7 min. read

Python程序设计作业#4

2021-07-16 | 2020-11-18 | 1 | 0

Python程序设计#4作业

截止时间：2020年11月16日23:59:59

作业题目

在作业#3的基础上实现localProxy命令行参数账号登录

在作业#3的基础上实现remoteProxy多账号认证

remoteProxy采用SQLite3数据库进行用户账号管理（用户名、密码）

remoteProxy使用aiosqlite操作SQLite3数据库

作业内容

程序源代码嵌入下方的code block中。

local_proxy

```
1  import asyncio
2  import struct
3  import socket
4  import logging
5  logging.basicConfig(level=logging.INFO)
6  import nest_asyncio
7  nest_asyncio.apply()
```

```
import sys
import getopt
VERSION = 5
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('127.0.0.
            http_connect = 'CONNECT ' + address.decode() + ':' + str(port[0]) + '
            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):
    while True:
        data = await reader.read(4096)
        if not data:
            logging.info('disconnect')
            break
        writer_remote.write(data)
        await writer_remote.drain()

async def write_trans(reader_remote, writer):
    while True:
        data = await reader_remote.read(4096)
        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('127.0.0.1',10010
    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()

    asyncio.run(main())
```

remote-proxy

```
import asyncio
import struct
import socket
import logging
logging.basicConfig(level=logging.INFO)
import nest_asyncio
nest_asyncio.apply()
import aiosqlite
async def handle(reader_local, writer_local):
    db = await aiosqlite.connect('account.db')
```

```
11      http_connect = (await reader_local.read(1024))
12      http_connect = http_connect.decode()
13      logging.info(http_connect)
14
15      i = 0
16      while(http_connect[i] != ':'):
17          i += 1
18      domain_name = http_connect[8 : i]
19      j = i
20      while(http_connect[j] != ' '):
21          j += 1
22      port = http_connect[i + 1 : j]
23      i = 0
24      while(http_connect[i] != '%'):
25          i += 1
26      j = i + 1
27      while(http_connect[j] != '%'):
28          j += 1
29      k = j + 1
30      while(http_connect[k] != '%'):
31          k += 1
32      username = http_connect[i + 1: j]
33      password = http_connect[j + 1: k]
34      print(username)
35      print(password)
36      sql = 'SELECT * FROM accout where username = \'' + username + '\' and password
37      print(sql)
38      cursor = await db.execute(sql)
39      row = await cursor.fetchall()
40      await cursor.close()
41      if(len(row) != 1):
42          logging.error('wrong account')
43          return
44      else:
45          logging.info('right account')
46      reader_remote,writer_remote = await asyncio.open_connection(domain_name,port)
47
48      reply = 'HTTP/1.1 200 OK\r\n\r\n'
49      writer_local.write(reply.encode())
50      await writer_local.drain()
51
52      tasks = [read_trans(reader_local, writer_remote), write_trans(reader_remote, w
53      await asyncio.wait(tasks)
54          await db.close()
55
56  async def read_trans(reader, writer_remote):
57      while True:
```

```
58          data = await reader.read(4096)
59          if not data:
60              logging.info('disconnect')
61              break
62          writer_remote.write(data)
63          await writer_remote.drain()
64
65  async def write_trans(reader_remote, writer):
66      while True:
67          data = await reader_remote.read(4096)
68          if not data:
69              logging.info('disconnect')
70              break
71          writer.write(data)
72          await writer.drain()
73  async def main():
74      server = await asyncio.start_server(handle, '127.0.0.1', 10010)
75      async with server:
76          await server.serve_forever()
77
78  asyncio.run(main())
```

代码说明（可选）

源代码中不要出现大段的说明注释，如果需要可以可以在本节中加上说明。

Python程序设计作业#7 Python程序设计作业#2

来发评论吧~

Powered By Valine
v1.4.14