杭州電子科技大学

网络安全理论与技术实验

实 验 报 告

学　　院	网络空间安全学院
专　　业	信息安全
班　　级	信安卓越
学　　号	16031330
学生姓名	易涛
教师姓名	吕秋云
完成日期	2018/12/21
成　　绩	

编程类实验总报告

一、 实验目的

1、Socket 下基于 TCP 协议的通信编程实验

（1）掌握基于 TCP 协议的 Socket API 编程的基本原理和方法

（2）通过自己编程实现简单的流套接字的 C/S 模型

2、端口扫描器编程实验

在理解扫描器的原理的基础上，编程实现一款简单的扫描器

3、注册表安全防护编程实验

通过编程实现注册表子键的创建、删除，以及子键键值查询和修改功能，加深对注册表的理解；同时了解注册表在微软系统安全方面的作用，深入分析注册表部分关键键值的功能（如系统启动项，文件关联等注册表键值）

4、恶意代码及防护编程实验

通过设计一款简单的恶意代码程序，深刻理解恶意代码编写的原理，同时设计查杀程序进而理解杀毒软件的工作机理

5、U 盘病毒

编写自己的 U 盘病毒，在实现自我复制传播的基本功能上，添加自己的恶意代码功能

二、 实验内容

1、Socket 下基于 TCP 协议的通信编程实验

在理解基于流套接字（TCP 协议）的编程时序的基础上，利用 VS2017 环境下的 Socket API 来实现简单的网络通信系统，即实现一个简易通信软件系统，其中一个软件（客户端）实现发送信息功能，另一个软件（服务端）实现接受信息功能

2、端口扫描器编程实验

掌握网络扫描器的不同方式，了解各不同扫描方式的实现原理。实现对本机以及输入 IP 地址来对其他网站的端口扫描器

3、注册表安全防护编程实验

设计一款能对注册表键值实现增加、删除、查询、修改的注册表工具，用户可通过该工具查看系统启动项的值，并进行添加删除修改操作，同时该工具可以检查文本文件关联是否正确

4、恶意代码及防护编程实验

恶意代码设计与实现涉及了文件系统编程、网络通信编程、注册表编程、定时编程、多线程编程、驻留程序编程，本实验要求编写一个利用各项编程技术实现简单的独立恶意代码程序，并且根据破坏特征设计查杀程序。此实验程序主要实现（1）自启动功能（2）自动驻留功能（3）实现每天固定时间，自动删除文件（4）更改文本默认打开方式，自动删除该文件（5）远程控制功能

5、U 盘病毒

病毒激活以后，每隔 60 秒扫 描一下本地计算机的 U 盘，如果有 U 盘存在就把 U 盘上的 AutoRun.inf 删除，把自己的 AutoRun.inf 写进去，同时复制自己到 U 盘，

并将 AutoRun.inf 和自身 利用文件属性隐藏。 当 U 盘上程序通过 AutoRun.inf 被激活以后， 将复制自身到操作系统的系统目录。同时当程序在 U 盘运行时，实现不断弹框。

三、 实验环境

1、Socket 下基于 TCP 协议的通信编程实验

Win10 64 位

VS 2017 社区版

2、端口扫描器编程实验

Win10 64 位

Pycharm 2018 专业版

Qtdesigner

(Qt Designer 的设计符合 MVC 的架构，其实现了视图和逻辑的分离，从而实现了开发的便捷。Qt Designer 中的操作方式十分灵活，其通过拖拽的方式放置控件可以随时查看控件效果。Qt Designer 生成的.ui 文件(实质上是 XML 格式的文件)也可以通过 pyuic5 工具转换成.py 文件)

Python 中用到的重要库介绍：

Pyqt5	pyqt5 是一套 Python 绑定 Digia QT5 应用的框架
Threading	多线程模块
Scapy	可以让用户发送、侦听和解析并伪装网络报文的模块
Socket	python 中的 socket 模块

3、注册表安全防护编程实验

Win10 64 位

Pycharm 2018 专业版

Qtdesigner

Python 中用到的重要库介绍：

Pyqt5	pyqt5 是一套 Python 绑定 Digia QT5 应用的框架
Threading	多线程模块
Winreg	操作注册表模块

4、恶意代码及防护编程实验

Win10 64 位

Pycharm 2018 专业版

Qtdesigner

Python 中用到的重要库介绍：

Pyqt5	pyqt5 是一套 Python 绑定 Digia QT5 应用的框架
Threading	多线程模块
Winreg	操作注册表模块
Socket	python 中的 socket 模块
Scapy	可以让用户发送、侦听和解析并伪装网络报文的模块
Pyinstaller	一个将 py 代码打包成 exe 的工具库

5、U 盘病毒

Win10 64 位

Pycharm 2018 专业版

Python 中用到的重要库介绍：

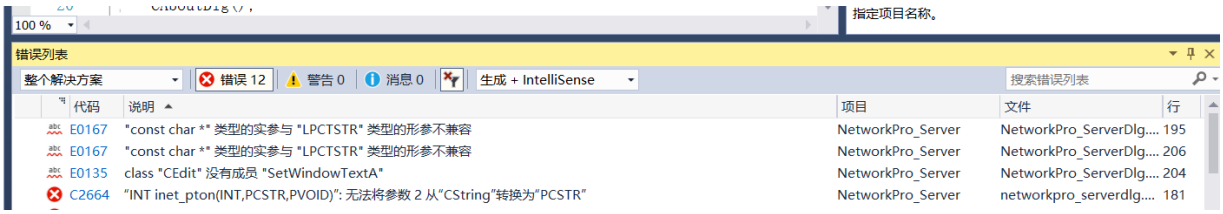
win32api	调用 windows api 的库
psutil	psutil 是一个跨平台库能够轻松实现获取系统运行的进程和系统利用率（包括 CPU、内存、磁盘、网络等）信息
shutil	python 内置库，高级的 文件、文件夹、压缩包 处理模块
tkinter	python 自带简单 GUI 库

四、 实验问题与解决分析

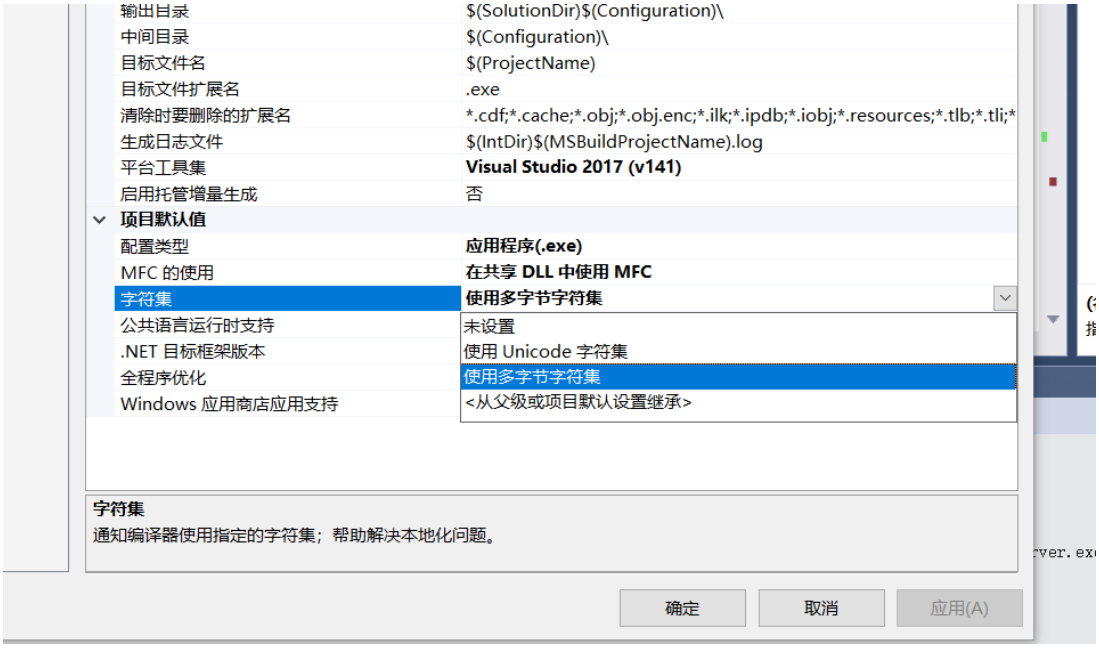
1、Socket 下基于 TCP 协议的通信编程实验

问题 1： 在 MFC 编程中使用 Format 函数时报错

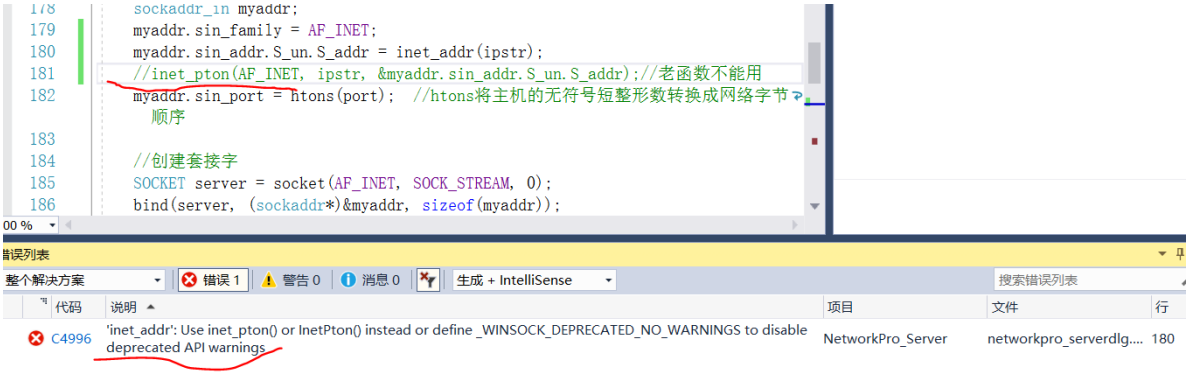
```
ipstr.Format("%d.%d.%d.%d", nFild[0], nFild[1], nFild[2], nFild[3]);//此
处format函数为void型，注意与python区分
```



解决办法： 上网查询报错代码，发现是创建 MFC 工程文件时，选择的字符集默认是 Unicode 字符集，将其更改成使用多字节字符集即可解决



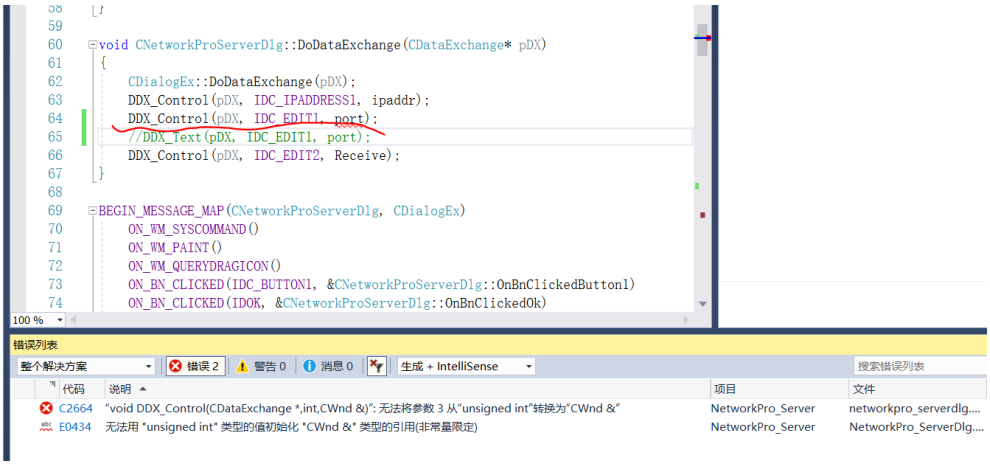
问题 2： 使用 MFC IP 地址控件，需要将输入的十进制 IP 地址转为二进制地址，需要用到 inet_addr 函数。但使用后编译会报错，如下图



解决方法：错误信息里已经给出了解决方法，就是不用 inet_addr 函数，改用 inet_pton 或 InetPton 函数。于是上网搜索 inet_pton 函数用法、定义。还发现错误原因

在VS2013以后的版本中，增加了inet_pton()、InetPton()之类的新函数，用于IP地址在“点分十进制”和“二进制整数”之间转换，并且能够处理ipv4和ipv6。而inet_addr是老函数，高版本VS在编译时默认使用了新函数，所以会报该错误。

问题 3：在将控件做变量绑定时，发生如下报错

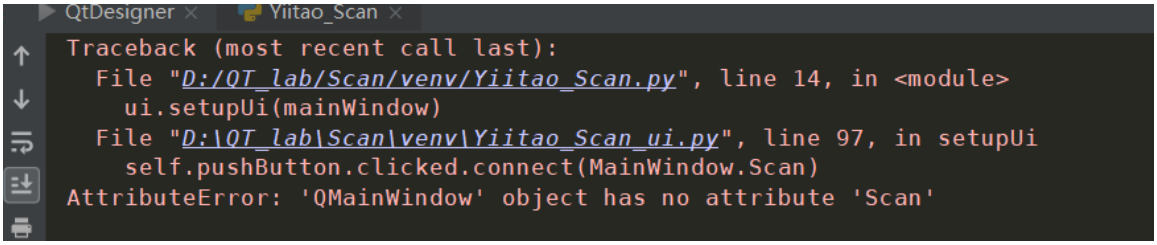


解决方法：搜索对应函数错误代码，并查询相关 DDX 函数各自用法，发现端口控件应当用 DDX_TEXT()函数，于是改用。

> **DDX_TEXT()的作用可以理解为把字符串变量和控件的文本（WindowText）关联起来，**
> **DDX_Control()的作用可以理解为把变量和控件本身关联起来，**
> **DoDataExchange(pDX)就是处理所有变量与其关联控件交换数据的函数。**

2、端口扫描器编程实验

问题 1：用 Qtdesigner 设计 ui 时，若 Qtdesigner 中添加按钮的信号/槽功能的话，则调用时会出现下图示错误



```
95
96              self.retranslateUi(MainWindow)
97              self.pushButton.clicked.connect(MainWindow.Scan)
98              # self.pushButton_2.clicked.connect(self.Sort)
99              self.pushButton_3.clicked.connect(self.Clear)
100             QtCore.QMetaObject.connectSlotsByName(MainWindow)
101
102         def retranslateUi(self, MainWindow):
```

解决方法：将错误信息复制至网上查询，发现某解决方法如下图，按照提示将上图中划线处的 MainWindow 改为 self，更正解决报错，但网上搜索并没搜寻到问题原因，于是自己猜测分析，将分析过程写在下方。

转 **关于某个PyQt5系列教程 "'QMainWindow' object has no attribute" 错误修正**

2017年11月21日 16:06:41 yuhouwucaihong 阅读数：3580 标签： python pyqt5

原文地址：http://www.cnblogs.com/tkinter/p/5632245.html

一直到运行main程序之前之前都没有什么错误；

但是运行后出错：

python IDLE报错：

self.pushButton.clicked.connect(MainWindow.firstPyQt5_button_click)
AttributeError: 'QMainWindow' object has no attribute 'firstPyQt5_button_click'

改正方法：将firstPyQt5.py中的self.pushButton.clicked.connect(MainWindow.firstPyQt5_button_click)

更正为：self.pushButton.clicked.connect(self.firstPyQt5_button_click)

问题原因（猜测）：首先分析 Qtdesigner 自动生成的 UI 代码的类结构如下图

```
8
9   import socket
10  import threading
11  from PyQt5 import QtCore, QtGui, QtWidgets
12
13  class Ui_MainWindow(object):
14      threads = []
15      open_port = []
16      close_port = []
17
18      def setupUi(self, MainWindow):
19          MainWindow.setObjectName("MainWindow")
20          MainWindow.resize(825, 866)
21          self.centralwidget = QtWidgets.QWidget(MainWindow)
22          self.centralwidget.setObjectName("centralwidget")
23          self.label = QtWidgets.QLabel(self.centralwidget)
24          self.label.setGeometry(QtCore.QRect(40, 50, 81, 21))
```

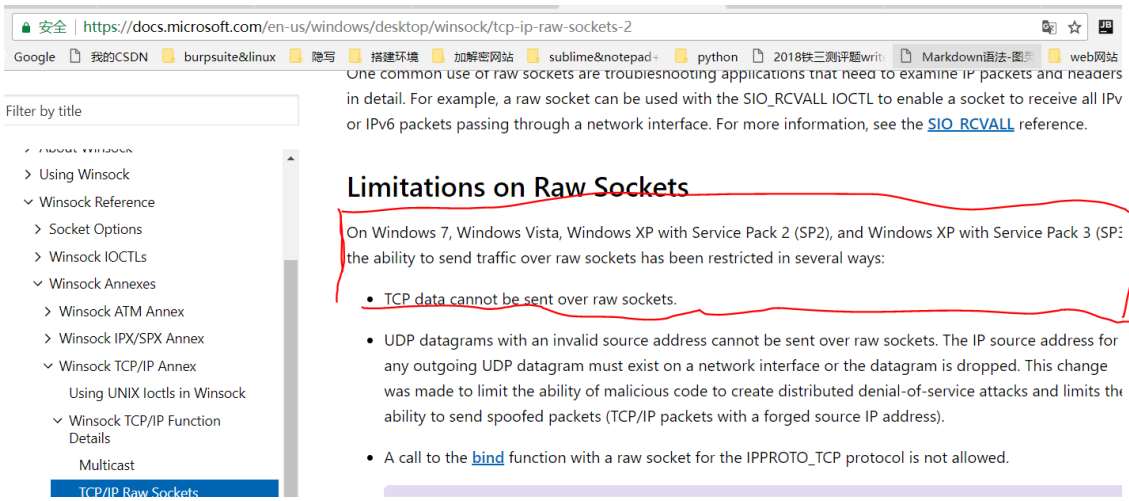
上图 UI 类生成界面，通过下图示调用方式代码调用。如下图划线处，将类 QmainWindow 实例化赋值给 mainWindow 变量，然后将 mainWindow 传入 Ui_MainWindow 类中。故猜测错误原因为传参错误，将窗口本身的函数赋值给了父

类的 MainWindow

```
8
9   if __name__ == '__main__':
10      app = QApplication(sys.argv)
11      mainWindow = QMainWindow()
12
13      ui = Ui_MainWindow()
14      ui.setupUi(mainWindow)
15      mainWindow.show()
16      sys.exit(app.exec_())
```

问题 2：在编写 SYN 扫描功能时，使用 scapy 库自己手动修改 TCP 报文发送时，发送失败。程序函数返回值显示发送成功，但 wireshark 抓不到包。

解决方法：上网查找问题，并没有找到相关类似问题，但得出几个可能原因，猜测可能是因为构造报文包时校验和未填好。于是查询相关校验和资料，发现并不是校验和的原因。中文检索找原因无果，于是使用英文搜索，搜索关键字 windows raw sockets，发现下列微软官网文档，里面说到，windows 对原始套接字做了限制，不能使用原始套接字发送 TCP data。于是只能将 SYN 扫描功能代码在 linux 中实现。



问题 3：对一个目标 IP 地址的范围端口做多线程扫描时，当线程量数量使用过多时，

程序会异常终止，如下图报错信息

```
D:\QT_lab\venv\Scripts\python.exe D:/QT_lab/venv/Yiitao_Scan.py

runtime error R6016
- not enough space for thread data

Process finished with exit code 0
```

解决方法：找到代码中关于多线程的编写部分，如下图划线处，代码编写时，将目标 IP 的关闭端口也显示在 QT 界面窗口中。造成线程对主窗口的写入竞争，于是出现所示报错。于是将图中划线处代码注释，即解决该问题。

```
117
118     def Connect(self,ipaddr,port,port_right,blank,pid):
119         #多线程
120         for i in range(port,port_right+1,blank):
121             try:
122                 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
123                 sock.settimeout(0.1)
124                 sock.connect((ipaddr,i))
125                 self.open_port.append(i)
126                 self.textEdit.append('{}:open'.format(i))
127             except:
128                 self.close_port.append(i)
129                 self.textEdit.append('{}:close'.format(i))
130                 # if blank == 1:
131                 #     self.textEdit.append('{}:close'.format(i))
132             finally:
133                 sock.close()
```

3、注册表安全防护编程实验

问题 1：访问注册表时权限不够，引发 WinError 异常

```
Traceback (most recent call last):
  File "D:/QT_lab/Labs/venv/Registry/test3.py", line 9, in <module>
    winreg.SetValueEx(key, 'Miracle',0, 1, '777')
PermissionError: [WinError 5] 拒绝访问。

Process finished with exit code 1
```

解决方法：由报错信息很容易联想到权限的设置问题，于是将 pycharm 赋予管理员权限运行。但这样只解决注册表的访问问题，修改注册表还是权限不够，猜测是函数中还有对应参数设置权限，故查阅 python 文档，找到相关修改注册表函数。发现定义如

下

winreg.OpenKey(*key*, *sub_key*, *reserved=0*, *access=KEY_READ*)

winreg.OpenKeyEx(*key*, *sub_key*, *reserved=0*, *access=KEY_READ*)

打开指定的键，返回 句柄对象。

key 是已经打开的键，或者是预定义的 HKEY_ *常数 之一。

sub_key 是一个字符串，用于标识要打开的子键。

reserved 是保留的整数，必须为零。默认值为零。

access 是一个整数，指定描述密钥所需的安全访问的访问掩码。默认值为 KEY_READ。其他允许值请参见 访问权。

结果是指定键的新句柄。

如果函数失败，则引发 OSError。

在 3.2 版更改: 允许使用命名参数。

在 3.3 版更改: 见 以上。

34.3.2.2. 访问权

有关更多信息，请参阅 注册表项安全和访问。

winreg.KEY_ALL_ACCESS

组合STANDARD_RIGHTS_REQUIRED，KEY_QUERY_VALUE，KEY_SET_VALUE，KEY_CREATE_SUB_KEY，KEY_ENUMERATE_SUB_KEYS，KEY_NOTIFY 和 KEY_CREATE_LINK 访问权限。

winreg.KEY_WRITE

组合STANDARD_RIGHTS_WRITE，KEY_SET_VALUE 和 KEY_CREATE_SUB_KEY 访问权限。

winreg.KEY_READ

组合STANDARD_RIGHTS_READ，KEY_QUERY_VALUE，KEY_ENUMERATE_SUB_KEYS 和 KEY_NOTIFY 值。

winreg.KEY_EXECUTE

相当于 KEY_READ。

winreg.KEY_QUERY_VALUE

需要查询注册表项的值。

winreg.KEY_SET_VALUE

需要创建，删除或设置注册表值。

于是更改相关注册表函数代码为下图所示，添加权限

```
    def GenerateKey(self,index,sub_path,flag = 0):
        # 遍历注册表键值,flag = 0枚举键 = 1 枚举值
        import winreg
        root_key = [winreg.HKEY_CLASSES_ROOT,
                    winreg.HKEY_CURRENT_USER,
                    winreg.HKEY_LOCAL_MACHINE,
                    winreg.HKEY_USERS,
                    winreg.HKEY_CURRENT_CONFIG]

        # 设置权限，否则不能改
        try:
            key = winreg.OpenKey(root_key[index],
                                 sub_path,
                                 0,
                                 winreg.KEY_ALL_ACCESS)
            key_list = []
            i = 0
```

问题 2：将注册表值查询出来添加至 QtreeWidget 控件子节点时，出现函数参数类型不匹配错误

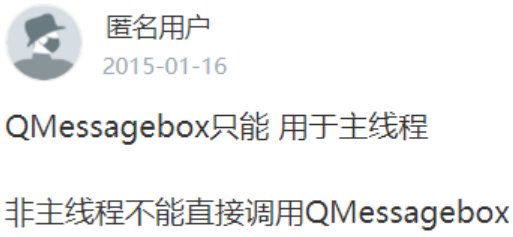
```
    def CreteValueTree(self,value_list):
        # 生成前先删除
        while self.treeWidget_2.takeTopLevelItem(0):
            pass
        for i in range(len(value_list)):
            tmpNode = QTreeWidgetItem(self.treeWidget_2)
            for j in range(3):
                # 有些b'字节串显示会出错
                tmpNode.setText(j,str(value_list[i][j]))
```

解决方法：因为要将系统注册表的值先查询保存下来再写入 QtreeWidget 控件，但系统注册表中的键值类型不一，而 QtreeWidget 子节点设置文本信息时要求必须是 str 类字符串，故发送报错，所以解决方法如上图，在添加值之前将值用 str 函数进行强制类型转换

问题 3：在编写注册表键值防护函数时，采用线程实现，本想用 QmessageBox 控件进行消息的提示交互。但程序会异常卡死奔溃。

```
84      def Listen(self,default,key):
85          while self.flags == 1:
86
87              try:
88                  value,reg_type = winreg.QueryValueEx(key,default['name'])
89                  if value != default['value'] or reg_type != default['type']:
90                      # 线程里不让MessageBox,错误
91                      # msgBox = QMessageBox()
92                      # msgBox.setWindowTitle("防护注册表")
93                      winreg.SetValueEx(key,default['name'],0,default['type'],default['value'])
94                      print("检测到有程序正在修改注册表\n\t已恢复默认值")
95                      # exec_阻塞进程错误
96                      # msgBox.exec()
97              except Exception as e:
98                  print("Listen线程",e)
```

解决方法：上网查询，得出原因如下图，于是放弃使用 QmessageBox 控件进行交互，改用 print



4、恶意代码及防护编程实验

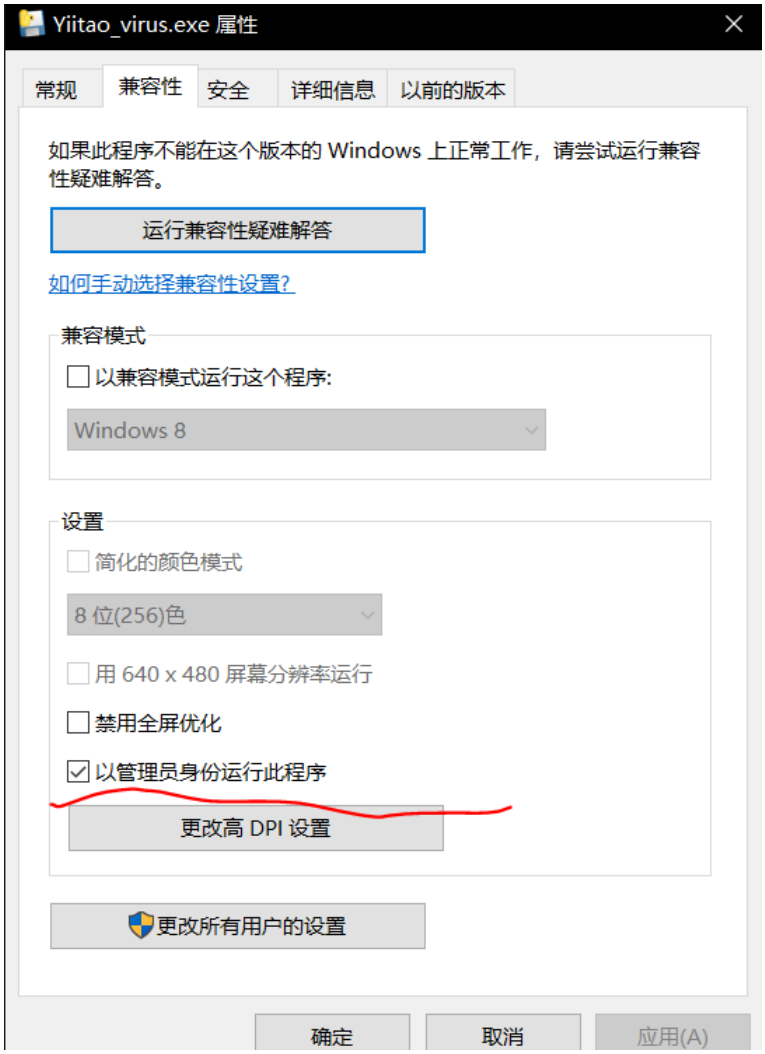
问题 1：编写恶意代码双击 txt 文件删除功能时，需要更改注册表 txt 文件关联程序路径为恶意代码程序路径。但更改注册表完成后，双击 txt 文件会打开指定恶意程序，但是 txt 文件并不会被删除。

解决方法：双击 txt 文件会启动指定恶意程序，排除注册表未写入错误。思考分析双击文件时，必须给程序指定文件路径才能对其进行操作。故怀疑是更改注册表键值时忘记添加参数标记，故查看默认注册表关联键值写法。如下图，默认 txt 文件关联应用程序为 Notepad.exe，但注册表键值中，在其路径后加了 %1 ，猜测此标记作用为传参，故修改代码如下图 2。更改后测试，问题得到解决

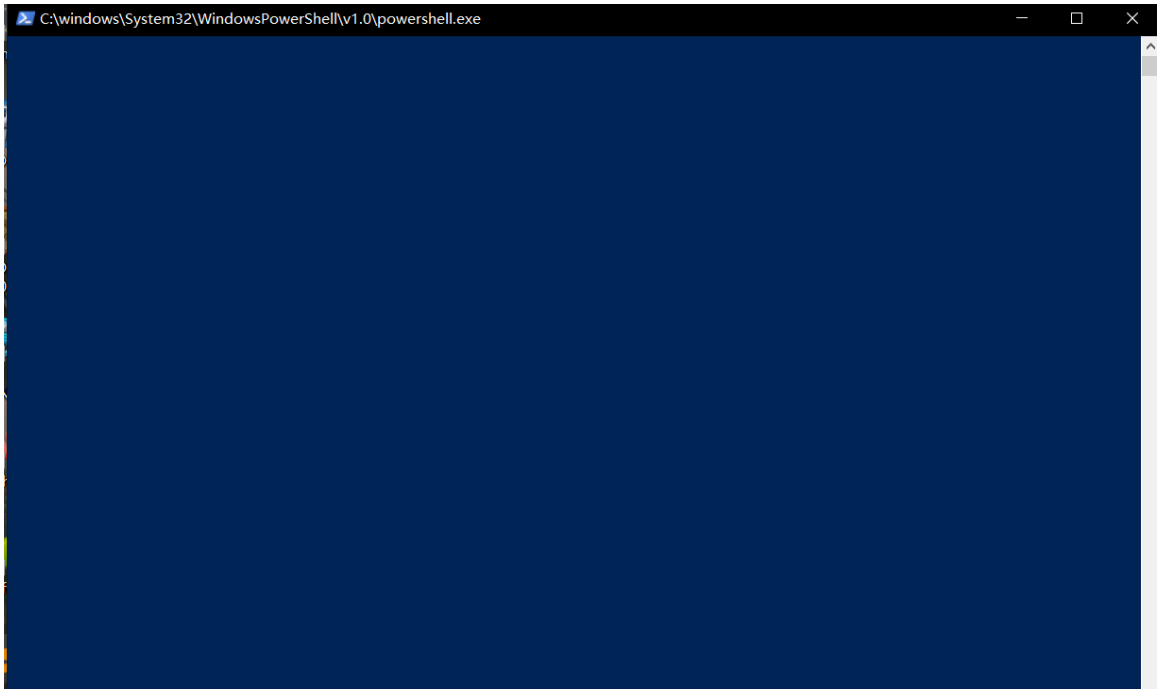

```
111                                    winreg.KEY_ALL_ACCESS)
112
113             # 更改txt文件打开默认程序
114             # 错误 添加注册表要加 %1 否则无法获取文件路径
115             winreg.SetValueEx(key, '', 0, 2, self.self_path + ' %1')
116
```

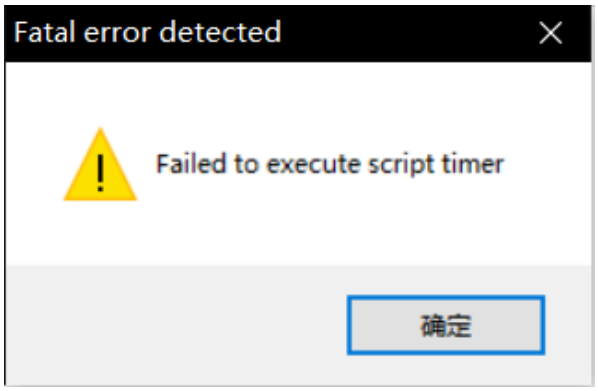
问题 2：编写自启动功能时，需要对注册表中与系统启动项相关键值进行添加，使用 pycharm 编译测试时，能够正常执行，但打包成 exe 后，运行 exe 不能。

解决方法：实现程序自启动功能，需要对注册表中与系统启动项相关键值进行修改，需要管理员权限才能正常执行，所以还需要对打包生成的 exe 加上管理员权限运行。



问题 3：编写远程控制功能发送网卡抓包信息时，需要用到 scapy 库，使用 pycharm 能够运行，但打包成 exe 文件后，运行报错，报错详细情况如下两图

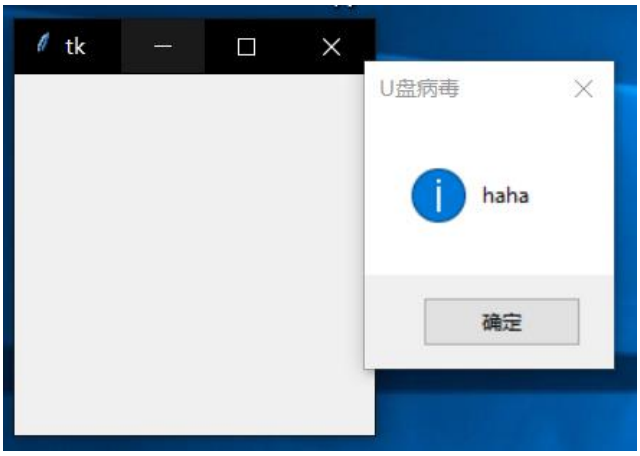


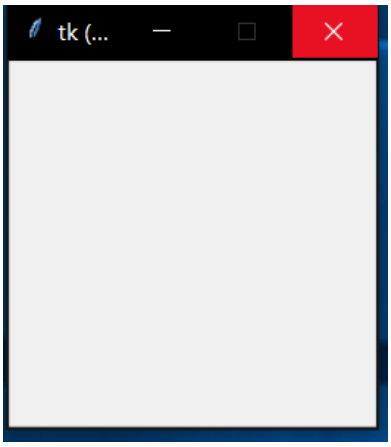


解决方法：因为使用 python 打包成 exe 文件的库很多，我选用的是比较流行的 pyinstaller 第三方库，可能对 scapy 库打包不兼容。于是只能将远程控制主机发送网卡抓包功能使用 pycharm 以脚本方式运行。

5、U 盘病毒

问题 1： 实现在程序在 U 盘上运行时弹框错误，会出现 tk 主窗口，且主窗口会阻塞卡死





解决方法：因为是初次使用 tkinter 库，不熟悉其用法，所以猜测可能是代码调用方法出错，于是针对出现的异常情况（出现 tk 主窗口、tk 主窗口卡死）等两个问题特定搜索。

于是，找到 tkinter 隐藏主窗口的方法，使用 withdraw 函数

弹完子窗口后，调用 destory 方法，并将 **mainloop** 注释掉

mainloop 函数中有着事件处理的循环和 Tkinter 窗口生命周期的控制

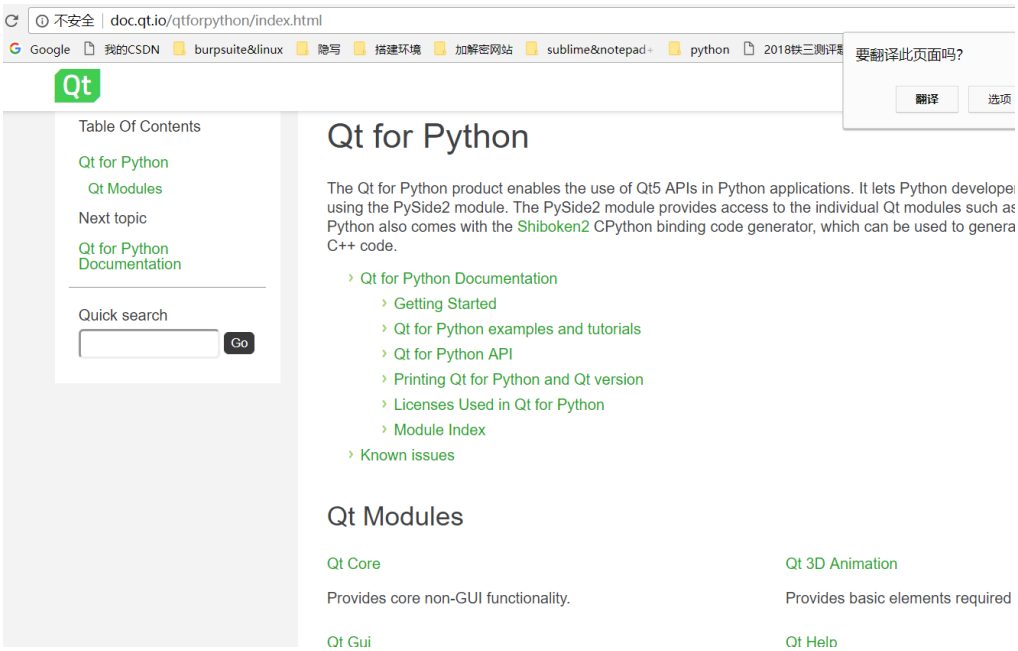
```
def MesgBox():
    root = tkinter.Tk()
    root.withdraw()  # ****实现主窗口隐藏
    for i in range(5):
        tkinter.messagebox.showinfo("U盘病毒",'haha')
        # time.sleep()
    root.destroy()
    # root.mainloop()
```

因为 U 盘病毒编写较为轻松，代码量较少，故没有遇到其他大问题，因此就不再赘述

五、 实验总结及心得

1、本学期在这门实验课程中，尝试了用 python 写界面编程，学会了 pyqt5 的简单使用。从 pyqt 的安装开始，从一开始一个控件都不会写到现在能够很轻松快速的写出一个基本界面。还是有很大进步的。

2、回想起编写用 pyqt5 编写第一个实验时，各种控件都还不会用，代码不知道如何调用，网上搜索也是各说风云，得不到自己想要的答案。于是选择了查询 pyqt 官方文档，官方文档有点是比较全面，各种函数定义都能找到，但缺点就是是全英文，且很多函数如何调用没有具体小例子，很难理解。于是就通过官方文档查找能够实现自己功能的函数，再通过搜索具体函数的用法学习。通过这种学习方法，不仅 pyqt5，其他几个实验用到的以前从没见过的库也能快速学习使用。**学会了一种高效快速的编程学习方法，对日后的工作学习有很大帮助。**



3、因为用到 pyqt5 编程，需要用到很多类，编写类函数。对类的理解更深入了，也**深刻体会**到类的方便（类里面的变量共享和继承复用等）。以前学习类的时候，不太习惯使用，但现在体会到类的方便后，在以后的代码编写中，会多多使用面向对象编程

4、学会了一些 python 的高级用法。包括如何打包成 exe、多线程、协程、python 操控 windows api、操控注册表、python GUI、网络编程等等

5、通过自己编写注册表、恶意代码、U 盘病毒等程序，了解到一些恶意程序的实现流程，如何自启动、隐藏进程、定时执行等。