哈爾濱工業大學

网络安全实验报告

题　　目 基于 libnet 的程序设计

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

掌握 libnet 数据包的构造原理。

二、 实验内容

1.掌握 libnet 数据包的构造原理

2.编程实现基于 libnet 的数据包构造，结合前面实验给出验证过程。能够对源码进行解释。

三、实验过程

基于 libnet 的数据包构造

实验基本信息：

实验环境：Ubuntu 20.04 x64

编程语言：C 语言

1. 需求分析

需要使用 libnet 构造并发送一个数据包，并验证这个数据包被成功发送了。验证这一过程需要用到实验二中的捕包程序 pcap，将生成的数据包从虚拟机 B 发送到虚拟机 A，虚拟机 A 中的捕包程序会自动将其捕获，通过检查各项信息，证明捕获的数据包就是从虚拟机 B 此程序 mackPack 中发送的数据包。

追加：在虚拟机 A 中编写接收来自相应端口 udp 数据包的程序 recvUDP，验证该数据包确实可以被正确接收。

2. 程序结构

(1) makePack 程序的大致结构为：

① 初始化 libNet 句柄，其中需要设置好网卡接口。

```
// 初始化
    libnet_t* libNetHandle = libnet_init(LIBNET_LINK_ADV, "ens33", NULL);
    libnet_ptag_t pack = 0;
```

② 定义源、目的主机的 MAC 地址以及源、目的主机的 IP 地址和端口；其中从参数接收的 IP 地址字符串需要调用 libnet 中的函数将其转为网络字节序。

```
// 定义参数
    unsigned char srcMac[6] = {0x00, 0x0C, 0x29, 0xB0, 0xE2, 0xD0};
    unsigned char dstMac[6] = {0x00, 0x0C, 0x29, 0xAF, 0x6A, 0x59};

    char *srcIpStr = argv[1];
    int srcPort = atoi(argv[2]);
    char *dstIpStr = argv[3];
    int dstPort = atoi(argv[4]);
    printf("源IP: %s 目的IP: %s\n",srcIpStr, dstIpStr);
    printf("源端口: %d 目的端口:%d\n", srcPort, dstPort);
    unsigned long srcIp = libnet_name2addr4(libNetHandle, srcIpStr, LIBNET_RESOLVE);
    unsigned long dstIp = libnet_name2addr4(libNetHandle, dstIpStr, LIBNET_RESOLVE);
```

③ 设置 UDP 需要发送的数据。

④ 依照顺序依次构建数据包：UDP 数据包->IP 数据报->以太网帧。

```
// 构建UDP包
    pack = libnet_build_udp(
        srcPort,
        dstPort,
        8 + dataLen,
        0,
        sendData,
        dataLen,
        libNetHandle,
        0
    );
```

图 构建 UDP 数据包

```
// 构建IP数据报
    pack = libnet_build_ipv4(
        20 + 8 + dataLen,   // 包长度
        0,                  // 服务类型
        500,                // ip标识
        0,                  // 片偏移
        10,                 // 生存时间
        17,                 // UDP协议号
        0,                  // 校验和
        srcIp,
        dstIp,
        NULL,               // 负载，上面生成了UDP包
        0,                  // 长度
        libNetHandle,
        0
    );
```

图 构建 IP 数据报

```
// 构建以太网帧
    pack = libnet_build_ethernet(
        (uint8_t *)dstMac,
        (uint8_t *)srcMac,
        ETHERTYPE_IP,
        NULL,
        0,
        libNetHandle,
        0
    );
```

图 构建以太网帧

⑤ 发送数据

(2) recvUDP 程序结构：

① 创建 socket 句柄

② bind 连接

③ recvfrom()监听相应端口接收的 UDP 数据包。

3. 进一步验证

(1) 设置 makePack 中 UDP 所发送的数据为:" Data From B."；同时，在 recvUDP 中使用 strcmp()函数对接收到的数据进行判断。

```
// UDP接收数据
        char buffer[BUFFER_SIZE], strData[BUFFER_SIZE];
        memset(buffer, 0, BUFFER_SIZE);
        memset(strData,0, BUFFER_SIZE);
        recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)&client, &len);
        sprintf(strData,"Data From B.");
        printf("buff = %s\n", buffer);
        if(strcmp(buffer, strData) == 0){
            printf("正确接收了B发来的信息\n");
        }else{
            printf("未正确接收了B发来的信息\n");
        }
```

图 recvUDP 对数据进行判断

(2) 同时在目标主机上运行实验二所构建的 pcap 程序对 UDP 数据包进行捕获，查看其中的源、目的 IP 以及源、目的端口判断 makePack 程序所构建的数据包是否正确，并是否正确发送到目标主机。

四、实验结果

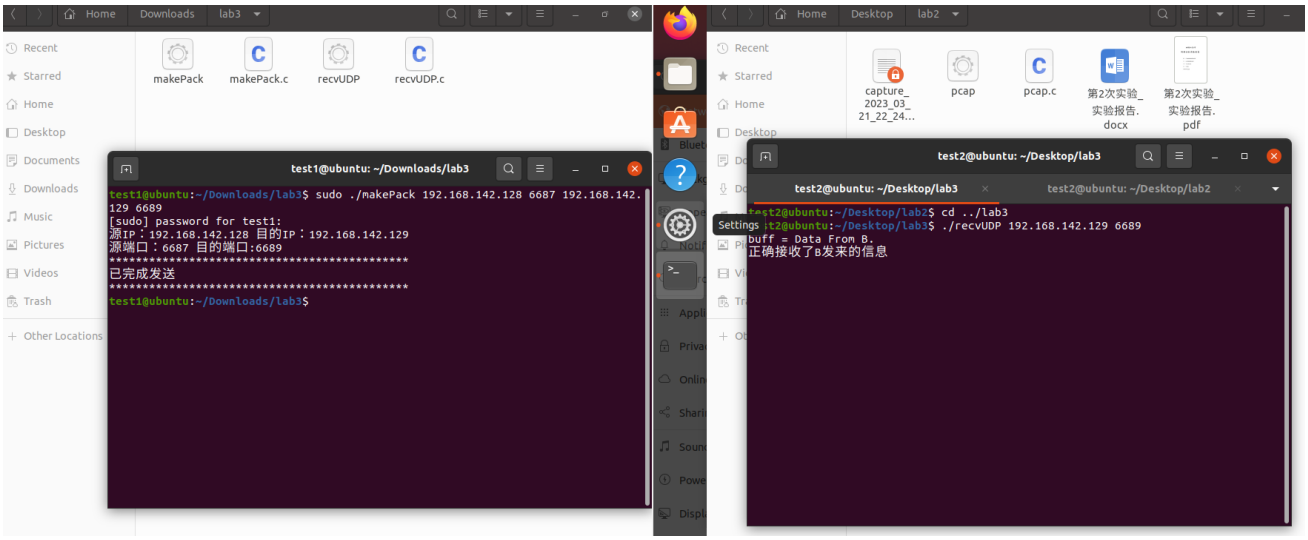

图 recvUDP 检验结果

在实验过程中，主机 A 运行 makePack 程序，其中源 IP 地址为 192.168.142.128，源端口为 6687；目的 IP 地址为 192.168.142.129，目的端口为 6689；

从上图的实验结果中可以看到，makePack 程序已正确发送数据包，并且主机 B 上的 recvUDP 程序也正确接收到该数据。

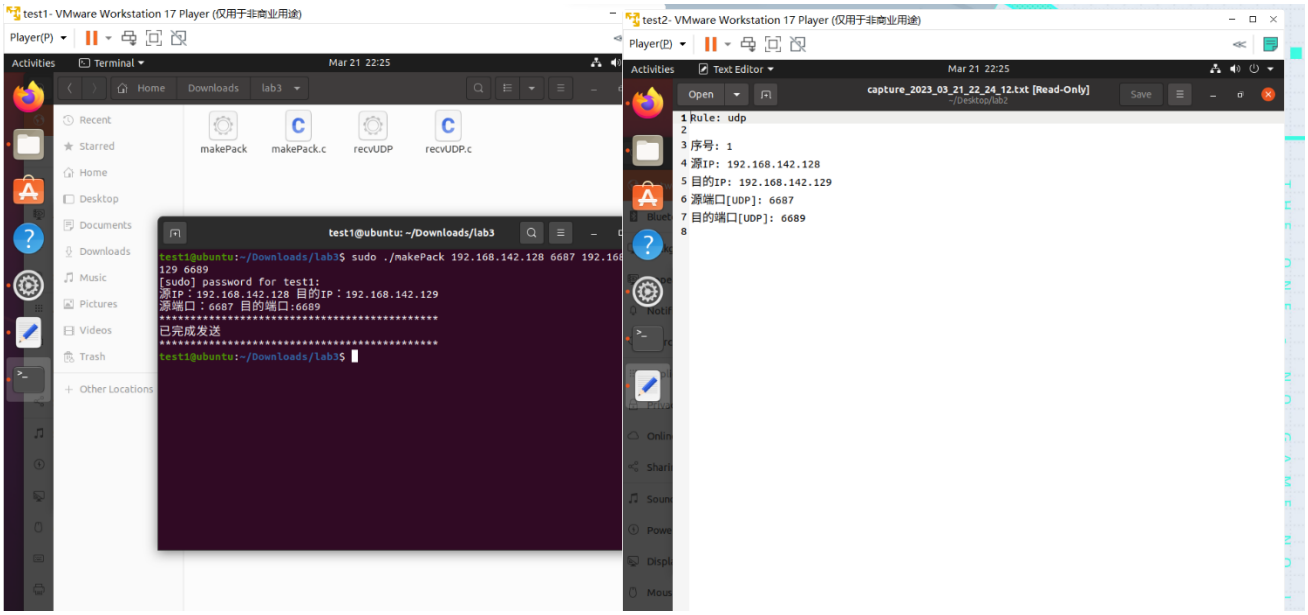

图 pcap 捕获结果

从上图可以看到，主机 B 上的 pcap 程序捕获到该数据包，且正确解析该数据包的参数；

五、心得体会

(1) 掌握基于 libnet 开发流程，并正确依次构建了 UDP 数据包、IP 数据报以及以太网帧。

(2) 更深入了解了网络中两台主机之间发送数据的具体过程，以及在各个网络层次结构中所进行的具体操作；