**海南大学计算机与网络空间安全学院**

**实 验 报 告**

|  |  |
| --- | --- |
| **评定成绩** | **指导教师** |
|  |  |

**实验课程：** 网络协议与分析

**实验名称：**Socket原始套接字编程-自制Sniffer

1. 实验目的
   1. 掌握TCP/IP协议族的分层机制及数据包格式
   2. 掌握Socket编程的基础知识
   3. 掌握抓包的基本原理
2. 实验任务

编程实现抓取局域网中所有经过本机、含有（明文）用户名和密码信息的数据包，并把抓取的相关内容显示出来。

1. 实验环境

Win10+visual studio2019

1. 实验内容

**sniffer程序结构流程:**

socket（）创建原始套接字

定义IPHeader结构

bind（）创建原始套接字

WSAIoctl（）将网卡设置为混杂模式

recv（）接收IP数据包

解IP包，并查找用户密码

将用户密码信息输出到屏幕

丢弃，准备接收下个IP包

错误包或没有找到密码

找到密码

**代码实现：**

#include <winsock2.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#pragma comment(lib,"WS2\_32.lib")

#pragma warning(disable: 4996)

#define MAX\_PACK\_LEN 4096

#define MAX\_ADDR\_LEN 16

#define MAX\_HOSTNAME\_LEN 255

#define SIO\_REVALL \_WSAIOW(IOC\_VENDOR,1)

**typedef** **struct** \_iphdr

{

    unsigned **char** h\_lenver;

    unsigned **char** tos;

    unsigned **short** total\_len;

    unsigned **short** ident;

    unsigned **short** frag\_and\_flags;

    unsigned **char** ttl;

    unsigned **char** proto;

    unsigned **short** checksum;

    unsigned **int** sourceIP;

    unsigned **int** destIP;

}IP\_HEADER;

SOCKET SockRaw;

**int** DecodeIpPack(**char** **const**\*, **int**);

**void** CheckSockError(**int**, **char** **const**\*);

**void** main(**int** argc, **char** **const**\*\* argv)

{

**int** iErrorCode;

**char** RecvBuf[MAX\_PACK\_LEN] = { 0 };

    WSADATA wsaData;

**char** name[MAX\_HOSTNAME\_LEN];

**struct** hostent\* pHostent;

    SOCKADDR\_IN sa;

**DWORD** dwBufferLen[10];

**DWORD** dwBufferInLen = 1;

**DWORD** dwBytesReteuned = 0;

    printf("----Now sniffing pass,CTRL+C to exit...\n\n");

    iErrorCode = WSAStartup(0x0202, &wsaData);

    CheckSockError(iErrorCode, "WSAStartup");

    SockRaw = socket(AF\_INET, SOCK\_RAW, IPPROTO\_IP);

    CheckSockError(SockRaw, "socket");

    iErrorCode = gethostname(name, MAX\_HOSTNAME\_LEN);

    CheckSockError(iErrorCode, "gethostname");

    pHostent = (**struct** hostent\*)malloc(**sizeof**(**struct** hostent));

    pHostent = gethostbyname(name);

    sa.sin\_family = AF\_INET;

    sa.sin\_port = htons(9999);

    memcpy(&sa.sin\_addr.S\_un.S\_addr, pHostent->h\_addr\_list[0], pHostent->h\_length);

    iErrorCode = bind(SockRaw, (PSOCKADDR)&sa, **sizeof**(sa));

    CheckSockError(iErrorCode, "bind");

    iErrorCode = WSAIoctl(SockRaw, SIO\_REVALL, &dwBufferInLen, **sizeof**(dwBufferInLen), &dwBufferLen, **sizeof**(dwBufferLen), &dwBytesReteuned, NULL, NULL);

    CheckSockError(iErrorCode, "Ioctl");

**while** (1)

    {

        memset(RecvBuf, 0, **sizeof**(RecvBuf));

        iErrorCode = recv(SockRaw, RecvBuf, **sizeof**(RecvBuf), 0);

        CheckSockError(iErrorCode, "recv");

        iErrorCode + DecodeIpPack(RecvBuf, iErrorCode);

        CheckSockError(iErrorCode, "Decode");

    }

}

**int** DecodeIpPack(**char** **const**\*buf, **int** iBufSize)

{

    IP\_HEADER \*pIpheader;

**int** iIphLen, iTTL;

**char** szSourceIP[MAX\_ADDR\_LEN], szDestIP[MAX\_ADDR\_LEN];

    SOCKADDR\_IN saSource, saDest;

**const** **char** \*SearchPass, \*start, \*end;

    pIpheader = (IP\_HEADER\*)buf;

    saSource.sin\_addr.s\_addr = pIpheader->sourceIP;

    strncpy(szSourceIP, inet\_ntoa(saSource.sin\_addr), MAX\_ADDR\_LEN);

    saDest.sin\_addr.s\_addr = pIpheader-> destIP;

    strncpy(szDestIP, inet\_ntoa(saDest.sin\_addr), MAX\_ADDR\_LEN);

    iTTL = pIpheader->ttl;

    iIphLen = **sizeof**(unsigned **long**) \* (pIpheader->h\_lenver & 0xf);

    SearchPass = buf + iIphLen + 20;

    start = strstr(SearchPass, "user");

**if** (start == NULL)

        start = strstr(SearchPass, "User");

**if** (start == NULL)

        start = strstr(SearchPass, "uid");

**if** (start == NULL)

        start = strstr(SearchPass, "Uid");

    end = strstr(SearchPass, "pass");

**if** (end == NULL)

        end = strstr(SearchPass, "Pass");

**if** (end == NULL)

        end = strstr(SearchPass, "PASS");

**if**(end==NULL)

        end = strstr(SearchPass, "pwd");

**if**(end==NULL)

        end = strstr(SearchPass, "psw");

**if** (start != NULL && end != NULL)

    {

        printf("\n\n----------Print begin : ------------");

        printf("\n%s->%s", szSourceIP, szDestIP);

        printf("bytes=%d TTL=%d \n", iBufSize, iTTL);

        printf("%s", SearchPass);

    }

**return** 0;

}

**void**  CheckSockError(**int** iErrorCode, **char** **const**\* pErrorMsg)

{

**if** (iErrorCode == SOCKET\_ERROR)

    {

        printf("%s Error:%d\n", pErrorMsg, GetLastError());

        closesocket(SockRaw);

        exit(0);

    }

}

**运行结果：**

基本理解了原理，编译都没问题但是执行时接收不到任何数据。

**五、实验总结**

通过本次实验我能熟练是用套接字编程的基础知识，掌握了TCP/IP协议族的分层机制及数据包格式，，理解了抓包的基本原理。通常我们是把TCP/IP协议族分为四层，但是如果是五层的话就是在链路层下再加个物理层。1、链路层/数据链路层/网络接口层，操作系统中的设备驱动程序和计算机的网络接口卡，负责处理与网络电缆或其他媒介的物理接口细节。该层协议主要包括ARP、RARP。2、网络层，处理分组在网络中的活动，如分组的选路；该层协议包括IP协议，ICMP协议，IGMP协议。3、运输层，主要为在两台主机上的应用程序提供端到端的通信。主要包括TCP协议、UDP协议。4、应用层，特定的应用程序细节。如Telnet、FTP、SMTP、SNMP等。