18 min. read

Python程序设计作业#6

2021-07-16 | 2020-11-30 | 2 | 0

Python程序设计#6作业

截止时间：2020年11月30日23:59:59

作业题目

在作业#5的基础上实现localProxy的图形管理界面localGui

localGui单独一个源文件

可通过图形界面（可以使用QDialog）关闭和开启localProxy

界面上提供remoteProxy的主机地址和端口、认证的用户名和密码（掩码显示）

建议使用QProcess类管理localProxy进程

可以实时查看localProxy的运行状态（是否运行、实时吞吐率）

localGui与localProxy之间采用WebSocket连接（localGui为client）

localProxy代码

localProxy代码嵌入下方的code block中。

```
import asyncio
import struct
```

```python
import socket
import logging
logging.basicConfig(level=logging.INFO)
import nest_asyncio
nest_asyncio.apply()
import sys
import getopt
import websockets
import argparse
VERSION = 5

gSendBandwidth = 0.0
gRecvBandwidth = 0.0
updateBytes = 0
downloadBytes = 0
remoteHost = ''
remotePort = 0
async def socks5(first, reader, writer):
    addr_from = writer.get_extra_info('peername')
    logging.info(f'connect from{addr_from}')
    header = await reader.read(1)
    header = first + header
    ver, num_method = struct.unpack("!BB", header)
    logging.info(f'ver == VERSION:{ver == VERSION}')
    logging.info('num_method = %d' % num_method)
    methods = []
    for i in range(num_method):
        methods.append(ord(await reader.read(1)))
    if 0 not in methods:#无需认证
        writer.close()
        writer.wait_closed()
        return
    #回应一个数据包，包括协议版本号，指定认证方法
    writer.write(struct.pack("!BB", VERSION, 0))
    await writer.drain()
    request = await reader.read(4)
    ver, cmd, rsv, atype = struct.unpack("!BBBB", request)
    assert ver == VERSION
    #ipv4
    if atype == 1:
        address = socket.inet_ntoa(await reader.read(4))
    #域名
    elif atype == 3:
        domain_length = await reader.read(1)
        address = await reader.read(domain_length[0])
    #ipv6
    elif atype == 4:
```

```
        address = socket.inet_ntop(socket.AF_INET6, await reader.read(16))
    else:
        writer.close()
        writer.wait_closed()
        return
    port = struct.unpack('!H', await reader.read(2))
    try:
        if cmd == 1:
            reader_remote,writer_remote = await asyncio.open_connection(remoteHos
            http_connect = 'CONNECT ' + address + ':' + str(port[0]) + ' HTTP/1.1
            print('http_connect')

            http_connect += ' %' + username + '%' + password + '%'
            print(http_connect)
            writer_remote.write(http_connect.encode())
            await writer_remote.drain()
            reply = await (reader_remote.read(1024))
        else:
            writer.close()
            writer.wait_closed()
    except Exception as error:
        logging.error(error)
    reply = struct.pack("!BBBBIH", VERSION, 0, 0, 1, 0, 0)
    writer.write(reply)
    await writer.drain()

    #第一个字节为0表示成功代理
    if cmd == 1 and reply[1] == 0:
        tasks = [read_trans(reader, writer_remote), write_trans(reader_remote, wr
        await asyncio.wait(tasks)

async def read_trans(reader, writer_remote):
    global updateBytes
    while True:
        data = await reader.read(4096) #上传
        updateBytes += len(data)
        if not data:
            logging.info('disconnect')
            break
        writer_remote.write(data)
        await writer_remote.drain()

async def write_trans(reader_remote, writer):
    global downloadBytes
    while True:
        data = await reader_remote.read(4096) # 下载
        downloadBytes += len(data)
```

```
        if not data:
            logging.info('disconnect')
            break
        writer.write(data)
        await writer.drain()

async def httptunnel(first, reader, writer):
    http_connect = (await reader.read(1024))
    http_connect = (first + http_connect).decode()
    http_connect += ' %' + username + '%' + password + '%'
    logging.info(http_connect)

    reader_remote,writer_remote = await asyncio.open_connection(remoteHost,remote
    writer_remote.write(http_connect.encode())
    await writer_remote.drain()


    reply = await (reader_remote.read(1024))
    writer.write(reply)
    await writer.drain()
    #连接建立成功
    tasks = [read_trans(reader, writer_remote), write_trans(reader_remote, writer
    await asyncio.wait(tasks)

async def test(reader, writer):
    first = await reader.read(1)
    if(first == b'\x05'):
        await socks5(first, reader, writer)
    elif(first == b'C'):
        await httptunnel(first, reader, writer)

username = ''
password = ''

# async def main():
#     global username, password
#     if(len(sys.argv) != 3):
#         logging.info('usage: local-proxy.py username, password')
#     else:
#         username = sys.argv[1]
#         password = sys.argv[2]
#     print(username)

#     print(password)
#     server = await asyncio.start_server(test, '0.0.0.0', 10086)
#     async with server:
#         await server.serve_forever()
```

```python

async def localConsole(ws, path):
    global gSendBandwidth #全局变量，表示当前上传带宽
    global gRecvBandwidth #全局变量，表示当前下载带宽
    try:
        while True:
            await asyncio.sleep(1) #每隔一秒向gui发送当前速率
            print(gSendBandwidth)
            print(gRecvBandwidth)
            msg = await ws.send(f'{gSendBandwidth} {gRecvBandwidth}')
            print(msg)
    except websockets.exceptions.ConnectionClosedError as exc:
        logging.error(f'{exc}')
    except websockets.exceptions.ConnectionClosedOK as exc:
        logging.error(f'{exc}')
        exit(1)

async def calcBandwidth(): #计算带宽函数
    global gSendBandwidth
    global gRecvBandwidth
    global updateBytes
    global downloadBytes
    while True:
        gSendBandwidth = updateBytes / 0.5  # Bps
        gRecvBandwidth = downloadBytes / 0.5
        updateBytes = 0
        downloadBytes = 0
        print(gSendBandwidth)
        print(gRecvBandwidth)
        await asyncio.sleep(0.5)#每0.5s计算一次带宽

async def localTask():
    _parser = argparse.ArgumentParser(description='socks5 https dual proxy')
    _parser.add_argument('-p', dest='listenPort', metavar='listen_port', required
    _parser.add_argument('-u', dest='username', metavar='username', required=True
    _parser.add_argument('-w', dest='password', metavar='password', required=True
    _parser.add_argument('-k', dest='consolePort', metavar='consolePort', require
    _parser.add_argument('remoteHost', nargs='?', help='remote host in local mode
    _parser.add_argument('remotePort', nargs='?', type=int, help='remote port in
    args = _parser.parse_args()
    # if(len(sys.argv) != 7):
    #     logging.info('usage: local-proxy.py ip, port, username, password, webso
    #     return
    global username, password, remoteHost, remotePort
    #if args.consolePort: # 这是localproxy的websocket监听端口
    ws_server = await websockets.serve(localConsole, '127.0.0.1', int(args.consol
    logging.info(f'CONSOLE LISTEN {ws_server.sockets[0].getsockname()}')
```

```

    asyncio.create_task(calcBandwidth()) # calcBandwidth()函数计算带宽
    username = args.username
    password = args.password
    remoteHost = args.remoteHost
    remotePort = args.remotePort
    srv = await asyncio.start_server(test, '127.0.0.1', int(args.listenPort)) #ip

    async with srv:
        await srv.serve_forever()
#gui传过来的参数有 ip 端口号 用户名 密码 websockets端口
asyncio.run(localTask())
```

localGui代码

localGui代码嵌入下方的code bock中。

```
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtNetwork import *
from PyQt5.QtWidgets import *
from PyQt5.QtWebSockets import *
from mainwindow import Ui_MainWindow
import sys
from PyQt5 import QtWidgets, uic
import logging
import os
import humanfriendly
class MainWindow(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        super(MainWindow,self).__init__(parent)
        #self.pushButton = QPushButton('start') #注意一会把
        uic.loadUi("mainwindow.ui", self) # 加载界面
        self.pushButton.clicked.connect(self.startClicked)

        self.process = QProcess()
        self.process.setProcessChannelMode(QProcess.MergedChannels)
        #self.process.finished.connect(self.processFinished)#当进程结束，触发process
        self.process.started.connect(self.processStarted)# 当进程已经开始了，触发pro
        self.process.readyReadStandardOutput.connect(self.processReadyRead) #信号

    def processReadyRead(self):
        data = self.process.readAll()
        try:
```

```
            msg = data.data().decode().strip()
            logging.debug(f'msg={msg}')
        except Exception as exc:
            #logging.error(f'{traceback.format_exc()}')
            exit(1)

    def processStarted(self):  #进程开始后，调用该函数
        process = self.sender() # 此处等同于 self.process 只不过使用sender适应性更好
        processId = process.processId()
        logging.debug(f'pid={processId}')
        self.pushButton.setText('stop')
        #self.processIdLine.setText(str(processId))  #先注释掉这里

        self.websocket = QWebSocket()
        self.websocket.connected.connect(self.websocketConnected)
        self.websocket.disconnected.connect(self.websocketDisconnected)
        self.websocket.textMessageReceived.connect(self.websocketMsgRcvd) #当收到对
        self.websocket.open(QUrl(f'ws://127.0.0.1:{self.consolePortLine.text()}/')

    def startClicked(self):  #当点击开始按钮时
        btn = self.sender()
        text = btn.text().lower()
        if text.startswith('start'):
            listenPort = self.listenPortLine.text()  #本地端口10086
            username = self.usernameLine.text()      #用户名
            password = self.passwordLine.text()      #密码
            consolePort = self.consolePortLine.text() #websockts端口
            remoteHost = self.remoteHostLine.text() #远程主机ip
            remotePort = self.remotePortLine.text() #远程主机端口
            pythonExec = os.path.basename(sys.executable)
            # 从localgui启动localproxy直接使用-w 提供用户密码，不再使用命令行交互输入，
            cmdLine = f'{pythonExec} local-proxy.py -p {listenPort} -u {username}
            print(cmdLine)
            logging.debug(f'cmd={cmdLine}')
            self.process.start(cmdLine)
        else:
            self.process.kill()
    def websocketConnected(self):
        self.websocket.sendTextMessage('secret') #连接建立后随便发的

    def websocketDisconnected(self):
        self.process.kill()

    #def format_bandwidth()

    def websocketMsgRcvd(self, msg):
        logging.debug(f'msg={msg}')
```

```
75          sendBandwidth, recvBandwidth, *_ = msg.split()
76          nowTime = QDateTime.currentDateTime().toString('hh:mm:ss')
77          print(sendBandwidth)
78          print(recvBandwidth)
79          self.sendBandwidthLine.setText(f'{nowTime} {sendBandwidth} Bps')
80          self.lineEdit_2.setText(f'{nowTime} {recvBandwidth} Bps')
81  app = QApplication(sys.argv)
82  app.aboutToQuit.connect(app.deleteLater)
83  form = MainWindow()
84  form.show()
85  app.exec_()
```

代码说明

源代码中不要出现大段的说明注释，所有文字描述在本节中以行号引用说明。

Python程序设计作业#5 Python程序设计作业#8

昵称 邮箱 网址(http://)

Just go go

提交

来发评论吧~

Powered By Valine
v1.4.14