6 min. read

Python程序设计作业#3

2021-07-16 | 2020-11-10 | 2 | 0

Python程序设计#3作业

截止时间：2020年11月09日23:59:59

作业题目

实现localProxy和remoteProxy分离式代理。

支持SOCKS5代理和HTTPS代理（基于#2作业的成果）。

localProxy收到的每个TCP连接单独建立代理TCP连接。

作业内容

程序源代码嵌入下方的code block中。

local proxy

```
import asyncio
import struct
import socket
import logging
logging.basicConfig(level=logging.INFO)
import nest_asyncio
nest_asyncio.apply()
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')
            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()
```

```
57      except Exception as error:
58          logging.error(error)
59      reply = struct.pack("!BBBBIH", VERSION, 0, 0, 1, 0, 0)
60      writer.write(reply)
61      await writer.drain()
62
63      #第一个字节为0表示成功代理
64      if cmd == 1 and reply[1] == 0:
65          tasks = [read_trans(reader, writer_remote), write_trans(reader_remote, wr
66          await asyncio.wait(tasks)
67
68  async def read_trans(reader, writer_remote):
69      while True:
70          data = await reader.read(4096)
71          if not data:
72              logging.info('disconnect')
73              break
74          writer_remote.write(data)
75          await writer_remote.drain()
76
77  async def write_trans(reader_remote, writer):
78      while True:
79          data = await reader_remote.read(4096)
80          if not data:
81              logging.info('disconnect')
82              break
83          writer.write(data)
84          await writer.drain()
85
86  async def httptunnel(first, reader, writer):
87      http_connect = (await reader.read(1024))
88      http_connect = (first + http_connect).decode()
89
90      logging.info(http_connect)
91
92      reader_remote,writer_remote = await asyncio.open_connection('127.0.0.1',10010
93      writer_remote.write(http_connect.encode())
94      await writer_remote.drain()
95
96
97      reply = await (reader_remote.read(1024))
98      writer.write(reply)
99      await writer.drain()
100     #连接建立成功
101     tasks = [read_trans(reader, writer_remote), write_trans(reader_remote, writer
102     await asyncio.wait(tasks)
103
```

```
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)

async def main():
    server = await asyncio.start_server(test, '127.0.0.1', 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()

async def handle(reader_local, writer_local):
    http_connect = (await reader_local.read(1024))
    http_connect = http_connect.decode()
    logging.info(http_connect)
    i = 0
    while(http_connect[i] != ':'):
        i += 1
    domain_name = http_connect[8 : i]
    j = i
    while(http_connect[j] != ' '):
        j += 1
    port = http_connect[i + 1 : j]

    reader_remote,writer_remote = await asyncio.open_connection(domain_name,port)

    reply = 'HTTP/1.1 200 OK\r\n\r\n'
    writer_local.write(reply.encode())
    await writer_local.drain()

    tasks = [read_trans(reader_local, writer_remote), write_trans(reader_remote, w
```

```
    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 main():
    server = await asyncio.start_server(handle, '127.0.0.1', 10010)
    async with server:
        await server.serve_forever()

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。

本地上传到algolia Python程序设计作业#5

提交

来发评论吧~