

**（深圳）**

Harbin Institute of Technology, Shenzhen

Advanced Computer Networking

Experiment Report

**Group Members:**

姓名 庄舒漫 学号22S15118

姓名 唐栋 学号22S151060

报告日期: 2022.06.09

# 1. Experiment Overview

## Experimental tasks

Experiment overview Experimental tasks Only project1 was implemented write a program that can capture the coming packets and revert the webpage which you select to test by using the captured data.

## Experimental steps

Capture the packets

Reconstruct http message

Revert the webpage with captured data

## Development platform

Os: win10

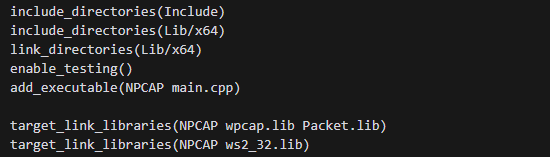
Compilation tools: mingw64,npcap,python3.9,cmake

Visualization platform：QT5

# 2. Experimental process

## 1. Npcap configuration

Put the corresponding npcap dynamic link library and function header file into the project folder on the vscode platform

Use statement：

Note that dynamic linking should be after the add\_executable statement, while loading static libraries include/link should be before this statement

Make will generate an executable file for the win platform



## 2. Capture packets according to npcap function

1. pcap\_findalldevs

NpCap provides the pcap\_findalldevs\_ex and pcap\_findalldevs functions to get a list of network interface devices on the computer; this function assigns a value to the passed-in pcap\_if\_t-the type is a linked list header that represents a device list; each such node contains name and description fields to describe the device. In addition, the pcap\_if\_t structure also contains a pcap\_addr structure; the latter contains a list of addresses, a list of masks, a list of broadcast addresses and a list of destination addresses; in addition, pcap\_findalldevs\_ex can also return remote adapter information and a pcap file list located in the given local folder.

2. pcap\_open

Used to open an adapter, actually calling pcap\_open\_live

it accepts five parameters：

**name：**The name of the adapter (GUID)

**snaplen：**Specify which parts of the captured packet to capture. In some operating systems (such as xBSD and Win32), the driver can be configured to only capture the initial part of the packet: this can reduce the amount of data copied between applications, thereby improving capture efficiency; in this experiment, the value is set to 65535, which is larger than the largest MTU that can be encountered, so you can always receive complete packets.

**flags：**The main meaning is that it contains a promiscuous mode switch; in general, the adapter only receives packets sent to itself, and those packets that communicate between other machines will be discarded. But promiscuous mode will capture all packets-because we need to capture packets from other adapters, we need to turn on this switch.

**to\_ms：** Specifies the timeout for reading data, in milliseconds; when using other APIs to read data on the adapter, these functions will respond within the time set here-even if there are no packets or capture fails; in statistical mode, to\_ms can also be used to define the statistical time interval: setting it to 0 means no timeout-if no packets arrive, it will never return; corresponding to -1: read operation returns immediately.

**errbuf：**A buffer for storing error message strings The function returns a handle of type pcap\_t.

3. pcap\_loop

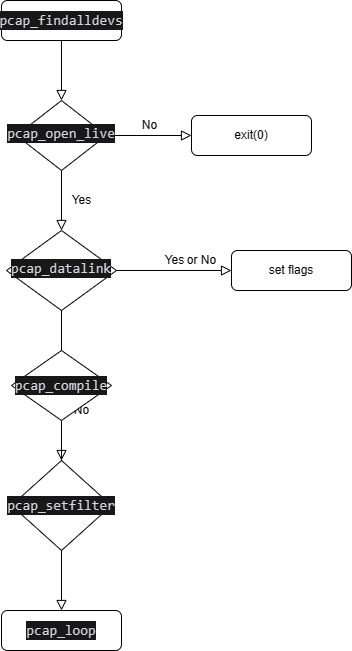
The API functions pcap\_loop and pcap\_dispatch are both used to capture packets in an opened adapter; but the former will know how to capture until the number of captured packets reaches the required number, while the latter will return after reaching the timeout set by the previous API (although this is not guaranteed); The former will block network applications for a short period of time, so general projects will use the latter as a function for reading packets; although in this experiment, using the former is enough. These two functions have a callback function; this callback function will be called when these two functions capture packets, used to process captured packets; this callback function needs to follow a specific format. But we need to note that we cannot find CRC redundancy check code-because after the frame arrives at the adapter, it will go through a check confirmation process; if this process succeeds, then the adapter will delete CRC; otherwise, most adapters will delete The whole package, so it cannot be confirmed by NpCap.

4. pcap\_datalink

Used to detect MAC layer, to ensure that it is processing an Ethernet, ensure MAC header is 14 bits long. The IP packet header is located behind MAC header, and source IP address and destination IP address are parsed from IP packet header. In this experiment, we mainly captured Ethernet packets and local loopback network packets. Since these two types of packets have different header formats, we must use this function to distinguish them and set flags.

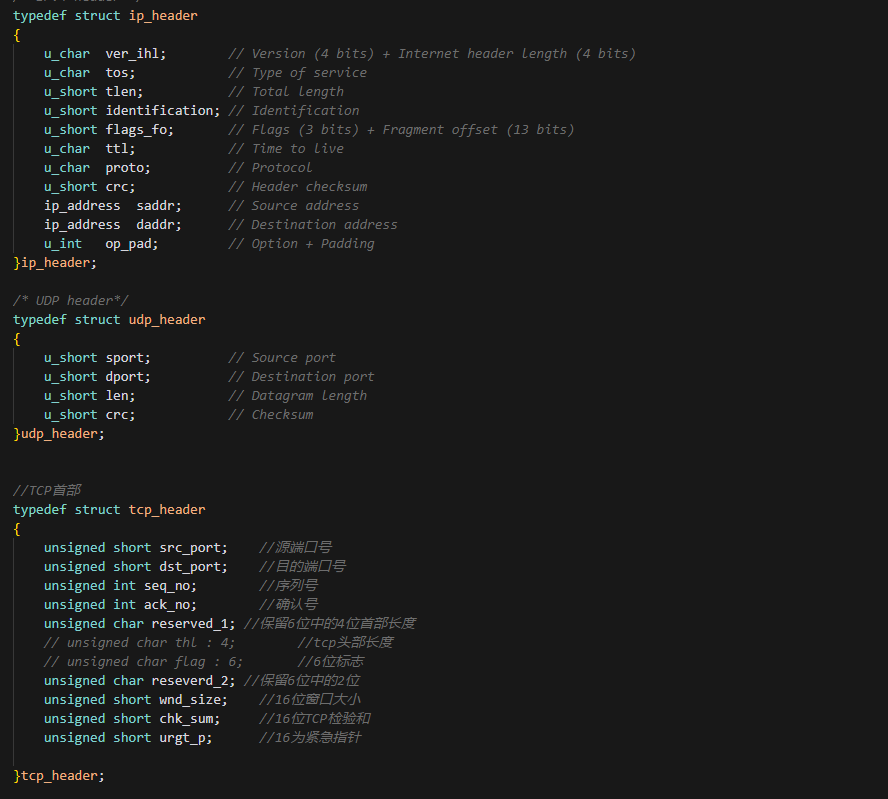
5. pcap\_compile & pcap\_setfilter

Used to set filters to avoid processing some useless packages and improve package processing efficiency. In this experiment we need to be able to enter filter rules according to filter statement format



## 3. Define network layer, transport layer, and application layer protocol header information

The message definition is as follows：



1. Here are some points to note：  
   Fields that exceed one byte definition should pay attention to endian, endian refers to the byte sorting method, big endian high data in low address, small endian low data in low address, pay attention to using windows-specific functions (not in mingw compiler) ,ntohs(short),ntohl(long integer)(win\_32), within a byte just take the value you need by bit
2. Regarding the datagram frame header, Ethernet is 14 bytes, and the local loopback packet is 4 bytes.
3. The minimum frame size of Ethernet is 60, and 0x00 padding will be used if it is smaller than this
4. The maximum packet length of Ethernet is 1514 = 54 + 1460 (data size)

## 4. Parse http packet content





Header information to note:  
**content\_type:** concerned about whether it is an html/txt text page or a png,jpeg image format, or a .css web renderer;

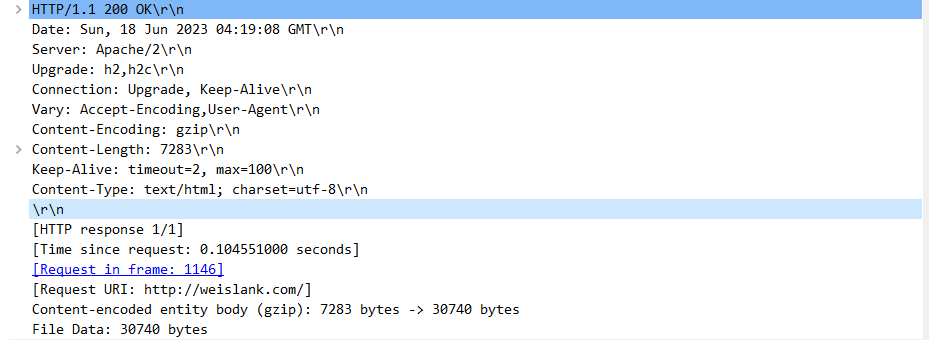
**Content\_encoding:** Now http web pages use gzip compression to speed up response, so you need to determine whether it is gzip compressed and decompress

**Charset:** mainly for what encoding binary string after decompression

**Content\_length:** to determine whether the entire contents of the packet are obtained

Example：  
Here I captured a website using the http protocol of ipv4 protocol, the website is [www.weislank.com](file:///D:\WeChat\wechat_files\WeChat%20Files\wxid_hyhfyueldjno22\FileStorage\File\2023-06\www.weislank.com)

Capture http response message content as shown below：



Using NotePad++ open it



At this time, the length size is indeed 7283.

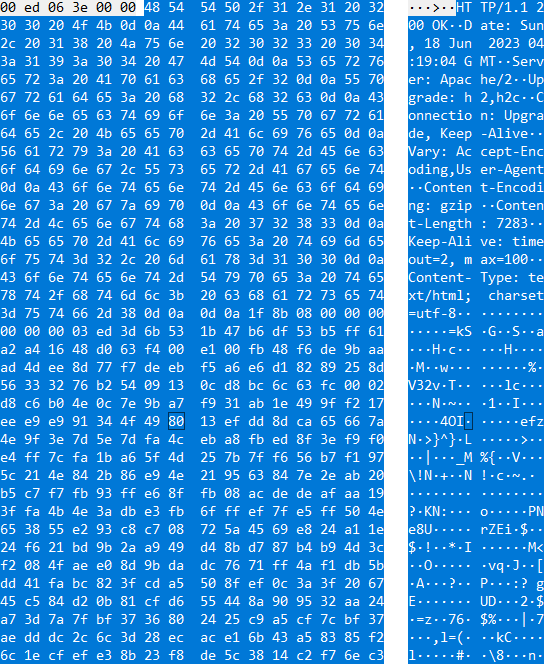
After decompressing with gzip and using utf-8 encoding, I got the html file as follows：

## 5.Reassemble datagram

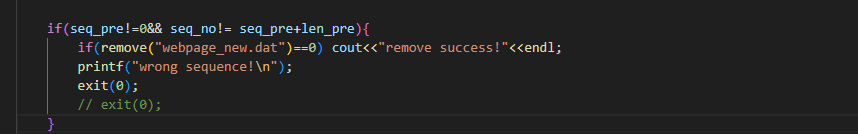
 Although the http datagram content obviously cannot be sent with one window, I found that the server did not fragment the ip datagram, but used different tcp packets + fin pieces to send the entire datagram So, the program implementation logic is as follows：

1. Find the first response packet

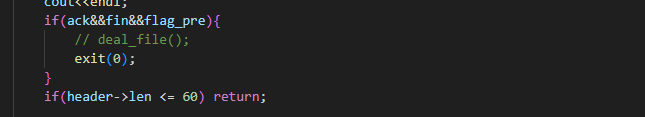
Packets starting with HTTP, remove the http protocol header information, write the data part content to a binary file



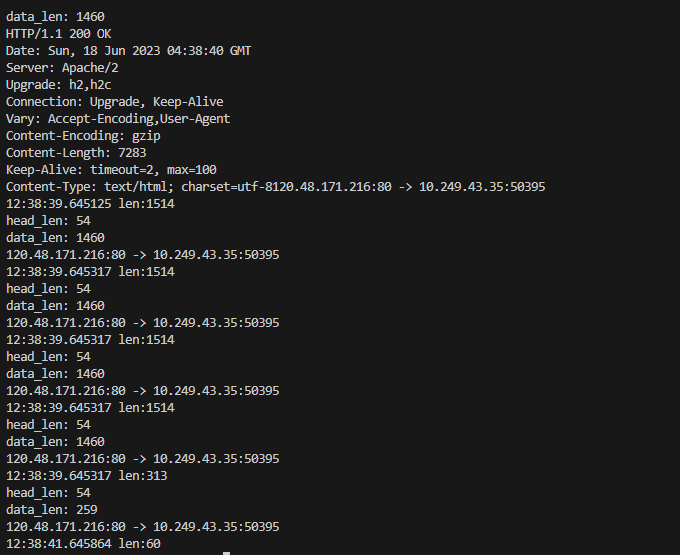
1. Determine whether the next sequence number == previous packet sequence number + data size (1460)



1. Determine whether fin flag is true, if true, and data size == message size, exit program

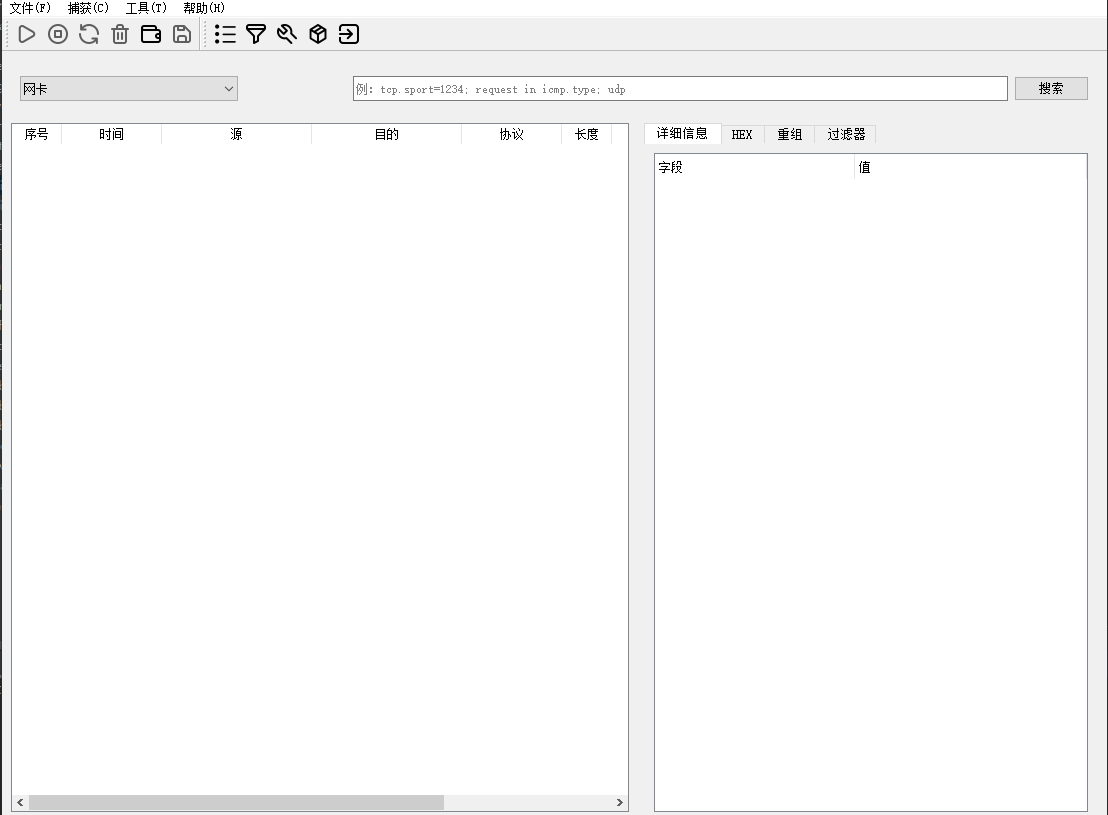


The final reassembled package information is：



## 6. Visual interface

Visualization is written in python, which is relatively simple and fast to implement. It mainly uses pyqt5 The window is as follows：



# Experimental results

## 1. Experimental conditions

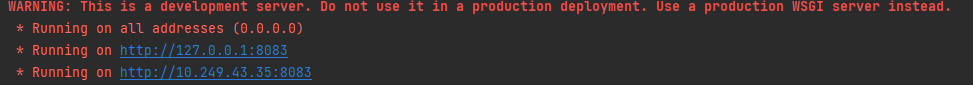
1.1  Operating conditions：

Access public website：[www.weislank.com](http://www.weislank.com)

Network card：Ethernet card

Filter statement： src host 120.48.171.216

Access your own web page：

 Network card： local loopback capture network card

Filter statement：src port 8083

1.2 Project structure：  
 Divided into c++ files and python files

a. C++ file function is to capture packets, reassemble packets, and analyze web page content

Specific running method：  
Run：The NPCAP.exe file under the C:\Users\18399\Desktop\project1\npcap\build path can be run, pay attention to network card selection and filter statement writing

b. Python files are mainly for visualization, page loading, and packet decompression

Specific running method：

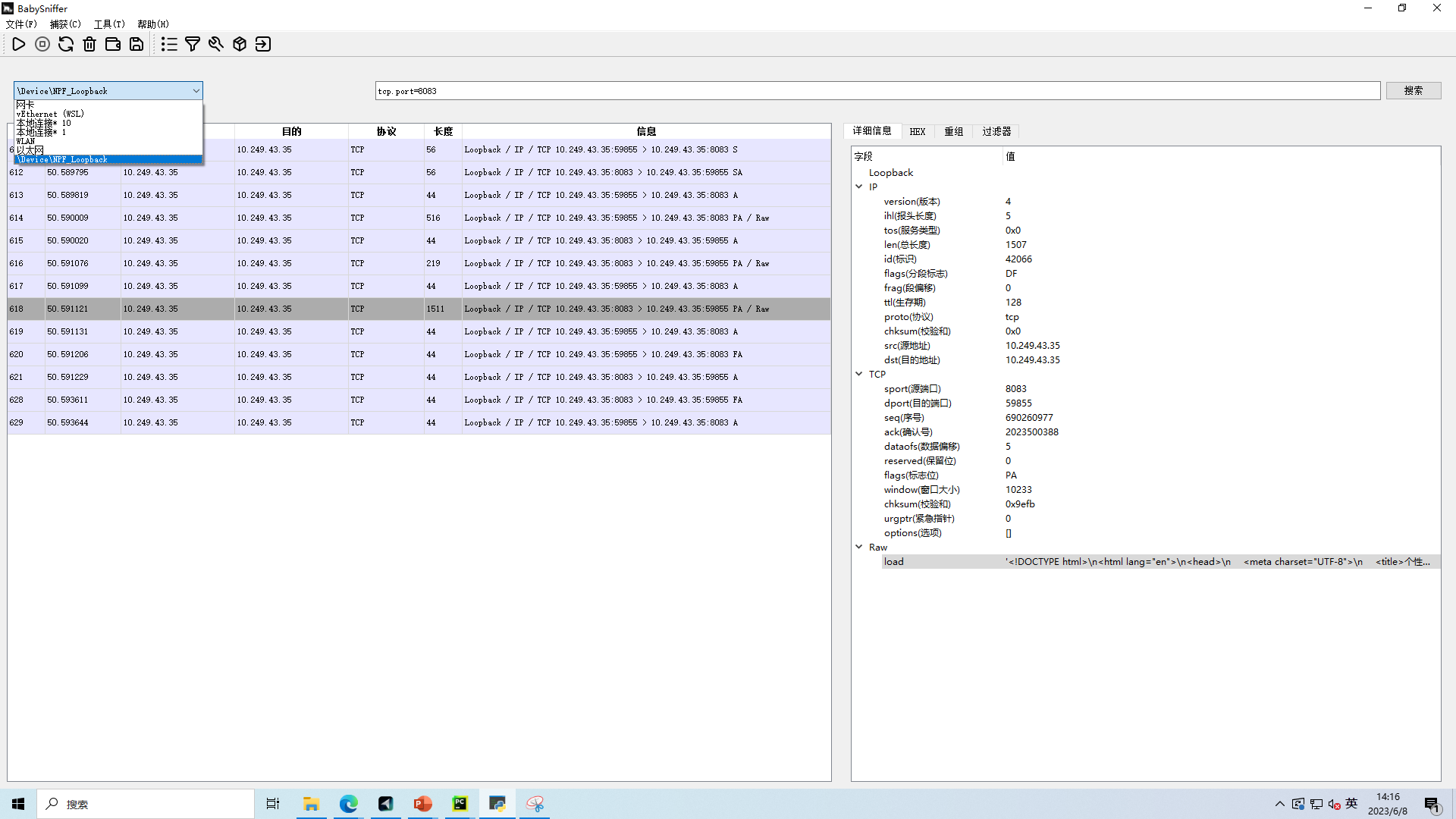
Run main.py under the C:\Users\18399\Desktop\project1\Sniffer-main\Sniffer-main\source path to achieve visualization

Run webpage.py under the C:\Users\18399\Desktop\project1\Sniffer-main\Sniffer-main\utils path to build a simple page

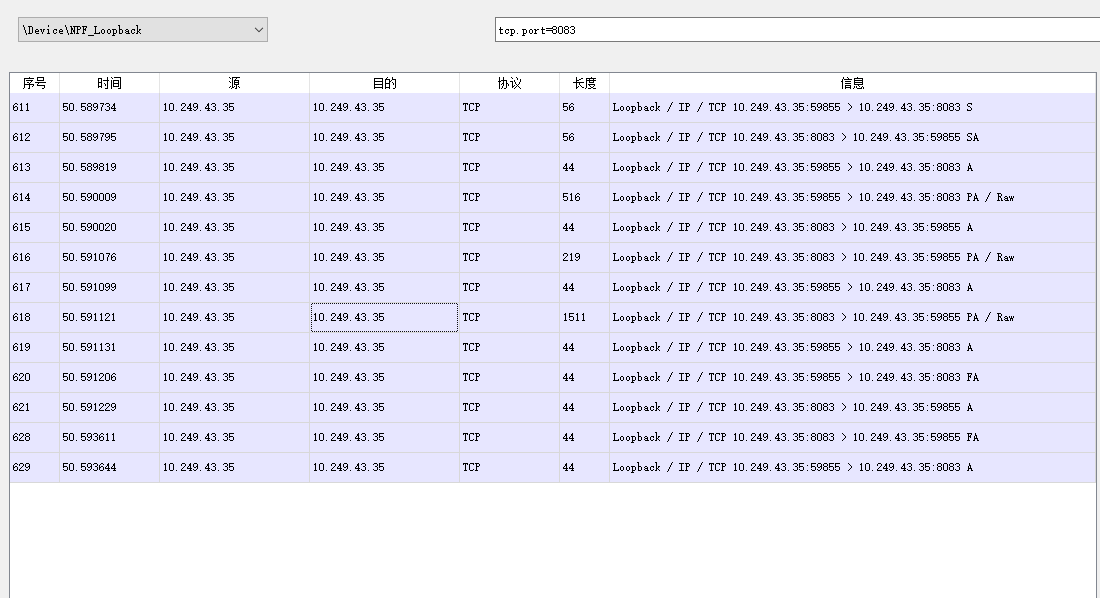
Run decode.py under the C:\Users\18399\Desktop\project1\Sniffer-main\Sniffer-main\utils path to decompress the binary file

## 2. Visual interface implementation function

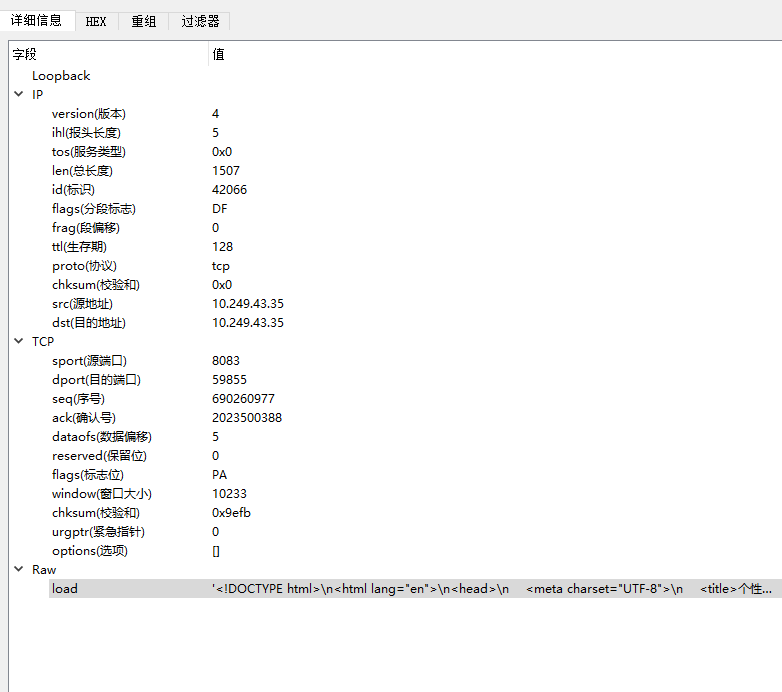
1. Network device identification



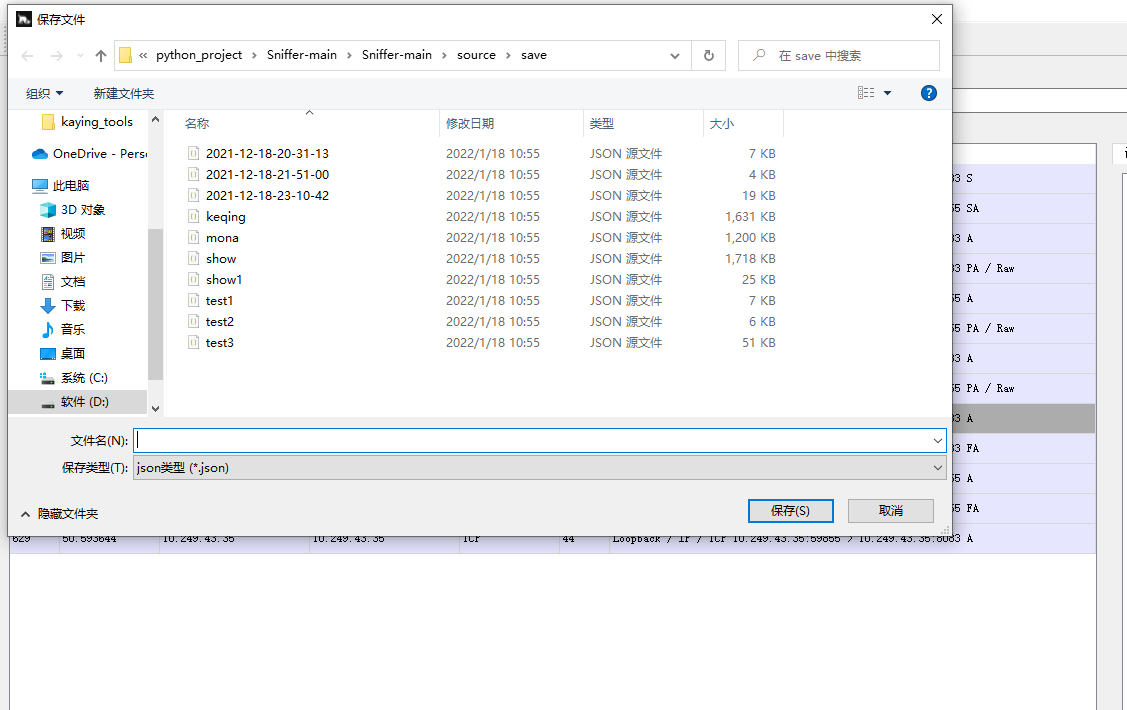
1. Packet filtering



1. Protocol header parsing



1. Save packet content



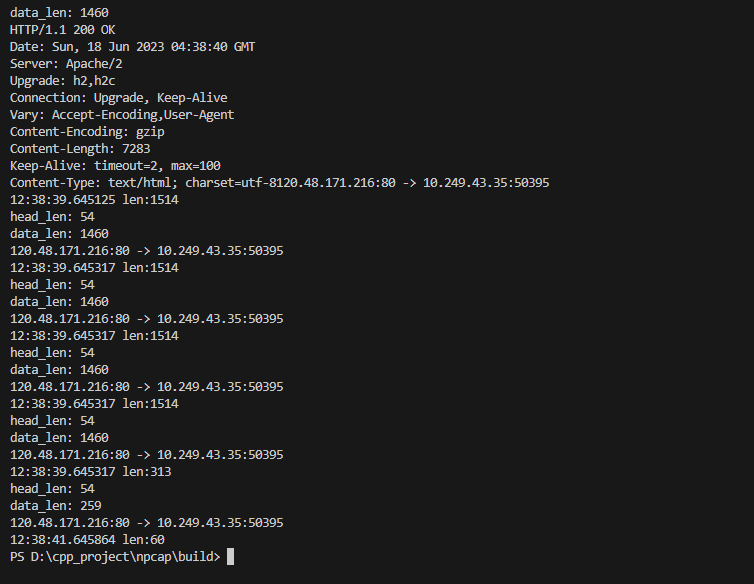
## **3. Restore web page**

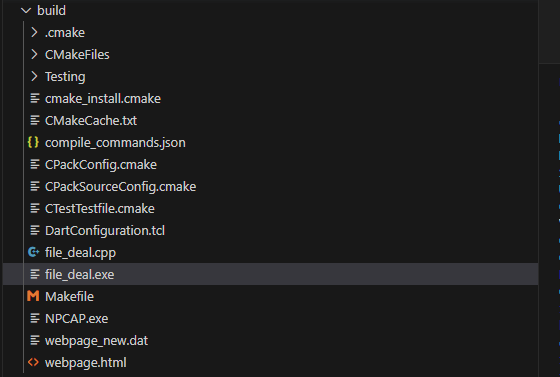
This part is written in c++

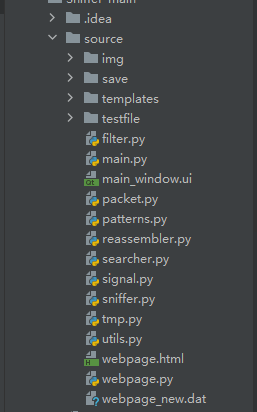
Running results

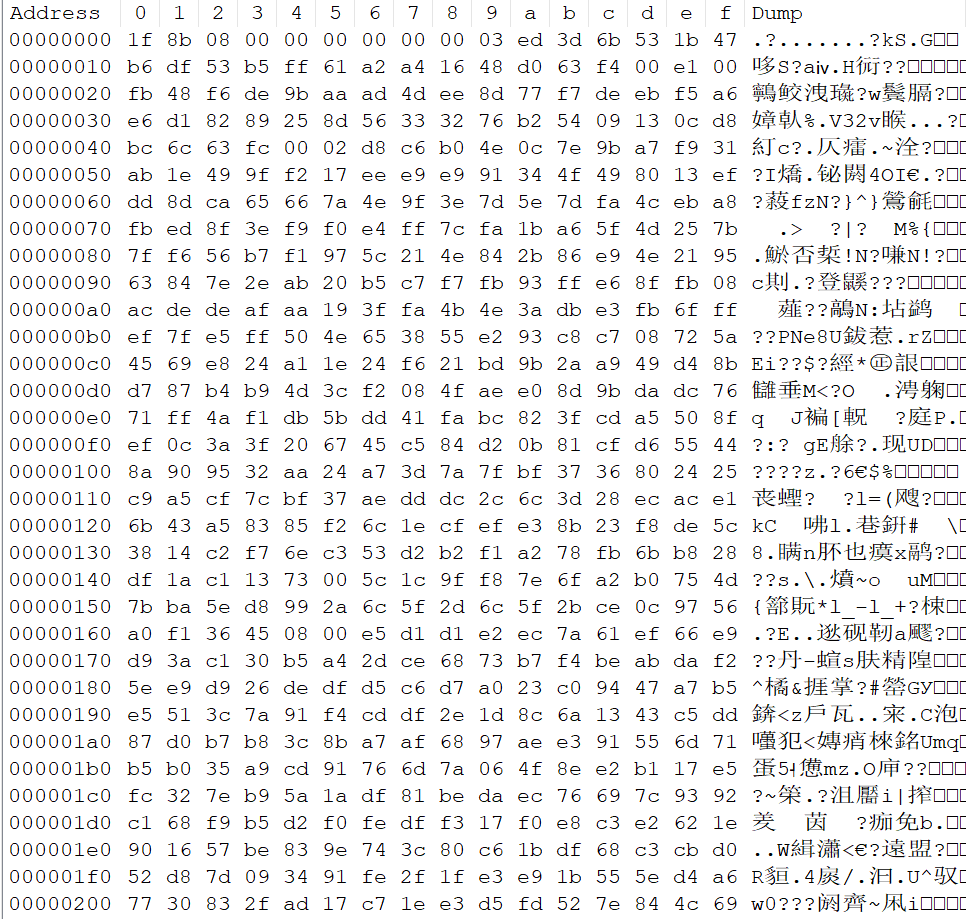
If you capture [www.weislank.com](file:///D:\WeChat\wechat_files\WeChat%20Files\wxid_hyhfyueldjno22\FileStorage\File\2023-06\www.weislank.com) NPCAP.exe will generate a binary file-------> webpage\_new.dat. Copy it to the decode.py folder, use decode.py to decompress it, and encode it to get the html web page-----------> Webpage.html

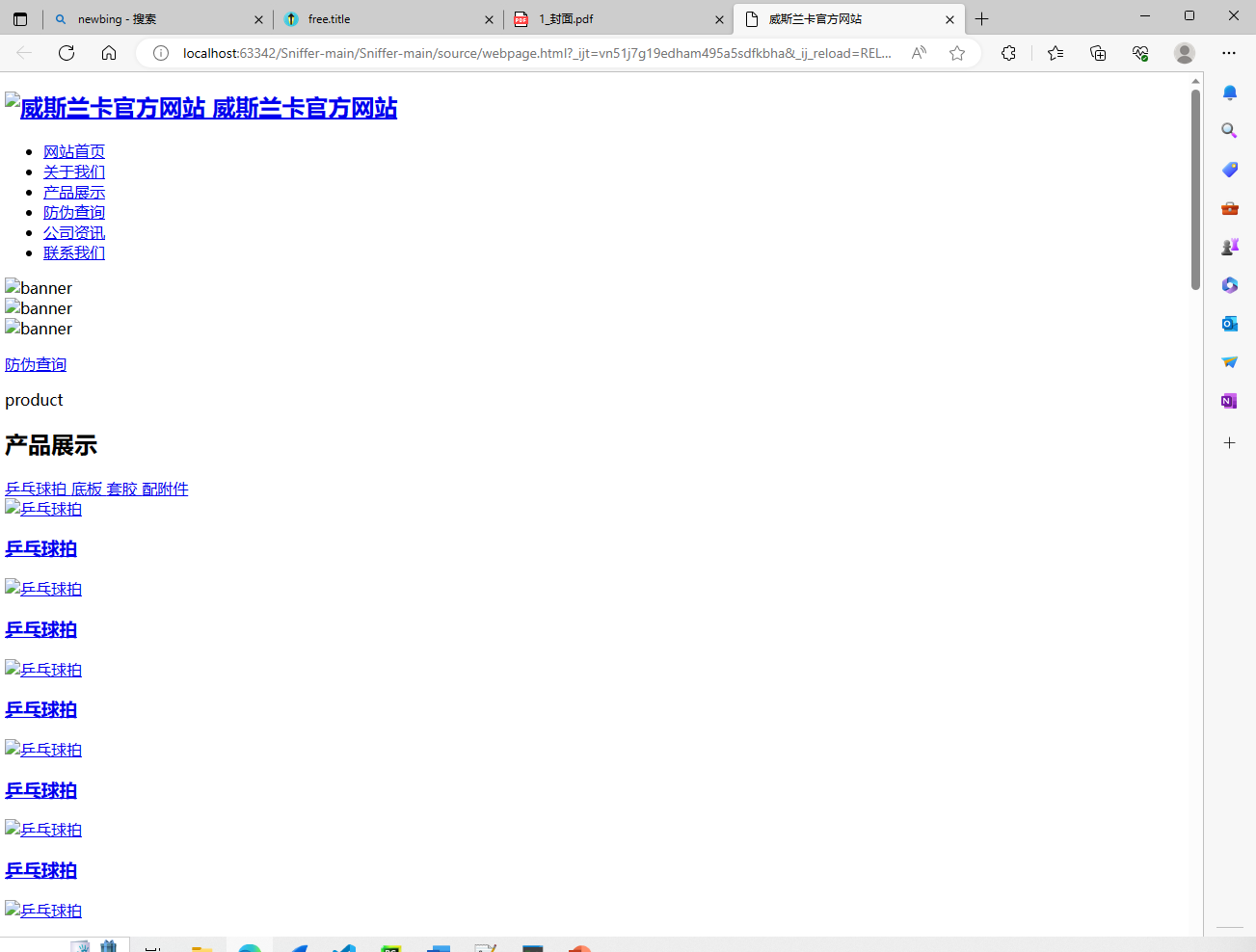
If you capture a local web page Directly generate file ----------> webpage.html under NPCAP path, open it



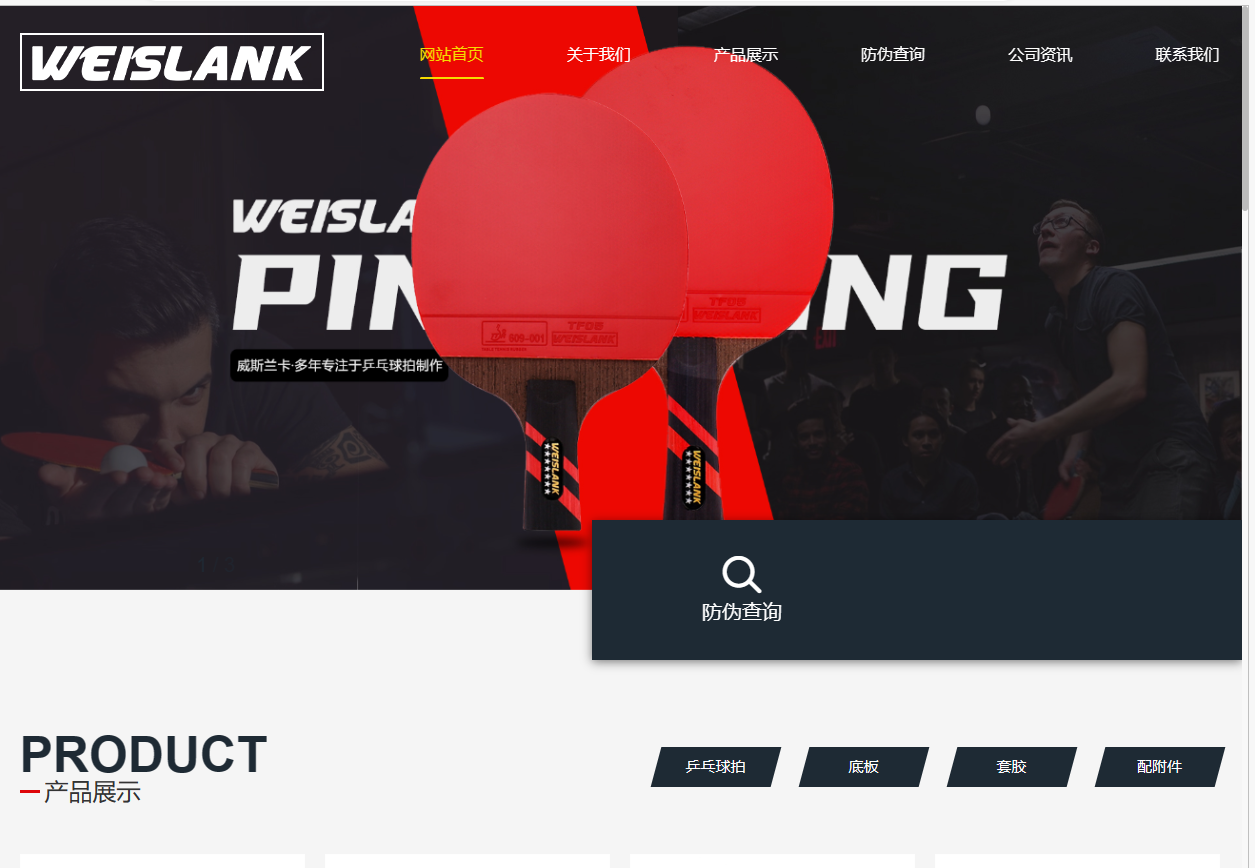




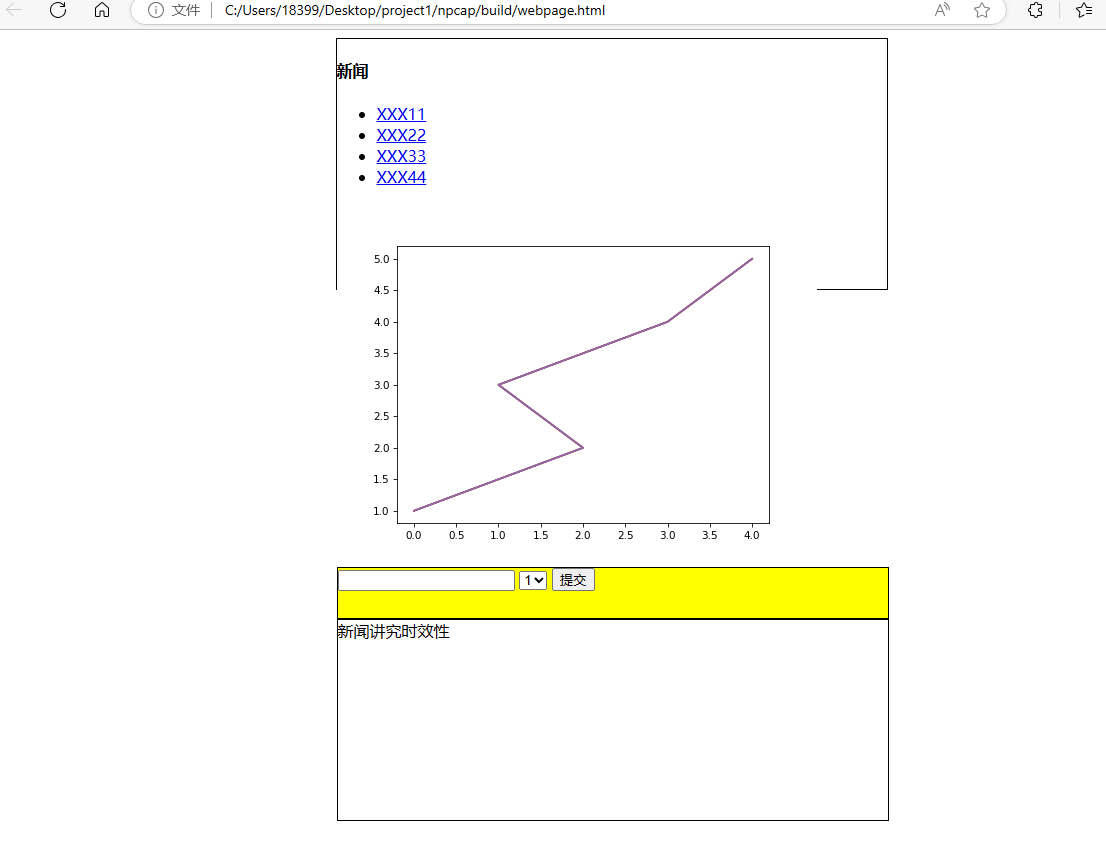




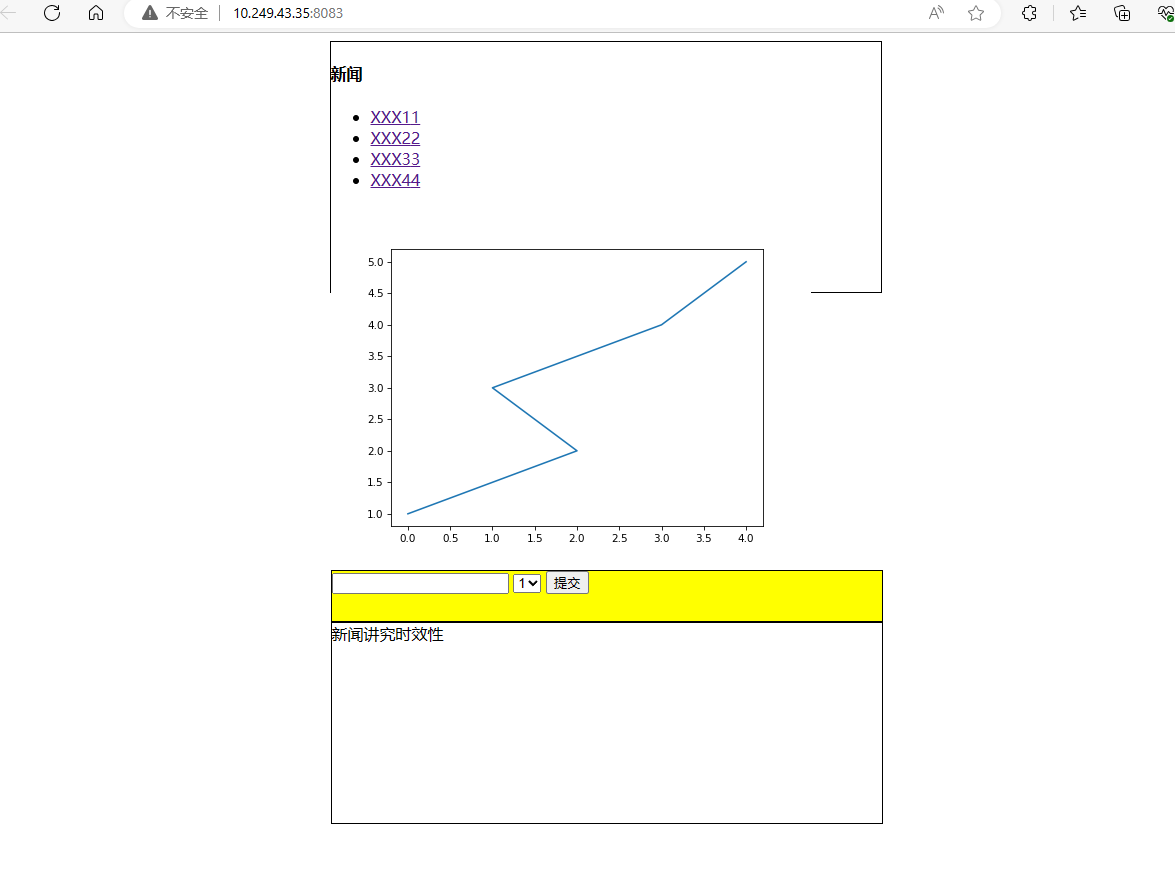
Parsing web pages



Original Page



Parsing web pages



Original Page