

电子科技大学信息与软件工程学院

# 实 验 报 告

学 号 2018091618008

姓 名 袁昊男

(实验) 课程名称 计算机病毒防治

理论教师 王 静

实验教师 郝晓青

# 电子科技大学

## 实验报告

学生姓名：袁昊男      学号：2018091618008      指导教师：王静

实验地点：信软楼 306      实验时间：2020.11.20

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：剖析木马程序实验

三、实验学时：8 学时

四、实验原理：

木马的全称为特洛伊木马，来源于古希腊神话。木马是具有远程控制、信息偷取、隐私传输功能的恶意代码，它通过欺骗或者诱骗的方式安装，并在用户的计算机中隐藏以实现控制用户计算机的目的。

### （一）木马的特性

木马程序为了实现其特殊功能，一般应该具有以下性质。

伪装性：木马程序伪装成合法程序，并且诱惑被攻击者执行，使木马代码会在未经授权的情况下装载到系统中并开始运行。

隐藏性：木马程序不被杀毒软件杀掉，会在系统中采用一些隐藏手段，不会让使用者觉察到木马的存在，例如进程在任务管理器中隐藏，文件不会出现在浏览器中等，从而实现长久对目标计算机的控制。

破坏性：通过木马程序的远程控制，攻击者可以对目标计算机中的文件进行删除，修改、远程运行，还可以进行诸如改变系统配置等恶性破坏系统。

窃密性：木马程序最大的特点就是可以偷取被入侵计算机上的所有资料，包括硬盘上的文件，以及屏幕信息，键盘输入信息等。

### （二）木马的入侵途径

木马的入侵主要是通过一些欺骗手段实施的。

捆绑欺骗：如果把木马文件与普通文件捆绑，并更改捆绑后的文件图标，伪造成与原文件类似。再通过电子邮件、QQ、MSN 等手段直接发送给用户，或者通过放在网上或者某个服务器中，欺骗被攻击者下载直接执行。

利用网页脚本入侵：木马也可以通过 Script, active, ASP, CGI 交互脚本的方式入侵，由于微软的浏览器在执行 Script 脚本上存在一些漏洞，攻击者可以利用这些漏洞实现木马的下载和安装。

利用漏洞入侵：木马还利用一些系统的漏洞入侵，如微软的 IIS 服务器存在多种溢出漏洞，通过缓冲区溢出攻击程序造成 IIS 服务器溢出，获得控制权限，然后在被攻击的服务器上安装并运行木马。

和病毒协作入侵：现在的病毒有多种自动感染和传播功能，而木马往往和病毒协同工作，在病毒感染目标计算机后，通过木马对目标计算机进行控制。

### （三）木马的种类

按照发展历程和主流技术的演变，可以将木马分为 5 个阶段。

第一代木马是出现在 20 世纪 80 年代，主要是 UNIX 环境中通过命令行界面实现远程控制。

第二代木马出现在 20 世纪 90 年代，随着 WINDOWS 系统的普及木马在 WINDOWS 环境中大量应用，它具备伪装和传播两种功能，具有图形控制界面，可以进行密码窃取、远程控制，例如 BO2000 和冰河木马。因为防火墙的普遍应用，第二代木马在进入 21 世纪之后不再有多少用武之地了，由于它采用黑客主动连接用户的方式，对于这种从外网发来的数据包都将被防火墙阻断。

第三代木马在连接方式上比第二代木马有了改进，通过端口反弹技术，可以穿透硬件防火墙，例如灰鸽子木马，但木马进程外联黑客时会被软件防火墙阻挡，经验丰富的网络管理员都可以将其拦截。

第四代木马在进程隐藏方面比第三代木马做了较大改动，木马通过线程插入技术隐藏在系统进程或应用进程中，实现木马运行中没有进程，网络连接也隐藏在系统进程或应用进程中，比如广外男生木马。第四代木马可以实现对硬件防火墙的穿透，同时它隐藏在系统或应用进程中，往往网络管理员很难识别，所以被软件防火墙拦截后往往又被放行，从而实现对软件防火墙的穿透。

第五代木马在隐藏方面比第四代木马又进行了进一步提升，它普遍采用了 ROOTKIT 技术，通过 ROOTKIT 技术实现木马运行时进程、文件、服务、端口等的隐藏，采用系统标准诊断工具难以发现它的踪迹。

除了按照技术发展分类之外，从功能上木马又可以分为：破坏型木马，主要功能是破坏并删除文件；密码发送型木马，它可以找到密码并发送到指定的邮箱中；服务型木马，它通过启动 FTP 服务或者建立共享目录，使黑客可以连接并下载文件；DOS 攻击型木马，它将作为被黑客控制的肉鸡实施 DOS 攻击；代理型木马，可使被入侵的机器作为黑客发起攻击的跳板；远程攻击型木马，可以使攻击者利用客户端软件进行完全控制。

#### （四）木马的连接方式

一般的木马都采用 C/S 运行模式，它分为两部分，即客户端和服务端木马程序。黑客安装木马的客户端，同时诱骗用户安装木马的服务端。下面简单介绍木马的传统连接技术、反弹端口技术和线称技术。

##### （1）木马的传统连接技术

第一代木马和第二代木马均采用传统的连接方式，即由木马的客户端程序主动连接服务端程序。当服务端程序在目标计算机上被执行后，一般会打开一个默认的端口进行监听，当客户端向服务器主动提出连接请求，服务端的木马程序就会自动运行，来应答客户端的请求，从而建立连接。

##### （2）木马的反端口技术

随着防火墙技术的发展，它可以有效拦截采用传统连接方式从外部主动发起连接的木马程序。但通常硬件防火墙对内部发起的连接请求则认为是正常连接，第三代之后的“反弹式”木马就是利用这个缺点，其服务端程序主动发起对外连接请求，连接到木马的客户端，就是说“反弹式”木马是服务端主动发起连接请求，而客户端被动的等待连接。

根据客户端的 IP 地址是静态的还是动态的，反弹端口可以有两种连接方式。

反弹端口的第一种连接方式，在设置服务端时要设置固定的客户端 IP 地址和待连接端口，所以这种方式只适用于客户端 IP 地址是公网 IP 且是静态 IP 的情况。

反弹端口的第二种连接方式，可实现服务端根据配置主动连接变动了 IP 的客户端。入侵者为了避免暴露自己的身份，往往通过跳板计算机控制被入侵用户的计算机，在跳板计算机中安装木马客户端软件，被入侵用户的计算机安装木马的服务端软件。当然，入侵者的跳板计算机有时可能失去入侵者的控制，这时入侵者就需要找到新的跳板计算机，同时使用户计算机上的木马服务端程序，能够连接到新跳板计算机上的木马客户端程序。为此，入侵者利用了一个“代理服务器”保存改变后的客户端 IP 地址和待连接的端口，只要跳板主机改变了 IP 地址，入侵者就可以更新“代理服务器”中存放的 IP 地址和端口号。远程被入侵主机上的服务端程序每次启动后，被设置为先连接“代理服务器”，查询最新木马客户端的 IP 和端口信息，再按照新的 IP 地址和客户端进行连接，因此这种连接方式可以适用于客户端和服务端都是动态 IP 地址的情况，而且还可以穿透更加严密的防火墙，当然客户端的 IP 要求是公网 IP 才行。

#### （五）木马的隐藏技术

木马为了防止被杀毒软件查杀，同时也避免被用户计算机发现，往往采用一些隐藏技术，在系统中实现隐身。

### **(1) 线程插入技术**

一般一个应用程序在运行之后，都会在系统中产生一个进程，同时，每个进程分别对应了一个不同的 PID（process ID，进程标示符）。系统会分配一个虚拟的内存空间地址段给这个进程，一切相关的程序操作，都会在这个虚拟的空间中进行。一个进程可以对应一个或多个线程，线程之间可以同步执行，一般情况下，线程之间是相互独立的，当一个线程发生错误的时候，并不一定会导致整个进程的崩溃，“线程插入技术”就是运用了线程之间运行的相对独立性，使木马完全溶进了系统的内核。这种技术把木马程序作为一个线程，把自身插入到其他应用程序的地址空间。而这个被插入的应用程序对于系统来说，是一个正常的程序，这样，就达到了彻底的隐藏效果。系统运行时会有许多的进程，而每个进程又有许多的线程，这就导致了查杀利用“线程插入”技术木马程序难度的增加。

### **(2) ROOTKIT 技术**

ROOTKIT 是一种隐藏技术，它使得恶意程序可以逃避操作系统标准诊断程序的查找。早期的 ROOTKIT 技术通过修改内存中的系统文件映像来逃避检测。这一类 ROOTKIT 技术主要依赖 HOOK 技术，比如 HOOK API 或者系统调用表。目前主流的 ROOTKIT 技术主要在内核态实现，例如 DKOM（直接内核对象操作）技术，通过动态修改系统中的内核数据结构来逃避安全软件的监测。由于这些数据结构随着系统的运行而不断更新变化，因此非常难于检测。

### **(六) 木马的检测**

木马的远程控制功能要实现，必须通过执行一段代码来实现。为此，木马采用的技术再新，也会在操作系统中留下痕迹。如果能够对系统中的文件、注册表做全面的监控，可以实现发现和清楚各种木马的目的。当然现有的监控手段还不一定能够做到全面的监控，但对系统的行为监控也已经非常深入了，通过运用多种监控手段和工具，可以协助发现植入的木马。当然，由于木马的机制不同，所以检测和查杀手段也不尽相同。

## **五、实验目的：**

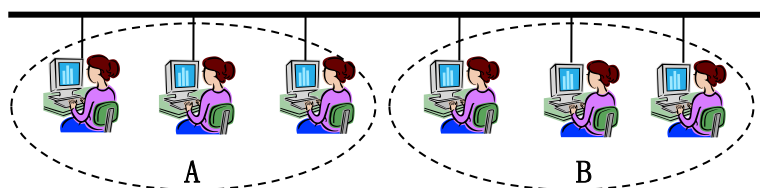
通过对木马的练习，理解与掌握木马传播与运行的机制；通过手动删除木马，掌握检查木马和删除木马的技巧，学会防御木马的相关知识，加深对木马的安全防范意识。

## **六、实验内容：**

- 1、Socket 编程实验。
- 2、Mini 木马程序剖析实验。
- 3、经典木马程序实验。

## 七、实验器材（设备、元器件）：

- 1、实验人数 50~80 人，每人 1 台计算机；两人一组完成本实验。
- 2、拓扑：（A、B 范围中的主机分别简称为 A 主机和 B 主机）



- 3、设备：以太网交换机 2~4 台；计算机 50~80 台。
- 4、软件：VC++ 6.0 软件或 Visual Studio。

## 八、实验步骤：

### （一）Socket 编程实验

- (1) 使用 VC 或 VS，新建一个“Win32 Console Application”类型的工程。
- (2) 在这个项目中编写一个基于 TCP Socket 的服务端“C++ Source File”，其流程是 `WSAStartup()`—`socket()`—`bind()`—`accept()`—`send()`，完成的功能是服务端监听主机 A 的某个端口，一旦有客户端 telnet 这个端口，就向客户端发送欢迎语句如“hello”等。
- (3) 在主机 A 上执行这个程序，使用“`netstat -an`”命令查看程序中指定的端口处于什么状态。
- (4) 选择主机 B 作为客户端，使用“`telnet IP port`”命令连接主机 A，记录运行结果。

### （二）Mini 木马程序剖析实验

- (1) 在主机 A 上编译组建执行 mini 木马程序。  
Mini 程序体现了木马的基本功能远程控制，无须客户端软件，服务端精简，占用非常少的 CPU 和内存资源，便于隐藏。但不能自启动，需要第三方软件加载到自启动项目或服务中。
- (2) 主机 A 上使用“`netstat -an`”命令查看端口 999 处于什么状态。
- (3) 选择主机 B 作为客户端，使用“`telnet IP 999`”命令连接主机 A，记录运行结果。
- (4) 主机 A 上使用“`netstat -an`”命令查看主机 B 的哪个端口和主机 A 的 999

端口建立了连接，状态是什么。同时在主机 A 上

(5) 主机 B 的 Telnet 窗口中，使用 `ipconfig` 和 “`net user`” 命令查看系统的 IP 地址和用户。

(6) 使用 “`net user 用户名 密码 /add`” 命令增加一个用户。

(7) 使用 “`net localgroup Administrators 用户名 /add`” 命令将该用户添加到 Administrators 组。

(8) 使用 “`net localgroup Administrators`” 命令查看该组下有哪些用户。

(9) 使用 `exit` 命令退出 telnet 连接。

### (三) 经典木马程序实验

(1) 关机程序 `shutdown`，阅读程序代码，执行程序自动关闭系统。

(2) 进程查杀程序 `process`

a) 阅读程序代码。

b) 打开一个 `cmd` 窗口，使用 `tasklist` 命令查看系统进程和 PID 号。记下当前 `cmd` 窗口的 PID 号。

c) 执行 `process` 程序，结果与 `tasklist` 命令比较，并在提示语句后输入 `cmd` 窗口的 PID 号，结果会怎样。

(3) 获取主机 IP 地址程序 `hostip`，阅读程序代码，执行程序列出当前主机地址。

(4) 多线程 TCP 扫描程序 `tcpscanner`，阅读程序代码，执行程序列出当前主机端口。

(5) 下载者程序 `download`，阅读程序代码，修改程序为 `ftp` 协议下载，执行程序，查看是否在主机上下载成功。

(6) 执行注册表读取程序 `read`，分析其取得是注册表哪个位置的值？取得的值是否跟注册表里的信息一致？

a) 注册表是以树状结构储存，每一个节点称为一个键值 (`key`)，每个 `key` 又包括子键值 (`subkey`) 及数据入口的值 (`value`)。读写注册表前，必须先将目标的子键打开，也就是取得一个操作的句柄。

b) 打开函数

```
LONG RegOpenKeyEx(  
    HKEY hKey, // 需要打开的主键的句柄  
    LPCTSTR lpSubKey, // 需要打开的子键的名称  
    DWORD ulOptions, // 保留，设为 0  
    REGSAM samDesired, // 安全访问标记，也就是权限  
    PHKEY phkResult // 得到的将要打开键的句柄  
);
```

## 九、实验数据及结果分析

### (一) Socket 编程实验

(1) 使用 VC 或 VS，新建一个“Win32 Console Application”类型的工程。

(2) 在这个项目中编写一个基于 TCP Socket 的服务端“C++ Source File”，其流程是 WSAStartup()—socket()—bind()—accept()—send()，完成的功能是服务端监听主机 A 的某个端口，一旦有客户端 telnet 这个端口，就向客户端发送欢迎语句如“hello”等。

代码：

```
1. #pragma comment(lib, "ws2_32.lib")
2. #include <winsock2.h>
3. #include <stdio.h>
4. int main()
5. {
6.     SOCKET mysock, tsock;    // 定义套接字
7.     struct sockaddr_in my_addr; // 本地地址信息
8.     struct sockaddr_in their_addr; // 连接者地址信息
9.     int sin_size;
10.    WSADATA wsa;
11.    WSAStartup(MAKEWORD(2, 2), &wsa); //初始化 Windows Socket
12.    //建立 socket
13.    mysock = socket(AF_INET, SOCK_STREAM, 0);
14.    //bind 本机的端口
15.    my_addr.sin_family = AF_INET;    // 协议类型是 INET
16.    my_addr.sin_port = htons(1234); // 绑定端口 1234
17.    my_addr.sin_addr.s_addr = INADDR_ANY;    // 本机 IP
18.    bind(mysock, (struct sockaddr*)&my_addr, sizeof(struct sockadd
        r));
19.    //listen, 监听端口
20.    listen(mysock, 10); // 等待连接数目
21.    printf("listen.....\n");
22.    //等待客户端连接
23.    sin_size = sizeof(struct sockaddr_in);
24.    tsock = accept(mysock, (struct sockaddr*)&their_addr, &sin_siz
        e); //有连接就发送 Hello!字符串过去
25.    send(tsock, "Hello!\n", sizeof("Hello!\n"), 0);
26.    printf("send ok!\n");
27.    //成功, 关闭套接字
28.    closesocket(mysock);
29.    closesocket(tsock);
30.    return 0;
31. }
```



(3) 在主机 A 上执行这个程序，使用 “netstat -an” 命令查看程序中指定的端口处于什么状态。

```
C:\Windows\system32\cmd.exe
C:\Users\90389>netstat -an

活动连接

 协议 本地地址           外部地址           状态
TCP    0.0.0.0:135         0.0.0.0:0          LISTENING
TCP    0.0.0.0:445         0.0.0.0:0          LISTENING
TCP    0.0.0.0:1234        0.0.0.0:0          LISTENING
TCP    0.0.0.0:5040        0.0.0.0:0          LISTENING
TCP    0.0.0.0:5357        0.0.0.0:0          LISTENING
TCP    0.0.0.0:7680        0.0.0.0:0          LISTENING
TCP    0.0.0.0:10253       0.0.0.0:0          LISTENING
TCP    0.0.0.0:49664       0.0.0.0:0          LISTENING
TCP    0.0.0.0:49665       0.0.0.0:0          LISTENING
TCP    0.0.0.0:49666       0.0.0.0:0          LISTENING
TCP    0.0.0.0:49667       0.0.0.0:0          LISTENING
TCP    0.0.0.0:49668       0.0.0.0:0          LISTENING
```

指定端口 1234 处于 LISTENING 监听状态。

(4) 选择主机 B 作为客户端，使用 “telnet IP port” 命令连接主机 A，记录运行结果。

A 机连接前：

```
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe
listen.....
```

B 机发起 telnet 连接：

```
tommy - zsh - 80x24
tommy@TMMacBook-Pro ~ % telnet 113.54.246.253 1234
Trying 113.54.246.253...
Connected to 113.54.246.253.
Escape character is '^]'.
Hello!
Connection closed by foreign host.
tommy@TMMacBook-Pro ~ %
```

A 机连接后:



```
Microsoft Visual Studio 调试控制台

listen.....
send ok!

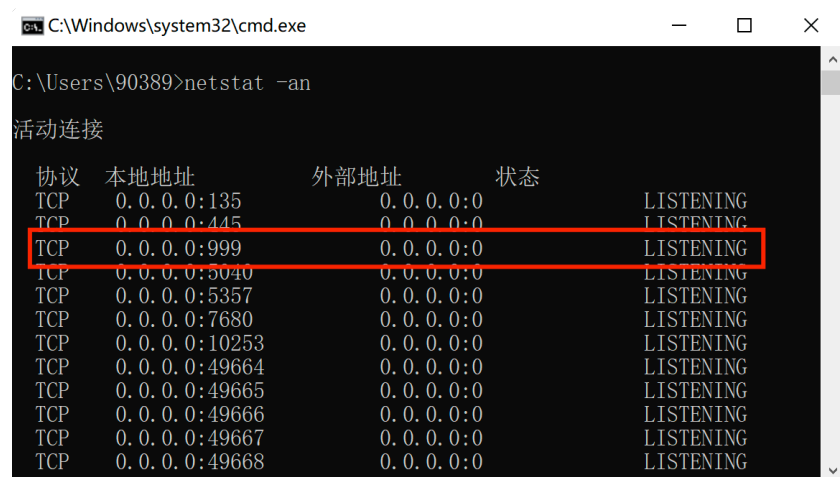
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe (进程 5308) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

## (二) Mini 木马程序剖析实验

(1) 在主机 A 上编译组建执行 mini 木马程序。

Mini 程序体现了木马的基本功能远程控制, 无须客户端软件, 服务端精简, 占用非常少的 CPU 和内存资源, 便于隐藏。但不能自启动, 需要第三方软件加载到自启动项目或服务中。

(2) 主机 A 上使用 “netstat -an” 命令查看端口 999 处于什么状态。



```
C:\Windows\system32\cmd.exe

C:\Users\90389>netstat -an

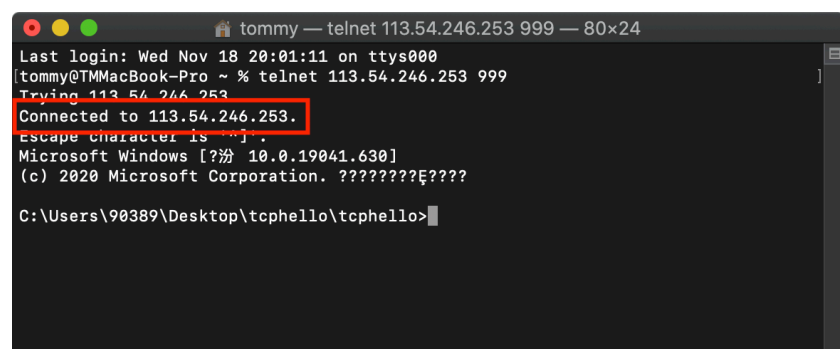
活动连接

```

协议	本地地址	外部地址	状态
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:999	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5040	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5357	0.0.0.0:0	LISTENING
TCP	0.0.0.0:7680	0.0.0.0:0	LISTENING
TCP	0.0.0.0:10253	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING

指定端口 999 处于 LISTENING 监听状态。

(3) 选择主机 B 作为客户端, 使用 “telnet IP 999” 命令连接主机 A, 记录运行结果。



```
tommy — telnet 113.54.246.253 999 — 80x24

Last login: Wed Nov 18 20:01:11 on ttys000
[tommy@TMMacBook-Pro ~ % telnet 113.54.246.253 999]
Trying 113.54.246.253...
Connected to 113.54.246.253.
Escape character is '^]'.
Microsoft Windows [? 10.0.19041.630]
(c) 2020 Microsoft Corporation. ????????
C:\Users\90389\Desktop\tcphello\tcphello>
```

(4) 主机 A 上使用 “netstat -an” 命令查看主机 B 的哪个端口和主机 A 的 999 端口建立了连接，状态是什么。

```

C:\Windows\system32\cmd.exe
TCP 0.0.0.0:10253 0.0.0.0:0 LISTENING
TCP 0.0.0.0:49664 0.0.0.0:0 LISTENING
TCP 0.0.0.0:49665 0.0.0.0:0 LISTENING
TCP 0.0.0.0:49666 0.0.0.0:0 LISTENING
TCP 0.0.0.0:49667 0.0.0.0:0 LISTENING
TCP 0.0.0.0:49668 0.0.0.0:0 LISTENING
TCP 0.0.0.0:49670 0.0.0.0:0 LISTENING
TCP 113.54.246.253:139 0.0.0.0:0 LISTENING
TCP 113.54.246.253:999 113.54.232.236:58377 ESTABLISHED
TCP 113.54.246.253:64435 42.236.43.101:80 SYN_SENT
TCP 127.0.0.1:10000 0.0.0.0:0 LISTENING
TCP 127.0.0.1:54360 0.0.0.0:0 LISTENING
TCP [::]:135 [::]:0 LISTENING
TCP [::]:445 [::]:0 LISTENING
TCP [::]:5357 [::]:0 LISTENING
TCP [::]:7680 [::]:0 LISTENING
TCP [::]:49664 [::]:0 LISTENING
TCP [::]:49665 [::]:0 LISTENING

```

主机 B 的 58377 端口与主机 A 建立了连接，状态是 ESTABLISHED。

(5) 主机 B 的 Telnet 窗口中，使用 ipconfig 和 “net user” 命令查看系统的 IP 地址和用户。（注：由于主机 B 中 telnet 程序不支持中文字符显示，因此产生乱码。

ipconfig:

```

tommy — telnet 113.54.246.253 999 — 80x24
Microsoft Windows [? 10.0.19041.630]
(c) 2020 Microsoft Corporation. ??????E????
C:\Users\90389\Desktop\tcphello\tcphello>ipconfig
ipconfig

Windows IP ???

???????? Ethernet0:

??????? DNS ?? . . . . . :
IPv6 ?? . . . . . : 2001:da8:6000:e02::63db
???????? IPv6 ?? . . . . . : fe80::e548:884e:75fa:6c08%5
IPv4 ?? . . . . . : 113.54.246.253
???????? . . . . . : 255.255.224.0
I??????? . . . . . : 113.54.224.1

???????? ??????????:

y??
??????? DNS ?? . . . . . : y???V?????
C:\Users\90389\Desktop\tcphello\tcphello>

```

net user:

```

tommy — telnet 113.54.246.253 999 — 80x24
tommy@TMMacBook-Pro ~ % telnet 113.54.246.253 999
Trying 113.54.246.253...
Connected to 113.54.246.253.
Escape character is '^'.
Microsoft Windows [? 10.0.19041.630]
(c) 2020 Microsoft Corporation. ??????E????
C:\Users\90389\Desktop\tcphello\tcphello>net user
net user

\\DESKTOP-COMRNF5 ???0??'

-----
90389 Administrator DefaultAccount
Guest VUSR_DESKTOP-COMRNF5 WDAGUtilityAccount
????J???g?

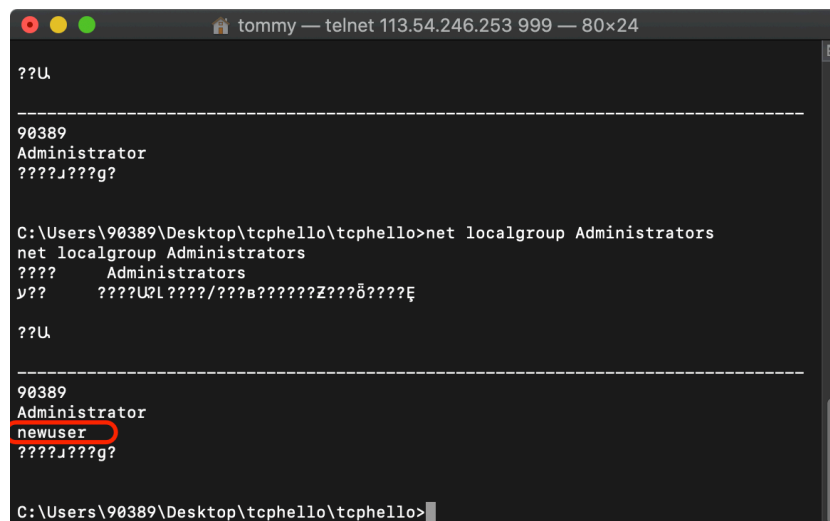
```

(6) 使用 “net user 用户名 密码 /add” 命令增加一个用户。

(7) 使用 “net localgroup Administrators 用户名 /add” 命令将该用户添加到 Administrators 组。

(8) 使用 “net localgroup Administrators” 命令查看该组下有哪些用户。

主机 B 查看:



```
tommy — telnet 113.54.246.253 999 — 80x24

??U

-----

90389
Administrator
????j??g?

C:\Users\90389\Desktop\tcphello\tcphello>net localgroup Administrators
net localgroup Administrators
????      Administrators
v??      ???U?L????/???a???????Z???õ?????E

??U

-----

90389
Administrator
newuser
????j??g?

C:\Users\90389\Desktop\tcphello\tcphello>
```

主机 A 查看:



```
管理员: 命令提示符

C:\Windows\system32>net localgroup Administrators
别名      Administrators
注释      管理员对计算机/域有不受限制的完全访问权

成员

-----

90389
Administrator
newuser
命令成功完成。

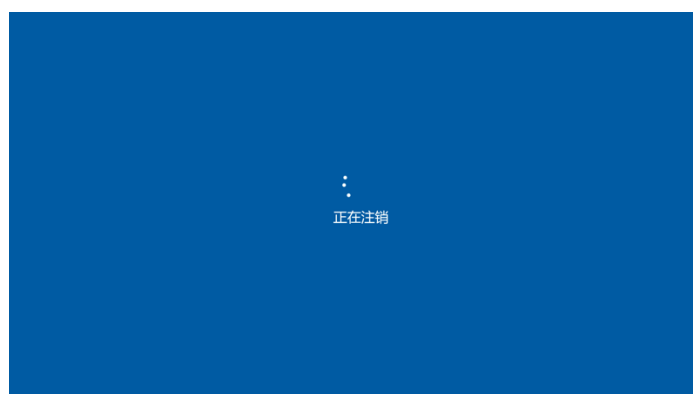
C:\Windows\system32>
```

新增用户为 newuser。

(9) 使用 exit 命令退出 telnet 连接。

### (三) 经典木马程序实验

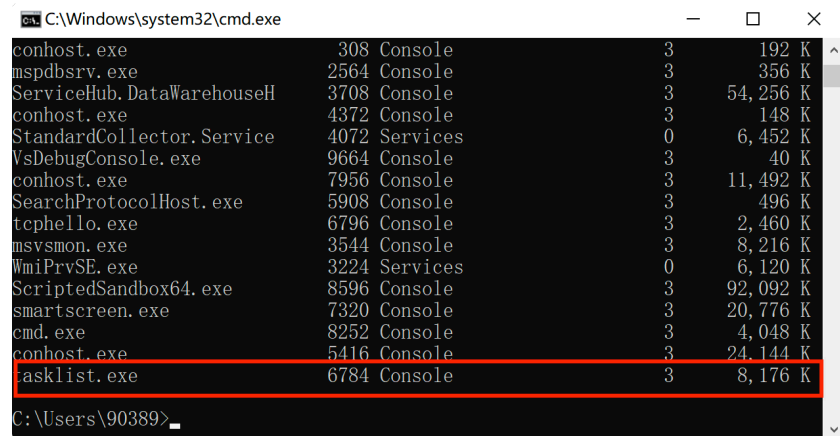
(1) 关机程序 shutdown, 阅读程序代码, 执行程序自动关闭系统。



## (2) 进程查杀程序 process

a) 阅读程序代码。

b) 打开一个 cmd 窗口，使用 tasklist 命令查看系统进程和 PID 号。记下当前 cmd 窗口的 PID 号。

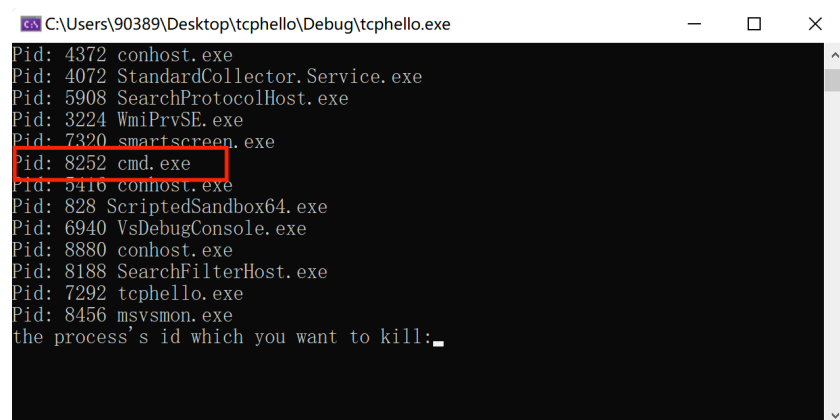


```
C:\Windows\system32\cmd.exe
conhost.exe           308 Console           3      192 K
mspdbsrv.exe         2564 Console           3      356 K
ServiceHub.DataWarehouseH 3708 Console           3 54,256 K
conhost.exe          4372 Console           3      148 K
StandardCollector.Service 4072 Services           0 6,452 K
VsDebugConsole.exe   9664 Console           3      40 K
conhost.exe          7956 Console           3 11,492 K
SearchProtocolHost.exe 5908 Console           3      496 K
tcphello.exe         6796 Console           3 2,460 K
msvsmon.exe          3544 Console           3 8,216 K
WmiPrvSE.exe         3224 Services           0 6,120 K
ScriptedSandbox64.exe 8596 Console           3 92,092 K
smartscreen.exe      7320 Console           3 20,776 K
cmd.exe              8252 Console           3 4,048 K
conhost.exe          5416 Console           3 24,144 K
tasklist.exe         6784 Console           3 8,176 K
C:\Users\90389>
```

当前 cmd 窗口的 PID 号为 8252。

c) 执行 process 程序，结果与 tasklist 命令比较，并在提示语句后输入 cmd 窗口的 PID 号，结果会怎样。

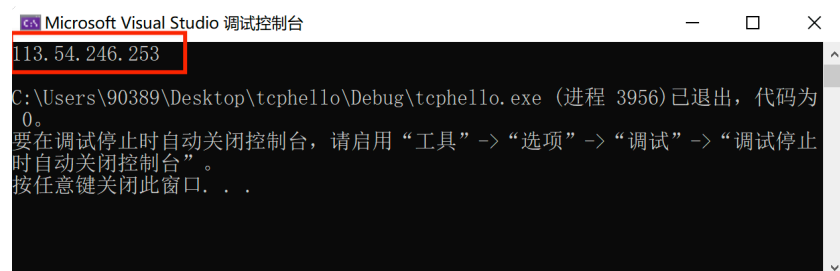
执行 process 程序：



```
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe
Pid: 4372 conhost.exe
Pid: 4072 StandardCollector.Service.exe
Pid: 5908 SearchProtocolHost.exe
Pid: 3224 WmiPrvSE.exe
Pid: 7320 smartscreen.exe
Pid: 8252 cmd.exe
Pid: 5416 conhost.exe
Pid: 828 ScriptedSandbox64.exe
Pid: 6940 VsDebugConsole.exe
Pid: 8880 conhost.exe
Pid: 8188 SearchFilterHost.exe
Pid: 7292 tcphello.exe
Pid: 8456 msvsmon.exe
the process's id which you want to kill: _
```

输入 cmd 窗口 PID 号后，cmd 进程被杀死。

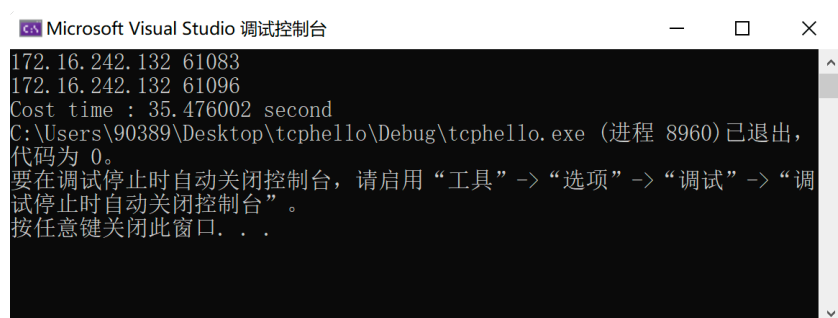
(3) 获取主机 IP 地址程序 hostip，阅读程序代码，执行程序列出当前主机地址。



```
Microsoft Visual Studio 调试控制台
113.54.246.253
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe (进程 3956) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。...
```

(4) 多线程 TCP 扫描程序 tcpscanner，阅读程序代码，执行程序列出当前主

机端口。

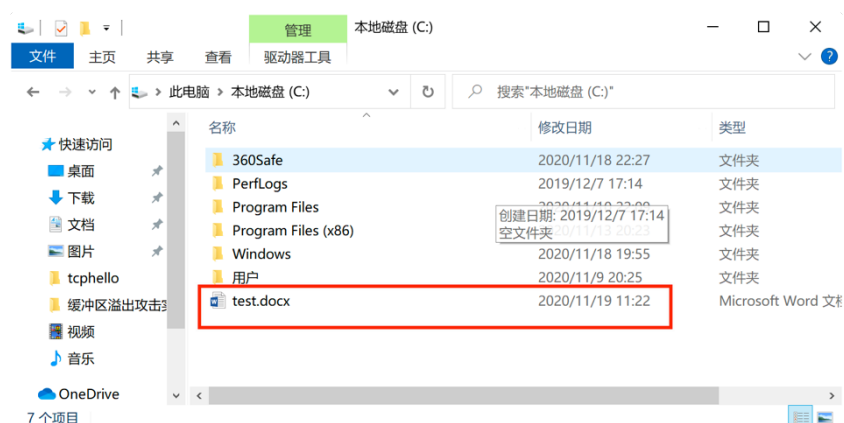


```
Microsoft Visual Studio 调试控制台
172.16.242.132 61083
172.16.242.132 61096
Cost time : 35.476002 second
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe (进程 8960) 已退出,
代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调
试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

(5) 下载者程序 `download`, 阅读程序代码, 修改程序为 `ftp` 协议下载, 执行程序, 查看是否在主机上下载成功。

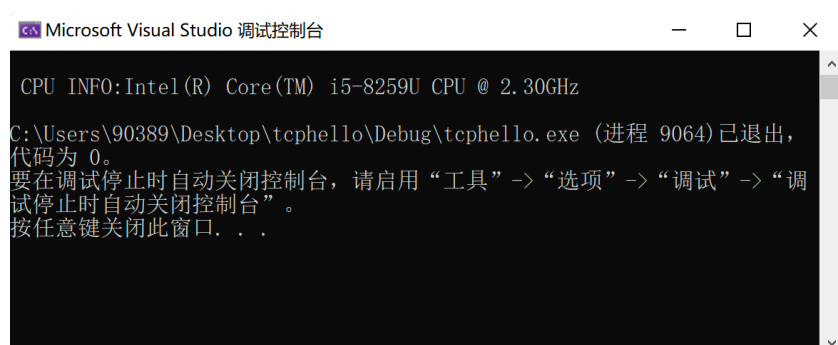


```
Microsoft Visual Studio 调试控制台
下载成功!
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe (进程 10136) 已退出,
代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调
试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

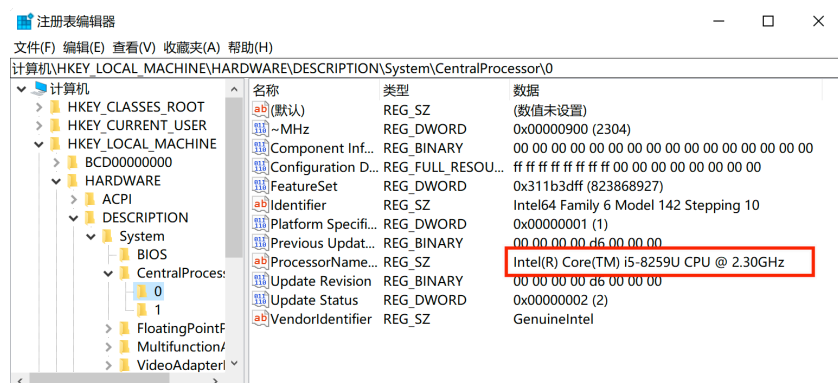


执行程序后, 在主机上成功下载 `test.docx` 文档。

(6) 执行注册表读取程序 `read`, 分析其取得是注册表哪个位置的值? 取得的值是否跟注册表里的信息一致?



```
Microsoft Visual Studio 调试控制台
CPU INFO: Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz
C:\Users\90389\Desktop\tcphello\Debug\tcphello.exe (进程 9064) 已退出,
代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调
试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```



其取得的是注册表中：

HKEY\_LOCAL\_MACHINE\Hardware\Description\System\CentralProcessor\0  
中 ProcessorNameString 的值。取得的信息与注册表中一致。

a) 注册表是以树状结构储存，每一个节点称为一个键值（key），每个 key 又包括子键值（subkey）及数据入口的值（value）。读写注册表前，必须先将目标的子键打开，也就是取得一个操作的句柄。

b) 打开函数

```
LONG RegOpenKeyEx(
    HKEY hKey, // 需要打开的主键的句柄
    LPCTSTR lpSubKey, // 需要打开的子键的名称
    DWORD ulOptions, // 保留，设为 0
    REGSAM samDesired, // 安全访问标记，也就是权限
    PHKEY phkResult // 得到的将要打开键的句柄
);
```

## 十、实验结论

按实验内容与步骤完成本实验，得到的实验结果与预期一致。了解了木马程序的攻击原理，通过 Socket 编程实验、Mini 木马剖析实验以及经典木马程序实验验证了攻击过程。

## 十一、总结及心得体会

通过 Socket 编程实验、Mini 木马剖析实验以及经典木马程序的编写与验证，了解木马的攻击原理、相关概念与危害，理解与掌握木马传播与运行的机制；通过手动删除木马，掌握检查木马和删除木马的技巧，学会防御木马的相关知识，加深对木马的安全防范意识。

## 十二、对本实验过程及方法、手段的改进建议

本实验设计与教材结合紧密、难度较小，通过 Socket 编程实验、Mini 木马剖析实验以及经典木马程序的编写与验证，强化了学生对相关概念的理解与实践，增强了在计算机病毒防治方面的意识，对网络安全课程的深入学习打下了坚实的基础。

**报告评分：**

**指导教师签字：**