

哈爾濱工業大學

# 网络安全实验报告

题    目 捕包软件的使用与实现

专    业 计算机科学与技术

学    号 7203610316

学    生 符兴

指导教师 王彦

## 一、实验目的

理解捕包程序捕包过程，可以自己编程捕包并从数据包中解析出需要的信息。

## 二、实验内容

1. 熟练使用 sniffer 或 wireshark 软件，对协议进行还原（能够找访问网页的四元组）；只需要写报告，不需要在实验课检查。

2. 利用 libpcap 或 winpcap 进行编程，能够对本机的数据包进行捕获分析（比如将本机所有数据包的四元组写到指定文件），按照自己的设想撰写需求分析和详细设计。（实验课检查程序）

## 三、实验过程

### （一）使用 wireshark 软件对协议进行还原

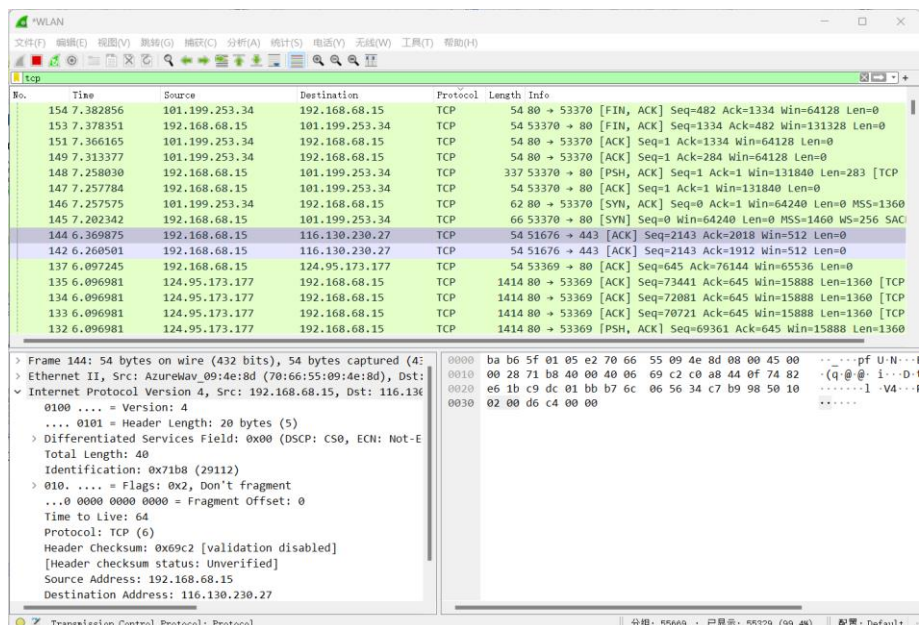
#### 实验基本信息：

实验环境：Windows11 x64

WireShark2.6.4

#### 1. 捕包并分析四元组

##### (1) TCP 分析



截图中这个 TCP 数据包，源 IP 为 192.168.68.15，目的 IP 为 116.130.230.27，源端口为 51676，目的端口为 443。

分析：

**以太网头部：**

前 6 个字节 ba:b6:5f:01:05:e2 为目的主机 MAC，往后 6 个字节 70:66:55:09:4e:8d 为源主机 MAC，

往后 2 个字节为上层协议，0x0800 表示 IPv4 协议；

**以太网头部结束，现在是 ip 头部：**

往后 1 个字节 0x45 表示 IP 版本为 4，头部长度为 5，往后 1 个字节为区分服务，0x00 表示默认，往后 2 个字节为总长度，0x0028 = 40，

往后 2 个字节为 id，值为 0x71b8，

往后 2 个字节为标志位+片偏移，值为 0x4000，

往后 1 个字节为 TTL，值为 0x40 = 64，

往后 1 个字节为上层协议，0x06 表示 TCP 协议，

往后 2 个字节为头部校验和，值为 0x69c2，

往后 4 个字节为源 ip 地址，转换为 10 进制就是 192.168.68.15，

往后 4 个字节为目的 ip 地址，转换为 10 进制就是 116.130.230.27；

**ip 头部结束，现在是 TCP 头部：**

往后 2 个字节为源端口 0xc9dc = 51676，

往后 2 个字节为目的端口 0x01bb = 443，

往后 4 个字节为 seq = 0xb76c0656，

往后 4 个字节为 ack = 0x34c7b998，

往后 1 个字节中前四位为头部长度为 5，

往后 1 个字节中的后 6 位为标志位，分别为 010000，

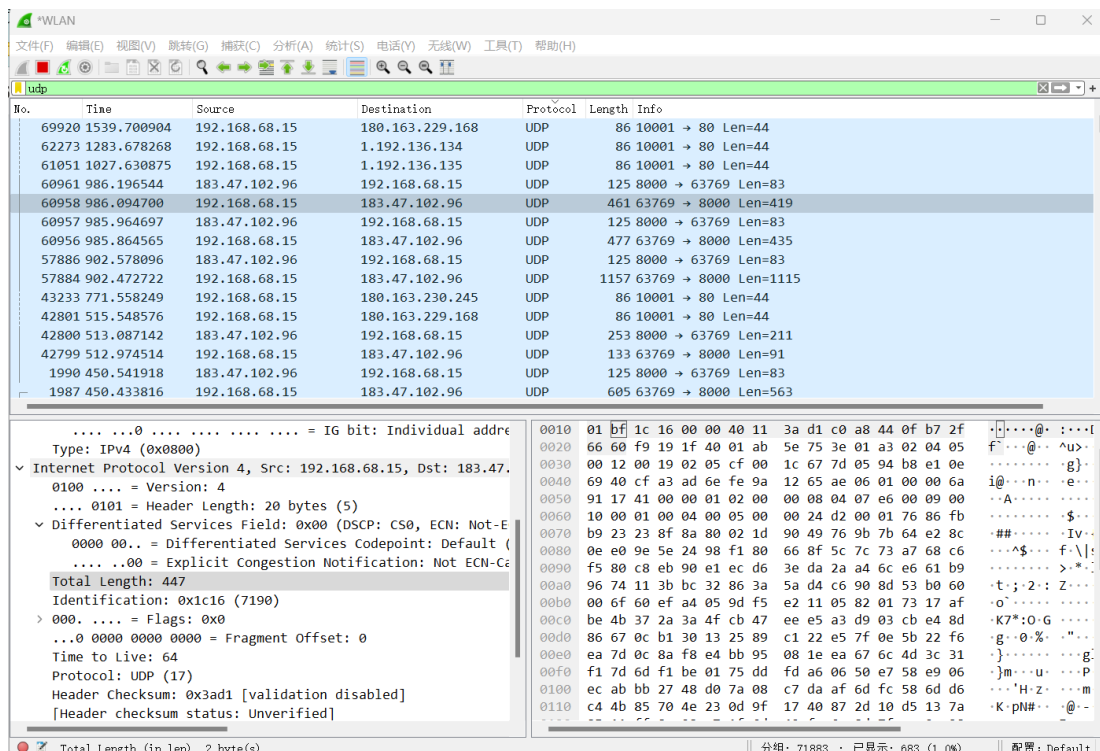
往后 2 个字节为窗口大小 0x0200 = 512，

往后 2 个字节为校验和 0xd6c4，

往后 2 个字节为紧急指针 0x0000；

**TCP 头部结束，接下来是数据。**

## (2) UDP 分析



截图中这个 UDP 数据包，源 IP 为 192.168.68.15，目的 IP 为 183.47.102.96，源端口为 63769，目的端口为 8000。

分析：

### 以太网头部：

前 6 个字节 ba:b6:5f:01:05:e2 为目的主机 MAC，往后 6 个字节 70:66:55:09:4e:8d 为源主机 MAC，

往后 2 个字节为上层协议，0x0800 表示 IPv4 协议；

### 以太网头部结束，现在是 ip 头部：

往后 1 个字节 0x45 表示 IP 版本为 4，头部长度为 5，

往后 1 个字节为区分服务，0x00 表示默认，

往后 2 个字节为总长度，0x01bf = 447，

往后 2 个字节为 id，值为 0x1c16，

往后 2 个字节为标志位+片偏移，值为 0x0000，

往后 1 个字节为 TTL，值为 0x40 = 64，

往后 1 个字节为上层协议，0x11 表示 UDP 协议，

往后 2 个字节为头部校验和，值为 0x3ad1，

往后 4 个字节为源 ip 地址，转换为 10 进制就是 192.168.68.15，  
往后 4 个字节为目的 ip 地址，转换为 10 进制就是 116.130.230.27；

**ip 头部结束，现在是 UDP 头部：**

往后 2 个字节为源端口 0xf917 = 63769，  
往后 2 个字节为目的端口 0x1f40 = 8000，  
往后 2 个字节为长度 0x01ab = 427  
往后 2 个字节为校验和 0x5e75

**UDP 头部结束，往后是数据部分。**

## (二) 利用 libpcap 编写捕包软件

实验环境：Ubuntu16.04 x64

编程语言：C 语言

### 1. 需求分析

本程序需要运用 libpcap 来捕获本机数据包，并获取数据包中的四元组，将其展示给用户。

程序功能：

- (1)捕获本机数据包（可以自定义过滤条件）；
- (2)逐层解析数据包，获得 IPv4 数据包的源 ip、目的 ip、源端口、目的端口；
- (3)将上述四元组写入文件（每次运行程序都新生成一个文件）。

### 2. 环境配置

直接终端执行安装：

```
Sudo apt install libpcap-dev
```

### 3. 数据结构设计

由于是逐层解析以太网数据帧，所以需要准备至少三种数据结构：以太网数据帧头、IPv4 数据报头、传输层报文头。具体如下：

数据结构的定义原则：1 字节数据定义为 u\_char，2 字节数据定义为 u\_short，其他 2 的倍数字节的数据（MAC 地址和 IP 地址）定义为 u\_char 数组（TCP 的序列号和 ack 定义为 u\_int，因为它们的表现形式就是一个数字，但地址我们通常是一个一个字节分开解析的）。

### (1) 以太网数据帧头

```
1 // 以太网数据帧头部结构体
2 struct ethernet
3 {
4     u_char eth_dsthost[ETHERNET_ADDR_LEN]; // 以太网MAC目的地址
5     u_char eth_srchoost[ETHERNET_ADDR_LEN]; // 以太网MAC源地址
6     u_short eth_type;                       // 协议类型
7 };
```

### (2) IPv4 数据报头

```
1 // IPv4报文头部结构体
2 struct ip
3 {
4     u_char ip_hlv;           // 版本号+头部长度
5     u_char ip_tos;           // 区分服务
6     u_short ip_len;          // IP数据报长度
7     u_short ip_id;           // 标识
8     u_short ip_off;          // 标志3位+片偏移13位
9     u_char ip_ttl;           // 生存时间
10    u_char ip_pro;            // 协议
11    u_short ip_checksum;      // 首部校验和
12    u_char ip_src[IP_ADDR_LEN]; // 源IP地址
13    u_char ip_dst[IP_ADDR_LEN]; // 目的IP地址
14 };
```

### (3) TCP 报文头

```
1 // TCP报文头
2 struct tcp
3 {
4     u_short tcp_srcport; // 源端口号
5     u_short tcp_dstport; // 目的端口号
6     u_int tcp_seq;        // 序列号
7     u_int tcp_ack;        // 确认号
8     u_char tcp_headlen;   // 4位头部长度+4位保留
9     u_char tcp_flag;      // 2保留位+6位标志位
10    u_short tcp_win;       // 窗口大小
11    u_short tcp_checksum;  // 校验和
12    u_short tcp_urp;       // 紧急指针
13 };
```

### (4) UDP 报文头

```
1 // UDP报文头部结构体
2 struct udp
3 {
4     u_short udp_srcport; // 源端口号
5     u_short udp_dstport; // 目的端口号
6     u_short udp_len;     // 总长度
7     u_short udp_checksum; // 校验和
8 };
```

## 4. 回调函数设计

(1)创建以太网帧头、IP 数据报帧头的数据结构，然后对捕获的数据报进行解析；其中，以太网帧头 14 字节，IP 数据报帧头根据首部长度的值乘以 4B 进行确定。其中，由于 IP 地址可能为 ipv6，所以使用 64 位进行存储网络地址转换后的结果。

```
// 解析以太网帧头
parseEthernet(&tmpE, datagram);
fprintf(filePtr, "src_mac: ");
for (int i = 0; i < 6; i++)
{
    fprintf(filePtr, " %02x", tmpE.eth_srchost[i]);
}
fprintf(filePtr, "\n");
fprintf(filePtr, "dst_mac: ");
for (int i = 0; i < 6; i++)
{
    fprintf(filePtr, " %02x", tmpE.eth_dsthost[i]);
}
fprintf(filePtr, "\n");
// 解析IP帧头
parseIp(&tmpIP, datagram);
char ip[64];
memset(&ip, 0, 64);
/// 输出的是点分表示
inet_ntop(AF_INET, &tmpIP.ip_src, ip, 64);
fprintf(filePtr, "src_ip: %s \n", ip);

memset(&ip, 0, 64);
inet_ntop(AF_INET, &tmpIP.ip_dst, ip, 64);
fprintf(filePtr, "dst_ip: %s \n", ip);
```

(2)在解析完 IP 数据报的首部信息后可以通过协议字段判断 IP 数据报往下报文的协议，其中程序只判断 TCP 和 UDP 的报文。TCP 的协议为 0x06，UDP 的协议为 0x11；获得协议号后根据相关的解析程序对其进行解析。

```
// 根据协议选择分析报文
if(tmpIP.ip_pro == 6){
    struct tcp tmpTCP;
    parseTcp(&tmpTCP, datagram);
    fprintf(filePtr, "src_port[TCP]: %u\n", tmpTCP.tcp_srcport);
    fprintf(filePtr, "dst_port[TCP]: %u\n", tmpTCP.tcp_dstport);
}else if(tmpIP.ip_pro == 17){
    struct udp tmpUDP;
    parseUdp(&tmpUDP, datagram);
    fprintf(filePtr, "src_port[UDP]: %u\n", tmpUDP.udp_srcport);
    fprintf(filePtr, "dst_port[UDP]: %u\n", tmpUDP.udp_dstport);
}

fprintf(filePtr, "\n");
```

## 5. 主函数设计

第一步：使用 `pcap_findalldevs()` 函数来获取网络设备。此步不用 `pcap_lookupdev()` 是因为官方并不推荐使用这个函数，有时第一个位置的网卡是一个虚拟网卡，用它进行下面的步骤会出现错误。

第二步：使用 `pcap_open_live()` 函数来获得捕包描述字，由于只需要捕获本机数据包所以设置为非混杂模式。

第三步：如果有过滤条件的话（作为程序运行参数读入），设置过滤条件。

第四步：生成本次捕包的 txt 文件（格式如“capture\_yyyy\_mm\_dd\_hh\_mm\_ss.txt”，capture 后是生成文件的时间），并写入过滤条件和标题栏。

第五步：使用 `pcap_loop()` 函数和回调函数 `ethernet_callback()` 来循环捕包。

## 6. 编译运行

使用下列命令编译：

```
gcc pcap.c -Wall -lpcap -o pcap
```

其中 -Wall 参数指打印所有警告信息，-lpcap 用来链接 pcap 库。使用下列命令运行：

```
sudo ./pcap
```

捕包程序在运行时需要 root 权限，否则无法正常打开。

## 四、实验结果

### （一）使用 wireshark 软件对协议进行还原

该部分已经在前面阐述完毕。

### （二）利用 libpcap 编写捕包软件

以下为捕包时程序和文件输出：



## (1) 设置 TCP 为过滤条件



## (2) 设置 UDP 为过滤条件



## 五、心得体会

1. 进一步掌握了以太网帧、IP 数据报、TCP 数据报以及 UDP 数据报的报文结构、各个字段的具体含义。
2. 深入学习使用 Wireshark 捕获数据包，并完成各层协议内容的分析。
3. 深入学习使用 libpcap 编写数据包的捕获程序，并将其结果输出到指定的文件当中，以便后续分析。