcap_if_t* ptr;
57          for (ptr = allAdapters, i = 0; i < num - 1; ptr = ptr->next, i++);
58
59          pcap_t* pcap_handle = pcap_open_live(ptr->name,     //设备名称
60              65536,
                                                //包长度最大值 65536允许整个包在所有
                    mac电脑上被捕获
61              PCAP_OPENFLAG_PROMISCUOUS,                                  /* 混
                    杂模式*/
62              1000,
                                                //读超时为1秒
63              errbuf);
                                        //错误缓冲池;//打开网络适配器, 捕捉实例,是
                    pcap_open返回的对象
64          if (pcap_handle == NULL)
65          {
66              cout << "无法打开该网卡接口" << endl;
67              pcap_freealldevs(allAdapters);
68              exit(0);
69          }
70          cout << "正在监听" << ptr->description << endl;
```

```
71          ……
72      }
```

```
ID 1? Name: Network adapter 'WAN Miniport (Network Monitor)' on local host
==============================================================
ID 2? Name: Network adapter 'WAN Miniport (IPv6)' on local host
==============================================================
ID 3? Name: Network adapter 'WAN Miniport (IP)' on local host
==============================================================
ID 4? Name: Network adapter 'MediaTek Wi-Fi 6 MT7921 Wireless LAN Card' on local host
  IP地址:               10.136.83.112
  网络掩码:             0.128.255.255
  广播地址:             255.255.83.112
==============================================================
ID 5? Name: Network adapter 'VMware Virtual Ethernet Adapter for VMnet8' on local host
  IP地址:               192.168.230.1
  网络掩码:             0.255.255.255
  广播地址:             255.168.230.1
==============================================================
ID 6? Name: Network adapter 'VMware Virtual Ethernet Adapter for VMnet1' on local host
  IP地址:               192.168.190.1
  网络掩码:             0.255.255.255
  广播地址:             255.168.190.1
==============================================================
ID 7? Name: Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host
  IP地址:               169.254.168.40
  网络掩码:             0.0.255.255
  广播地址:             255.255.168.40
==============================================================
```

图 1: 打印网卡信息

（二） 报文格式

为了伪造合法的 ARP 请求包，我们需要根据 ARP 报文格式正确设置，这里包括一个头部存储了 MAC 地址和帧类型，ARP 请求包的内容还包括一些硬件信息，MAC 地址和 IP 地址

```
typedef struct FrameHeader_t
{
    uint8_t DesMAC[6];  //  目的地址
    uint8_t SrcMAC[6];//    源网地址
    uint16_t FrameType;//   帧类型
}FrameHeader_t;

typedef struct ARPFrame_t
{//IP首部
    FrameHeader_t FrameHeader;
    uint16_t Hardware_Type;      //硬件类型
    uint16_t Protocol_Type;      //协议类型
    uint8_t HALen;               //硬件地址长度
    uint8_t PALen;               //协议地址长度
    uint16_t Operation;          //操作码
    uint8_t SendHA[6];//发送方MAC地址
    uint32_t SendIP;//发送方IP地址
    uint8_t RecvHA[6];//接收方MAC地址
    uint32_t RecvIP;//接收方IP地址
}ARPFrame_t;
```

（三） 报文内容

由于我们不知道目标地址的 MAC，采取广播的形式，将 DesMAC 设置为 0xffffffffff，将 SrcMAC 设置为本机的 MAC，但是这里我们采取伪造的地址，后面的硬件设置为 ARP 请求包

的固定内容

```
//报文内容
ARPFrame_t ARPFrame;
for (int i = 0; i < 6; i++)
{
    ARPFrame.FrameHeader.DesMAC[i] = 0xff;          //广播
    ARPFrame.FrameHeader.SrcMAC[i] = 0x0f;
    ARPFrame.SendHA[i] = 0x0f;
    ARPFrame.RecvHA[i] = 0;//表示目的地址未知
}

ARPFrame.FrameHeader.FrameType = htons(0x806);   //帧类型为ARP
ARPFrame.Hardware_Type = htons(0x0001);          //硬件类型为以太网
ARPFrame.Protocol_Type = htons(0x0800);          //协议类型为IP
ARPFrame.HALen = 6;                              //硬件地址长度为6
ARPFrame.PALen = 4;                              //协议地址长为4
ARPFrame.Operation = htons(0x0001);              //操作为ARP请求

uint32_t SerIP = ARPFrame.SendIP = htonl(0x00000000);//设置为任意IP地址
```

（四） 获取本机 IP 的 MAC 地址

1. 将目标 ARP 的目标 IP 设置为打开网卡的 IP
2. 发送 ARP 请求包
3. 不断接收数据包，直到找到接收到的数据包的源 IP 为目标 IP（在这里就是本机的 IP），目标 IP 为本机 IP，此时就是我们想要的 ARP 的相应包，跳出循环，这时候接收到的数据包的 SendHA 存储的就是我们想要的目的地址

```
uint32_t RecvIP; //接收方的IP
ARPFrame_t* RecvPacket;
for (pcap_addr_t*addr = ptr->addresses; addr != NULL; addr = addr->next)
{
    if (addr->addr->sa_family == AF_INET)
    {
        RecvIP = ARPFrame.RecvIP = inet_addr(inet_ntoa(((struct sockaddr_in*)
            (addr->addr))->sin_addr)); //把接收方的IP设置为打开的网卡的IP
    }
}


// 向以太网广播ARP请求
struct pcap_pkthdr* RecvHeader;
const u_char* RecvData;

if (pcap_sendpacket(pcap_handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) !=
    0)
```

```
{
    pcap_freealldevs(allAdapters);
    throw - 7;
} //发送失败的处理
else
{
    while (true) {
        pcap_next_ex(pcap_handle, &RecvHeader, &RecvData);
        RecvPacket = (ARPFrame_t*)RecvData;

        if (LocalIP == RecvPacket->SendIP && RecvIP == RecvPacket->RecvIP) {
            // 如果是期望的ARP响应包
            continue;  // 继续循环等待下一个包
        }

        // 根据网卡号寻找IP地址，并输出IP地址与MAC地址映射关系
        if (LocalIP == RecvPacket->RecvIP && RecvIP == RecvPacket->SendIP) {
            cout << "IP地址与MAC地址的对应关系如下: " << endl << "IP: ";
            print_IP(RecvPacket->SendIP);
            cout << "MAC: ";
            print_MAC(RecvPacket->SendHA);
            cout << endl;
            break;  // 结束循环，已经找到并输出了对应关系
        }
    }
}
```

（五） 获取远程 IP 的 MAC 地址

由于不同网段下的 IP 消息互不可达，所以在输入 IP 时要选取同网段下的 IP，并且要保证网络被分配，采用虚拟机同网段下的 IP 进行了测试

与获取本机 IP 的 MAC 地址的操作相似，将目标地址设置为输入的 IP，值得注意的是，我们在上一步返回的数据包中包含着本机的 IP 和 MAC 地址，此时就可以把 ARP 包中伪造的 IP 和 MAC 改为接收包中的本机 IP 和 MAC 地址，再进行发送，筛选相应包的原理与上一步相同

```
char IP[16];
cout << "==========请输入远程目的IP地址==========" << endl;
cin >> IP;
RecvIP = ARPFrame.RecvIP = inet_addr(IP);

LocalIP = ARPFrame.SendIP = RecvPacket->SendIP;
for (i = 0; i < 6; i++) {
    ARPFrame.SendHA[i] = ARPFrame.FrameHeader.SrcMAC[i] = RecvPacket->SendHA[
        i];
}

if (pcap_sendpacket(pcap_handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) !=
    0) {
```

```
12      cout << "发送失败！" << endl;
13      pcap_freealldevs(allAdapters);
14      throw - 6;
15  }
16  else {
17      while (true) {
18          pcap_next_ex(pcap_handle, &RecvHeader, &RecvData);
19          RecvPacket = (ARPFrame_t*)RecvData;
20
21          if (LocalIP == RecvPacket->SendIP && RecvIP == RecvPacket->RecvIP) {
22              // 如果是期望的ARP响应包
23              continue;  // 继续循环等待下一个包
24          }
25
26          if (LocalIP == RecvPacket->RecvIP && RecvIP == RecvPacket->SendIP) {
27              cout << "IP地址与MAC地址的对应关系如下：" << endl << "IP：";
28              print_IP(RecvPacket->SendIP);
29              cout << "MAC: ";
30              print_MAC(RecvPacket->SendHA);
31              cout << endl;
32              break;  // 结束循环，已经找到并输出了对应关系
33          }
34      }
35  }
```

四、 获取局域网下所有 IP 和 MAC

在实验的第一阶段已经获得了本机的 IP 和 MAC 地址，可以找到网卡所在的网段，通过遍历的方式发送 ARP 请求包。先清空 MAC 和 IP 的映射表，遍历 IP 地址，同时要检测这个 IP 是否存在在线设备，如果不存在就不会返回 ARP 响应包，最后通过接收到的 ARP 获得对应的 MAC，就可以获得 IP 和 MAC 的完整映射表

五、 运行结果

(一) 本机 MAC 地址

图 2: 本机 MAC 获取

（二） 远程 MAC 地址

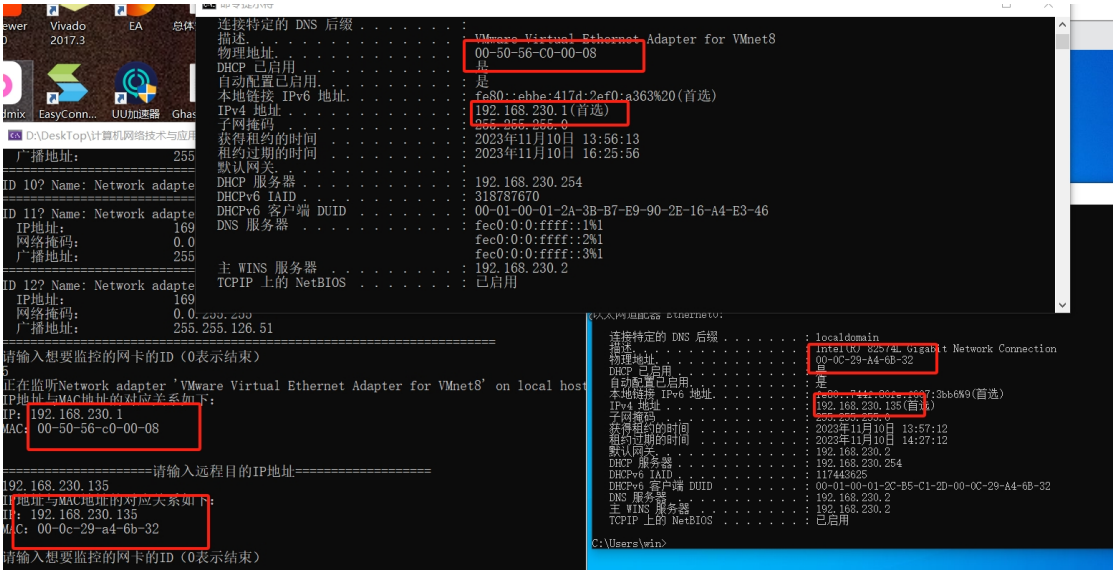
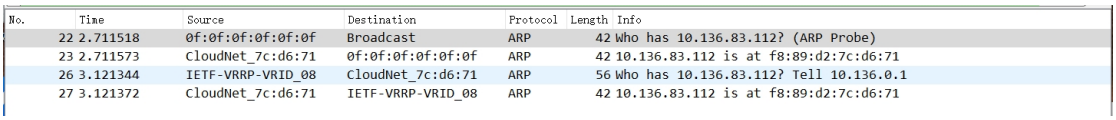


图 3: 远程 MAC 获取

（三） wireshark 抓包结果



No.	Time	Source	Destination	Protocol	Length	Info
22	2.711518	0f:0f:0f:0f:0f:0f	Broadcast	ARP	42	Who has 10.136.83.112? (ARP Probe)
23	2.711573	CloudNet_7c:d6:71	0f:0f:0f:0f:0f:0f	ARP	42	10.136.83.112 is at f8:89:d2:7c:d6:71
26	3.121344	IETF-VRRP-VRID_08	CloudNet_7c:d6:71	ARP	56	Who has 10.136.83.112? Tell 10.136.0.1
27	3.121372	CloudNet_7c:d6:71	IETF-VRRP-VRID_08	ARP	42	10.136.83.112 is at f8:89:d2:7c:d6:71

图 4: wireshark 抓包

六、 GitHub 链接

代码提交链接