

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机网络基础 |
| 实验名称： | 实现一个轻量级的WEB服务器 |
| 姓 名： | 徐铭 |
| 学 院： | 计算机学院 |
| 系： |  |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3210102037 |
| 指导教师： | 陆魁军 |

2023年 12 月 22 日

**浙江大学实验报告**

实验名称：实现一个轻量级的WEB服务器 实验类型： 编程实验

同组学生： 无 实验地点： 计算机网络实验室

# 实验目的

深入掌握HTTP协议规范，学习如何编写标准的互联网应用服务器。

# 实验内容

* 服务程序能够正确解析HTTP协议，并传回所需的网页文件和图片文件
* 使用标准的浏览器，如IE、Chrome或者Safari，输入服务程序的URL后，能够正常显示服务器上的网页文件和图片
* 服务端程序界面不做要求，使用命令行或最简单的窗体即可
* 功能要求如下：

1. 服务程序运行后监听在80端口或者指定端口
2. 接受浏览器的TCP连接（支持多个浏览器同时连接）
3. 读取浏览器发送的数据，解析HTTP请求头部，找到感兴趣的部分
4. 根据HTTP头部请求的文件路径，打开并读取服务器磁盘上的文件，以HTTP响应格式传回浏览器。要求按照文本、图片文件传送不同的Content-Type，以便让浏览器能够正常显示。
5. 分别使用单个纯文本、只包含文字的HTML文件、包含文字和图片的HTML文件进行测试，浏览器均能正常显示。

* 本实验可以在前一个Socket编程实验的基础上继续，也可以使用第三方封装好的TCP类进行网络数据的收发
* 本实验要求不使用任何封装HTTP接口的类库或组件，也不使用任何服务端脚本程序如JSP、ASPX、PHP等

# 主要仪器设备

联网的PC机、Wireshark软件、Visual Studio、gcc或Java集成开发环境。

# 操作方法与实验步骤

* 阅读HTTP协议相关标准文档，详细了解HTTP协议标准的细节，有必要的话使用Wireshark抓包，研究浏览器和WEB服务器之间的交互过程
* 创建一个文档目录，与服务器程序运行路径分开
* 准备一个纯文本文件，命名为test.txt，存放在txt子目录下面
* 准备好一个图片文件，命名为logo.jpg，放在img子目录下面
* 写一个HTML文件，命名为test.html，放在html子目录下面，主要内容为：

<html>

<head><title>Test</title></head>

<body>

<h1>This is a test</h1>

<img src="img/logo.jpg">

<form action="dopost" method="POST">

Login:<input name="login">

Pass:<input name="pass">

<input type="submit" value="login">

</form>

</body>

</html>

* 将test.html复制为noimg.html，并删除其中包含img的这一行。
* 服务端编写步骤（**需要采用多线程模式**）

1. 运行初始化，打开Socket，监听在指定端口（**请使用学号的后4位作为服务器的监听端口**）
2. 主线程是一个循环，主要做的工作是等待客户端连接，如果有客户端连接成功，为该客户端创建处理子线程。该子线程的主要处理步骤是：
3. 不断读取客户端发送过来的字节，并检查其中是否连续出现了2个回车换行符，如果未出现，继续接收；如果出现，按照HTTP格式解析第1行，分离出方法、文件和路径名，其他头部字段根据需要读取。

* **如果解析出来的方法是GET**

1. 根据解析出来的文件和路径名，读取响应的磁盘文件（该路径和服务器程序可能不在同一个目录下，需要转换成绝对路径）。如果文件不存在，第3步的响应消息的状态设置为404，并且跳过第5步。
2. 准备好一个足够大的缓冲区，按照HTTP响应消息的格式先填入第1行（状态码=200），加上回车换行符。然后模仿Wireshark抓取的HTTP消息，填入必要的几行头部（需要哪些头部，请试验），其中不能缺少的2个头部是Content-Type和Content-Length。Content-Type的值要和文件类型相匹配（请通过抓包确定应该填什么），Content-Length的值填写文件的字节大小。
3. 在头部行填完后，再填入2个回车换行
4. 将文件内容按顺序填入到缓冲区后面部分。

* **如果解析出来的方法是POST**

1. 检查解析出来的文件和路径名，如果不是dopost，则设置响应消息的状态为404，然后跳到第9步。如果是dopost，则设置响应消息的状态为200，并继续下一步。
2. 读取2个回车换行后面的体部内容（长度根据头部的Content-Length字段的指示），并提取出登录名（login）和密码（pass）的值。如果登录名是你的学号，密码是学号的后4位，则将响应消息设置为登录成功，否则将响应消息设置为登录失败。
3. 将响应消息封装成html格式，如

<html><body>响应消息内容</body></html>

1. 准备好一个足够大的缓冲区，按照HTTP响应消息的格式先填入第1行（根据前面的情况设置好状态码），加上回车换行符。然后填入必要的几行头部，其中不能缺少的2个头部是Content-Type和Content-Length。Content-Type的值设置为text/html，如果状态码=200，则Content-Length的值填写响应消息的字节大小，并将响应消息填入缓冲区的后面部分，否则填写为0。
2. 最后一次性将缓冲区内的字节发送给客户端。
3. 发送完毕后，关闭socket，退出子线程。
4. 主线程还负责检测退出指令（如用户按退出键或者收到退出信号），检测到后即通知并等待各子线程退出。最后关闭Socket，主程序退出。

* 编程结束后，将服务器部署在一台机器上（本机也可以）。在服务器上分别放置纯文本文件（.txt）、只包含文字的测试HTML文件（将测试HTML文件中的包含img那一行去掉）、包含文字和图片的测试HTML文件（以及图片文件）各一个。
* 确定好各个文件的URL地址，然后使用浏览器访问这些URL地址，如http://x.x.x.x:port/dir/a.html，其中port是服务器的监听端口，dir是提供给外部访问的路径，请设置为与文件实际存放路径不同，通过服务器内部映射转换。
* 检查浏览器是否正常显示页面，如果有问题，查找原因，并修改，直至满足要求
* 使用多个浏览器同时访问这些URL地址，检查并发性

# 实验数据记录和处理

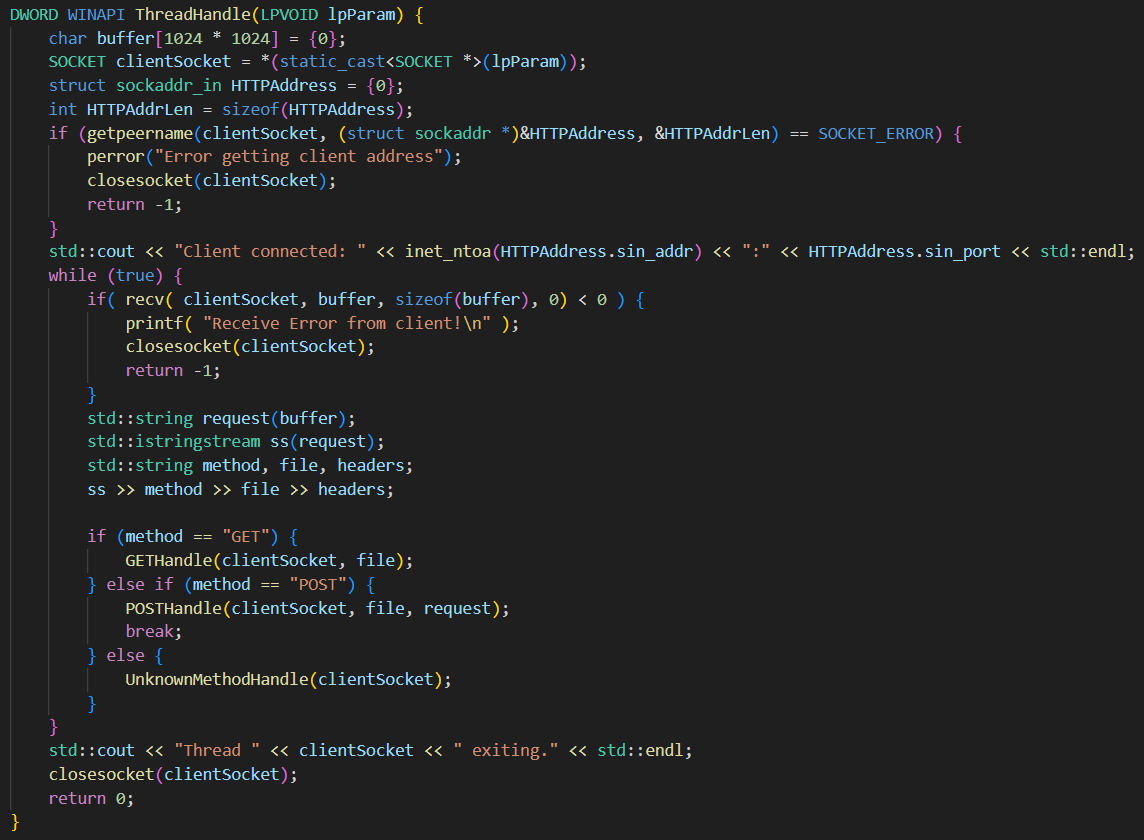
**请将以下内容和本实验报告一起打包成一个压缩文件上传：**

* **源代码：需要说明编译环境和编译方法，如果不能编译成功，将影响评分**
* **可执行文件：可运行的.exe文件或Linux可执行文件**
* 服务器的主线程循环关键代码截图（解释总体处理逻辑，省略细节部分）



上面的代码包含一个无限循环，负责处理客户端连接。在循环内部，我使用了accept函数接受客户端连接，并在成功时创建一个新的套接字(clientSocket)以处理连接的客户端的通信。如果accept返回INVALID\_SOCKET，则打印错误消息并继续下一次迭代。对于每个成功接受的连接，我会使用CreateThread函数创建一个新线程，调用ThreadHandle函数处理与新的客户端的通信。最终，在服务器循环退出时（当exit\_requested标志为true时），我会关闭主服务器套接字，并使用WSACleanup()清理Winsock API。

* 服务器的客户端处理子线程关键代码截图（解释总体处理逻辑，省略细节部分）



首先我创建了一个用于存储接收数据的缓冲区buffer；然后，通过将参数lpParam强制类型转换为SOCKET类型，获取客户端套接字 clientSocket；接着，我使用getpeername函数获取客户端的地址信息，并在控制台输出连接的客户端的IP地址和端口号。

接下来，我们进入一个无限循环来处理客户端的具体请求，不断接收来自客户端的数据。如果接收过程中出现错误，会打印错误消息，并关闭客户端套接字，然后退出线程。接收到数据后，将其解析，使用字符串流的方式从接收到的request这种获取HTTP请求的方法、文件和头部信息。

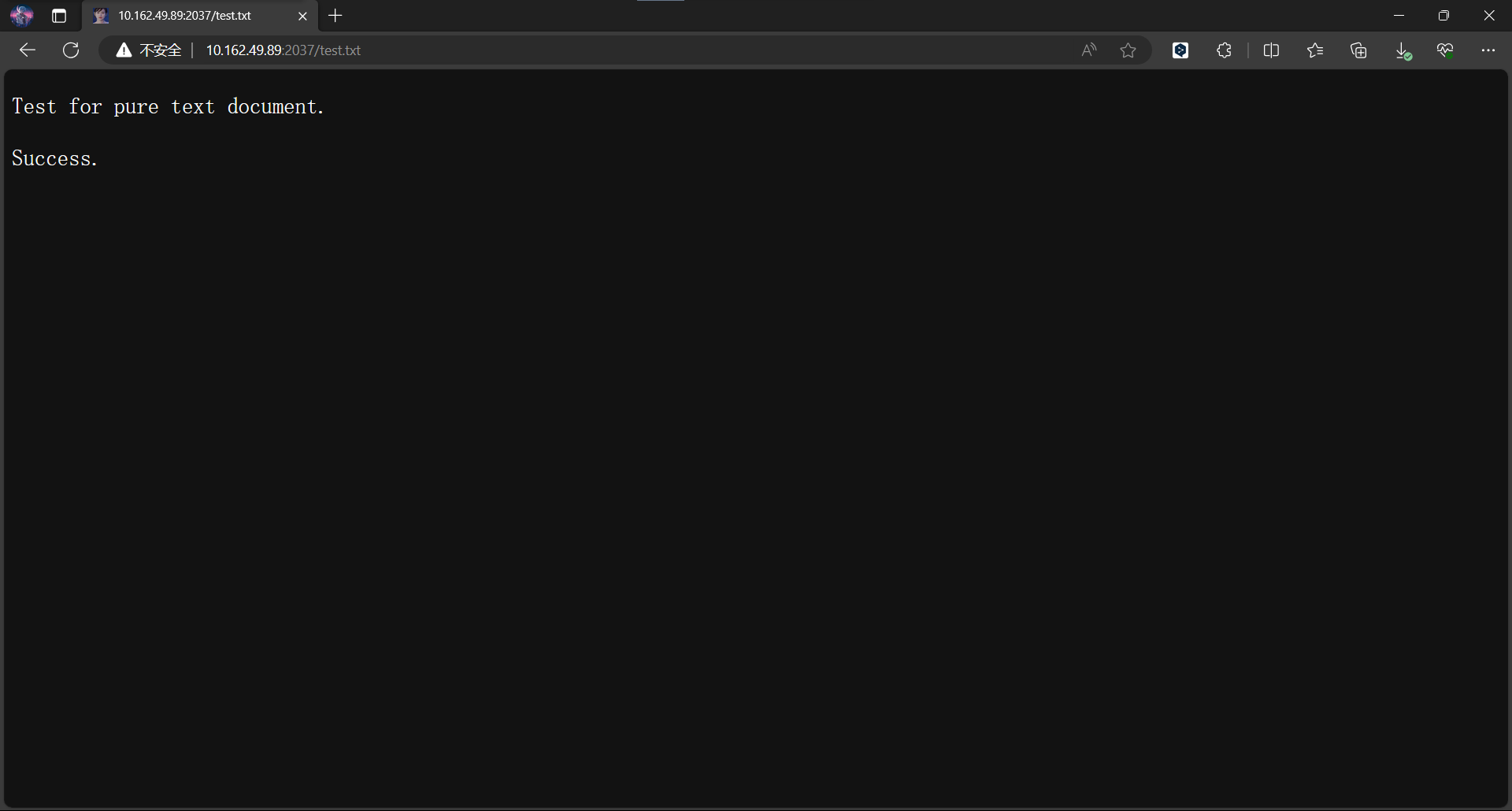
根据解析得到的HTTP方法，分别调用相应的处理函数：GETHandle用于处理GET请求，POSTHandle用于处理POST请求，而对于未知的HTTP方法，调用UnknownMethodHandle进行处理。

如果我们处理的是POST请求，那处理完后，线程打印一条消息表示正在退出，并关闭客户端套接字，最后返回0以结束线程；其他请求则继续接受这个客户端的数据，并判断新的请求是什么类型，即进入下一个循环

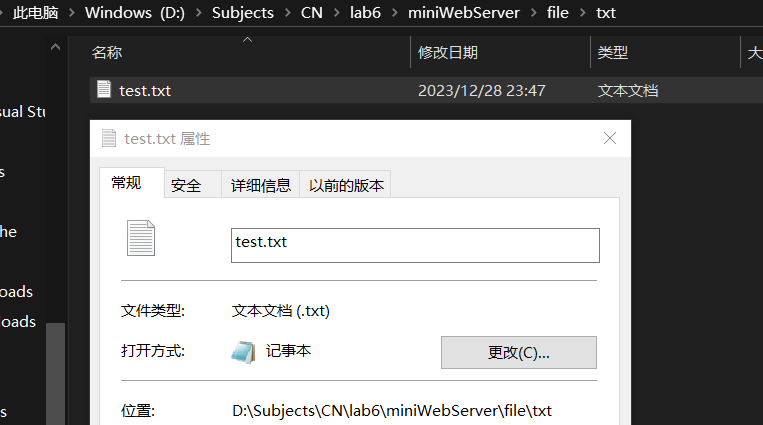
* 服务器运行后，用netstat –an显示服务器的监听端口



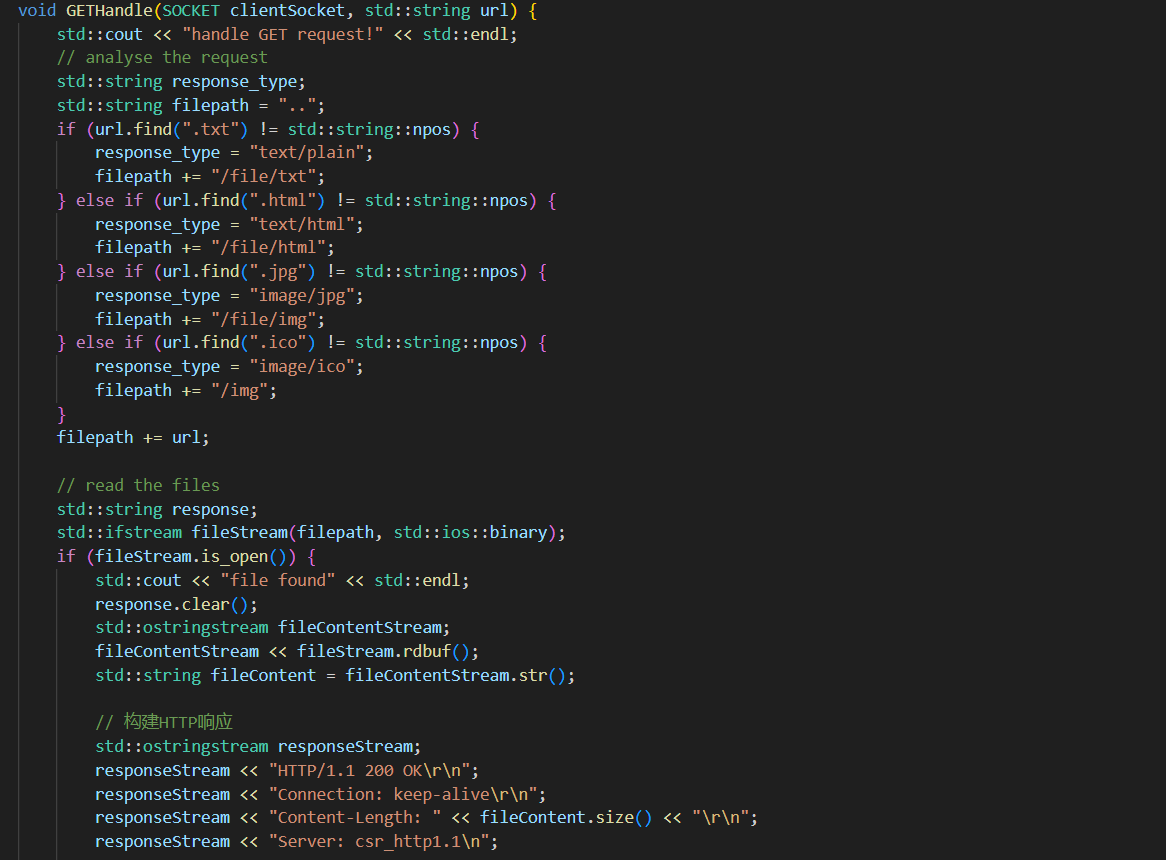
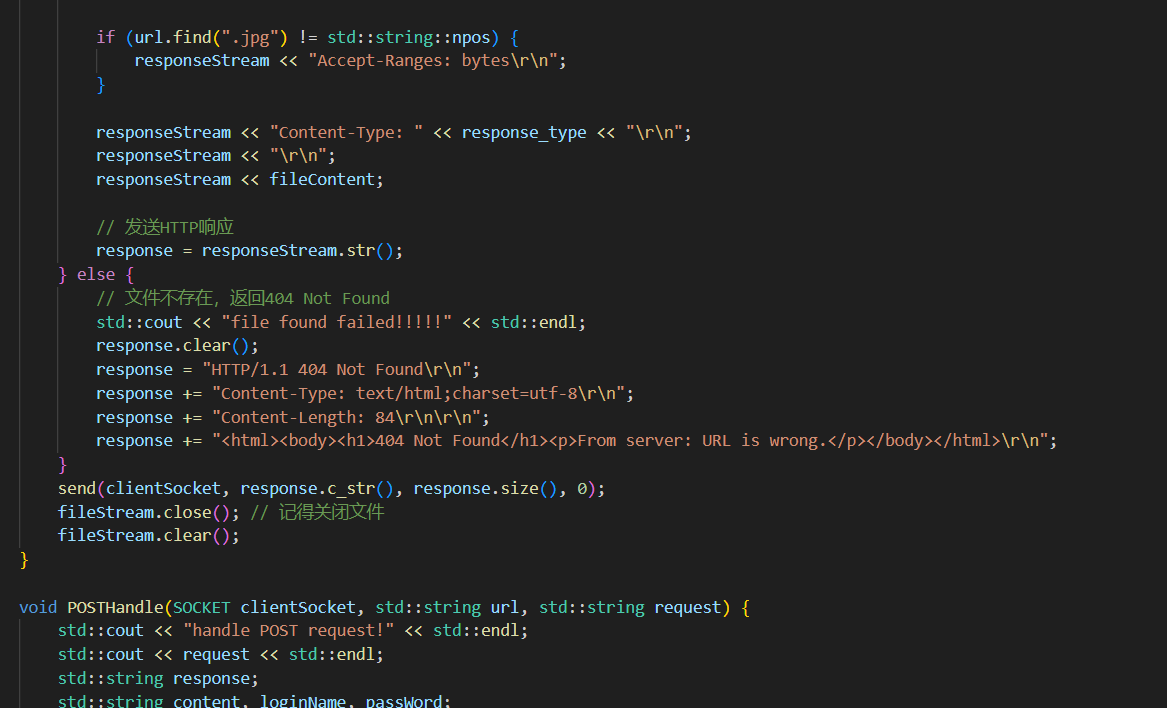
* 浏览器访问纯文本文件（.txt）时，浏览器的URL地址和显示内容截图。



服务器上文件实际存放的路径：

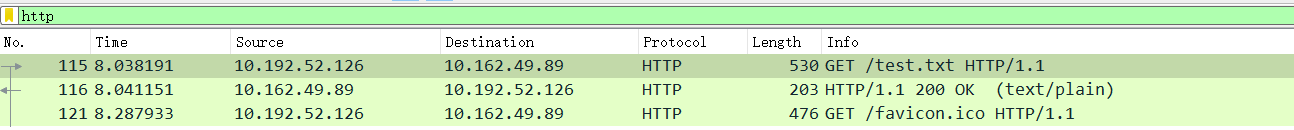


服务器的相关代码片段：

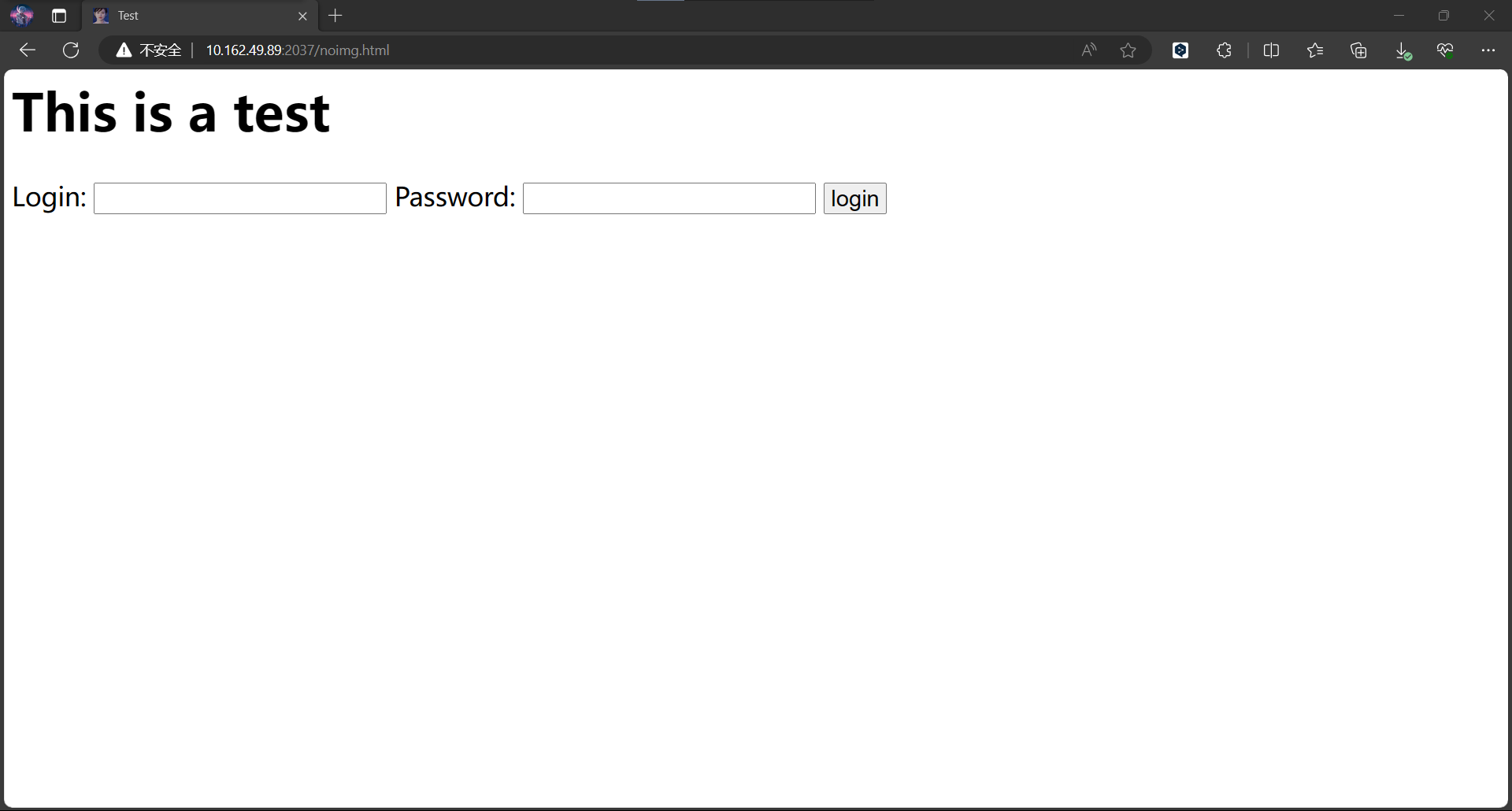


对获取到的GET请求中的请求行进行分解，获得URL中请求的文件内容，据此再取服务器本地磁盘文件中寻找对应的文件，找到后组装响应消息；

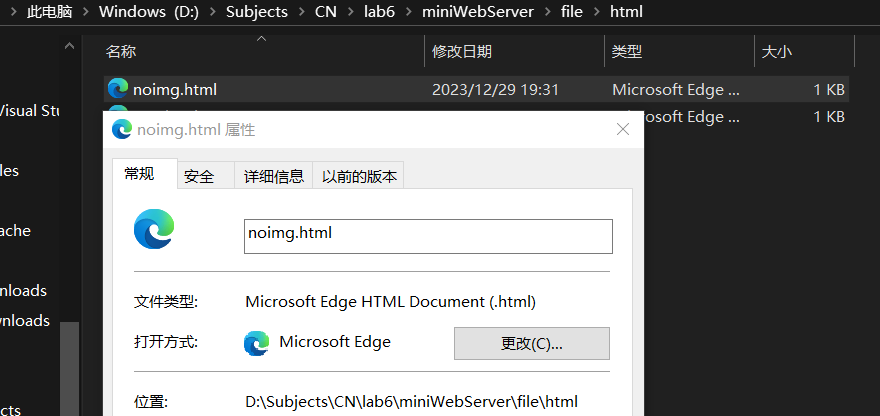
Wireshark抓取的数据包截图（通过跟踪TCP流，只截取HTTP协议部分）：



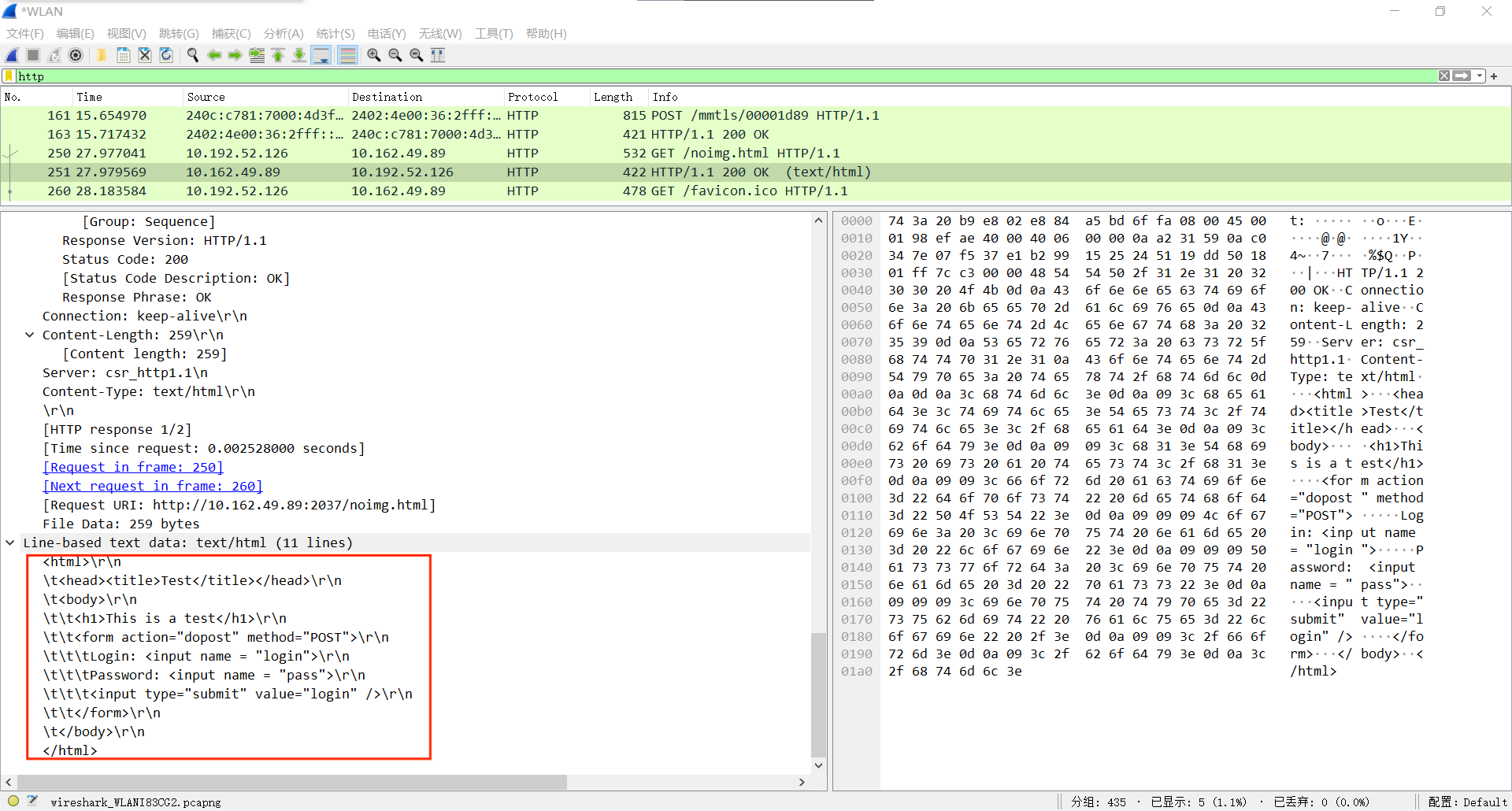
* 浏览器访问只包含文本的HTML文件时，浏览器的URL地址和显示内容截图。



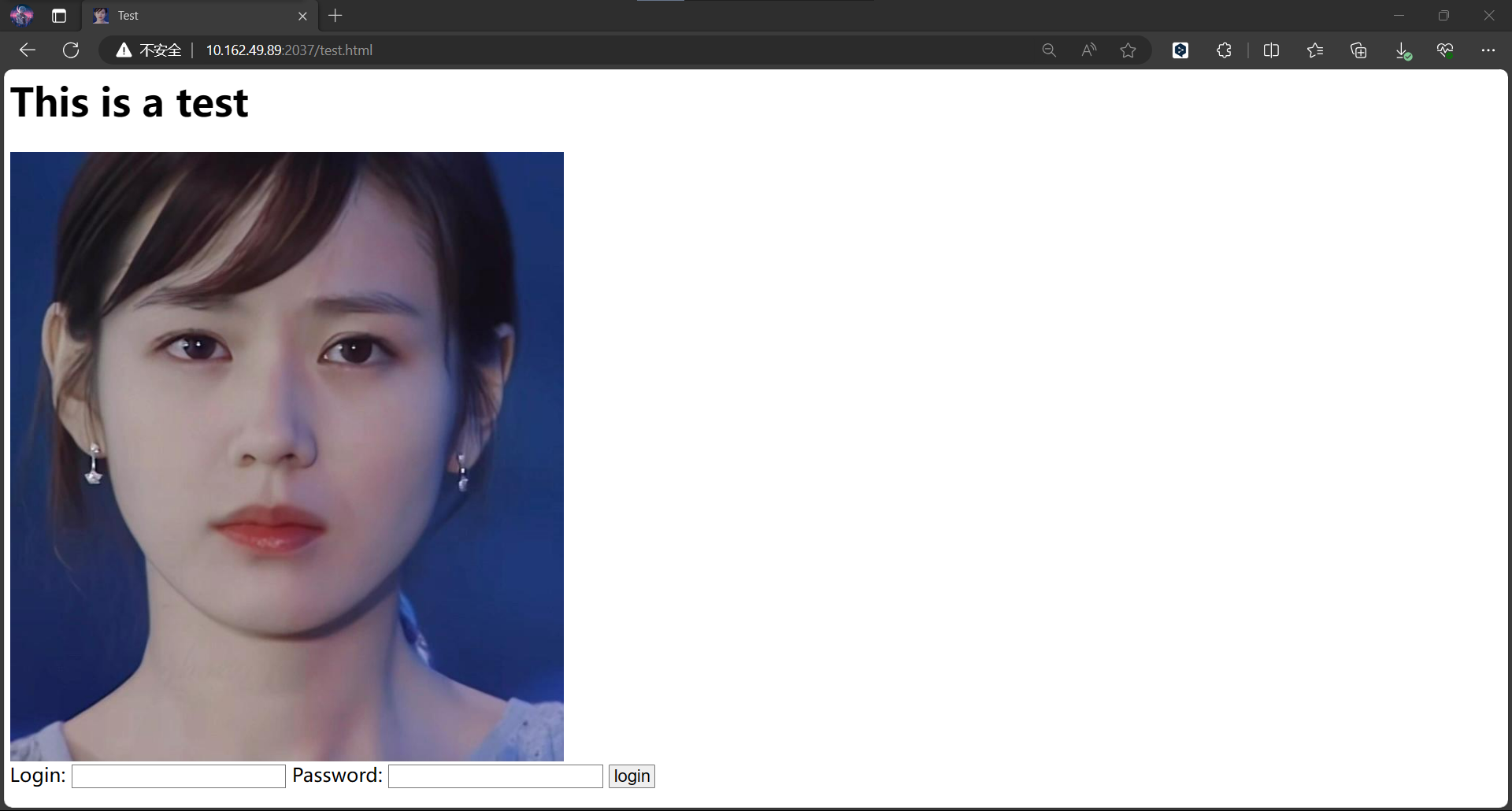
服务器文件实际存放的路径：



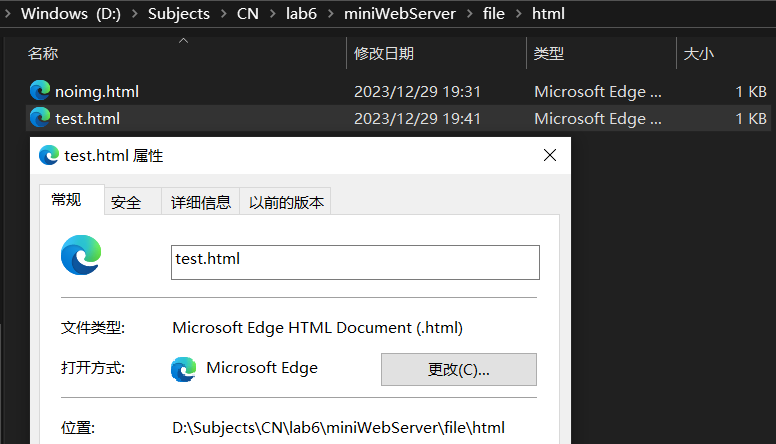
Wireshark抓取的数据包截图（只截取HTTP协议部分，包括HTML内容）：



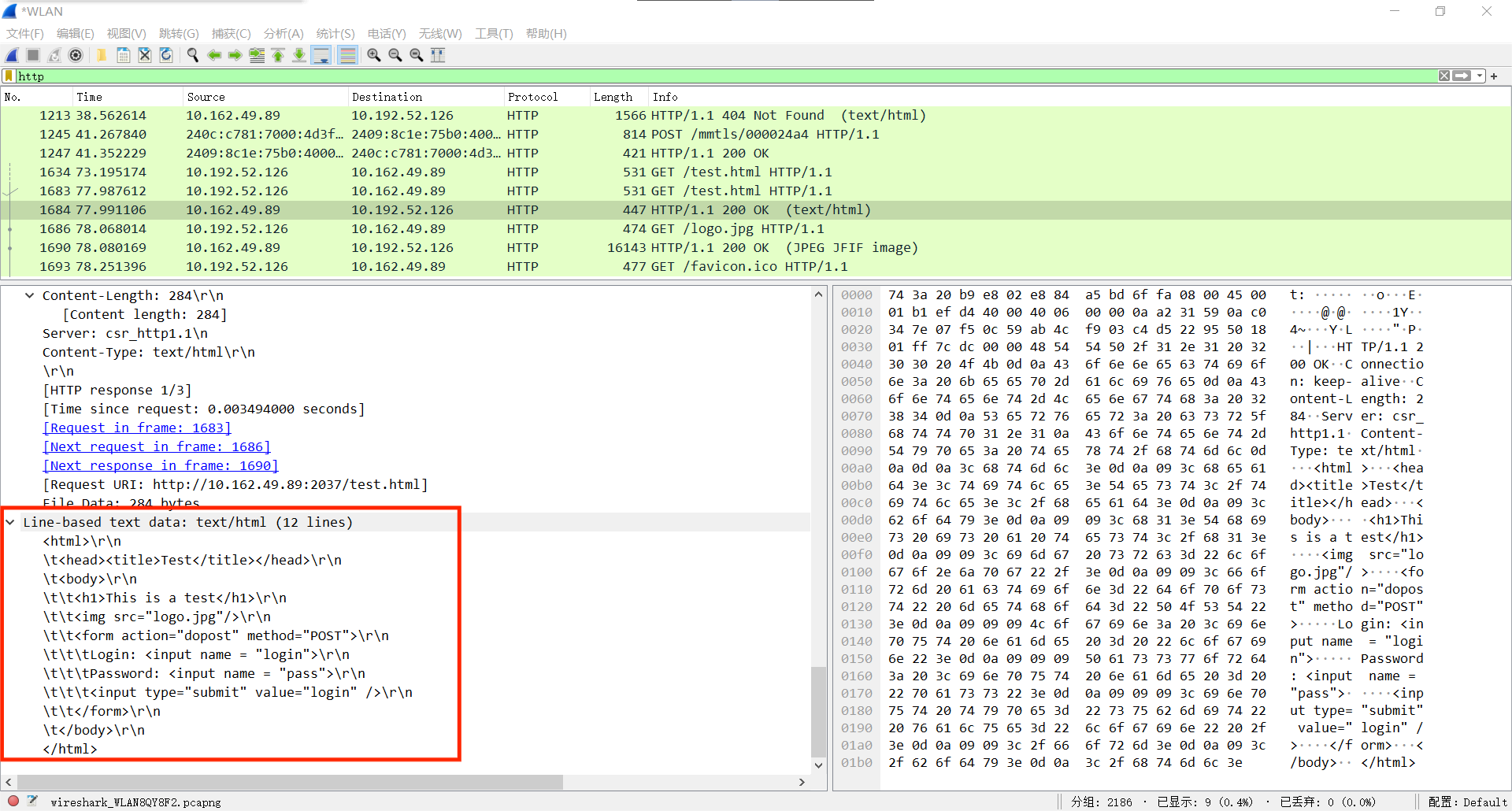
* 浏览器访问包含文本、图片的HTML文件时，浏览器的URL地址和显示内容截图。

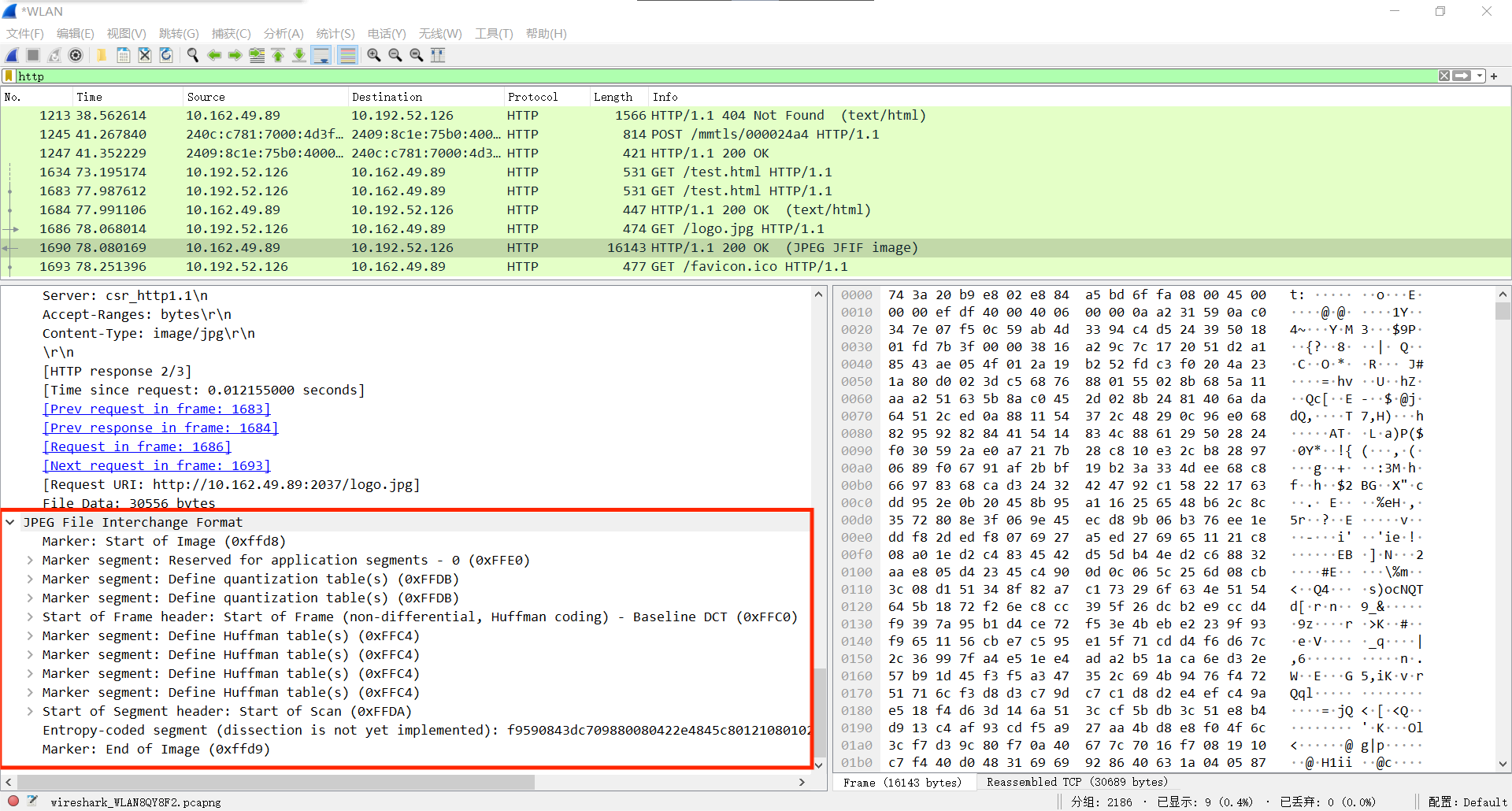


服务器上文件实际存放的路径：

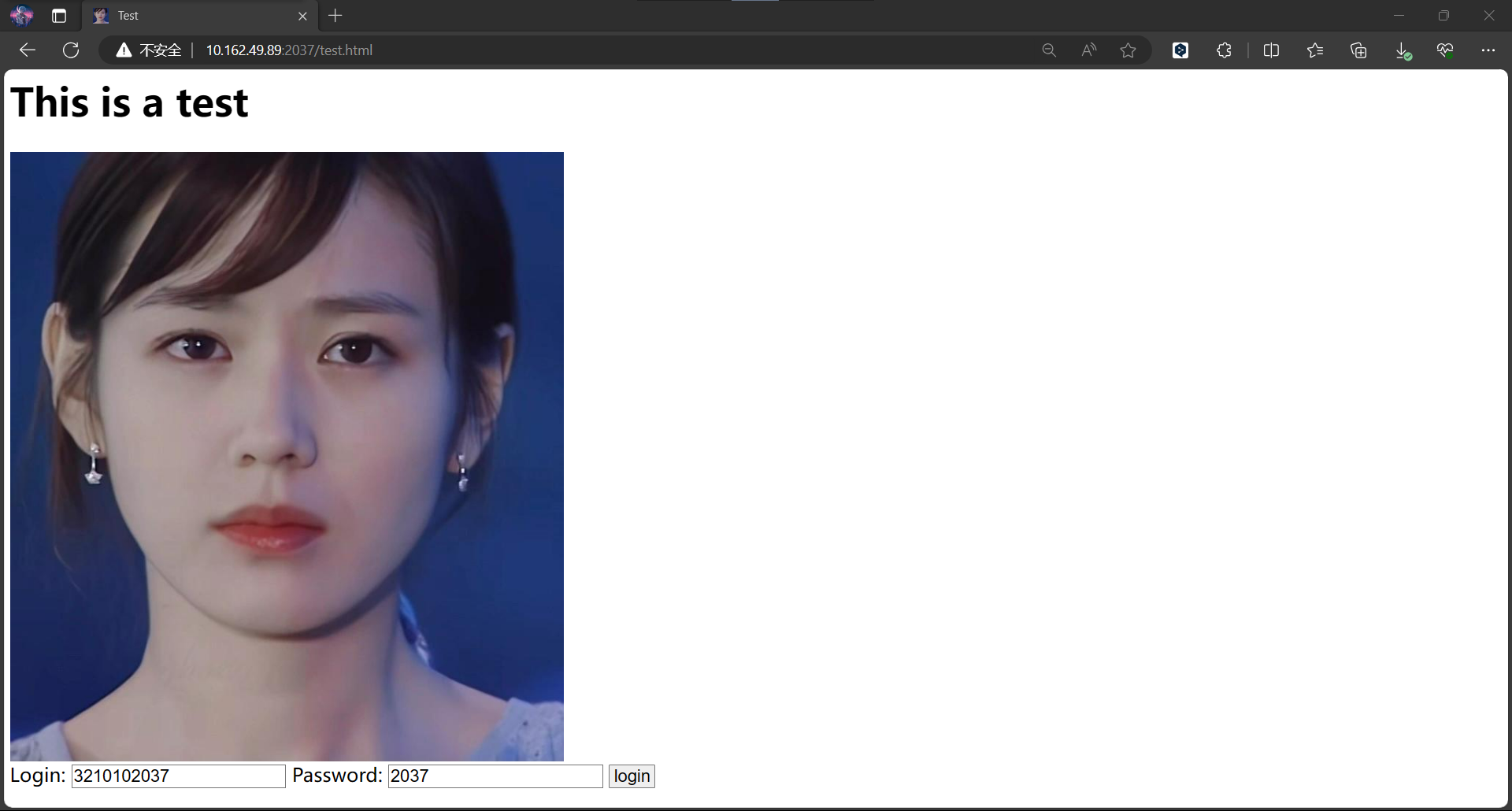


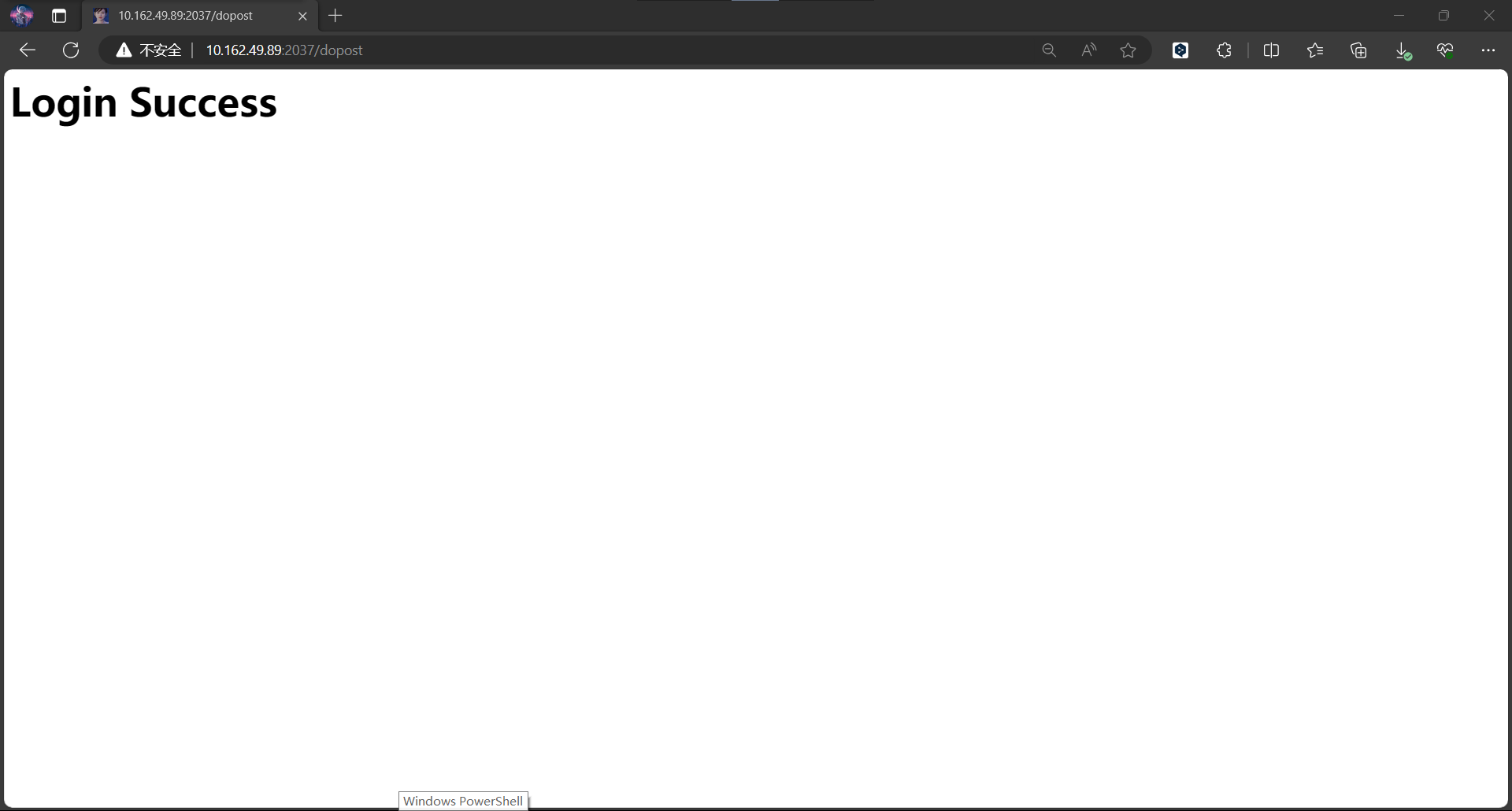
Wireshark抓取的数据包截图（只截取HTTP协议部分，包括HTML、图片文件的部分内容）：





* 浏览器输入正确的登录名或密码，点击登录按钮（login）后的显示截图。





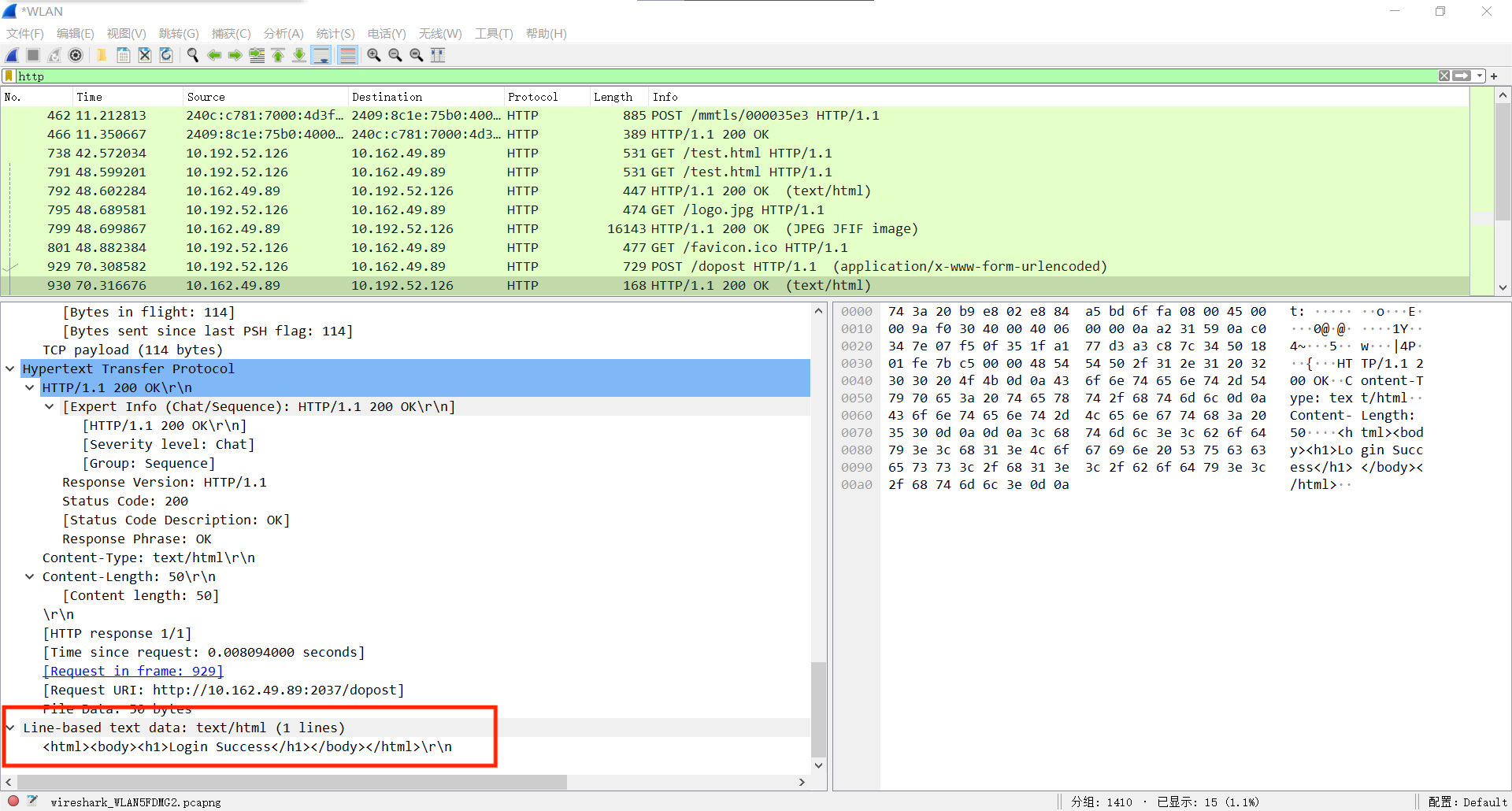
服务器相关处理代码片段：



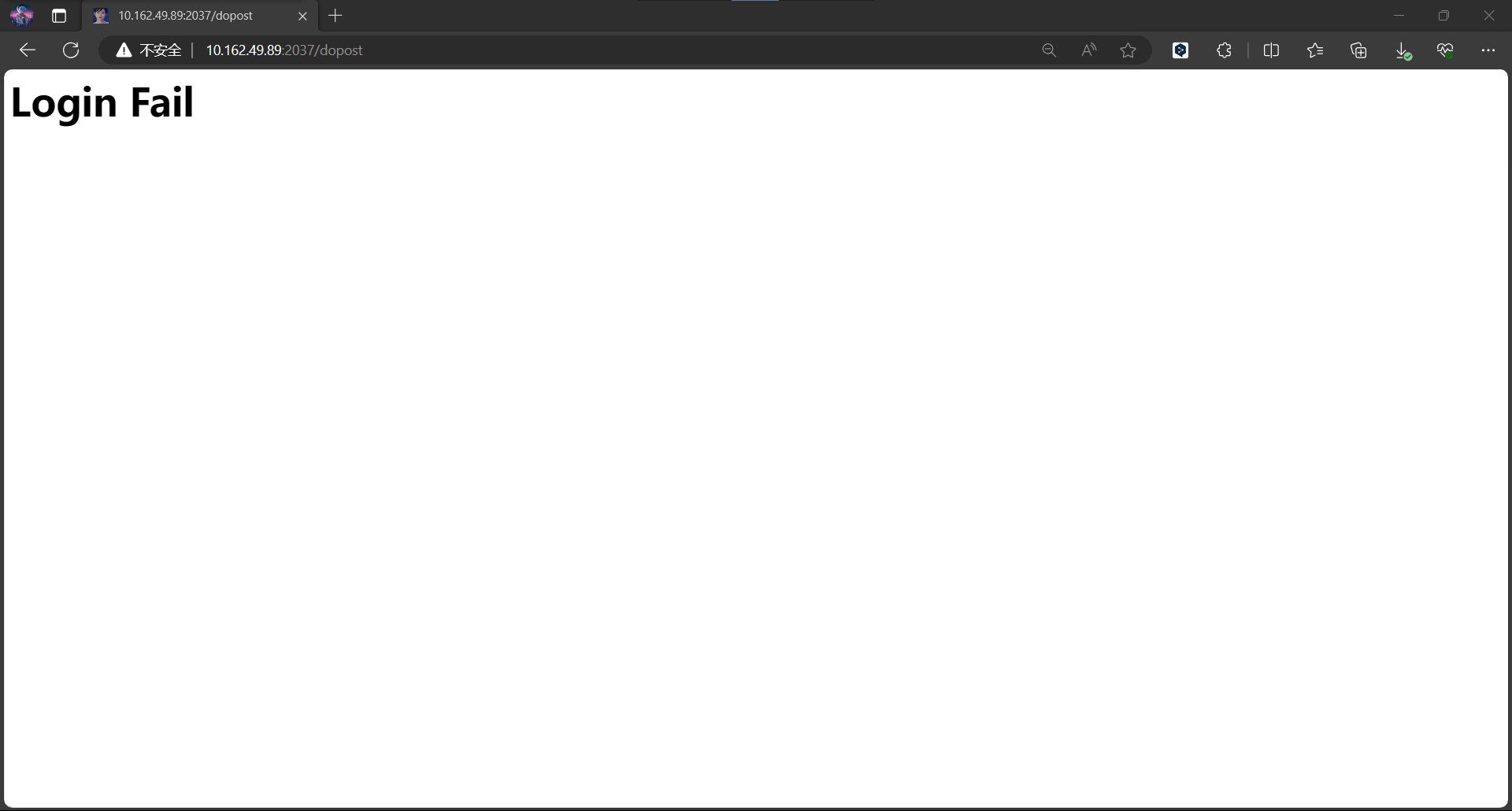
这里我们需要处理POST请求，首先根据请求头中的URL判断这是否是“dopost”请求，如果是继续进行分解，将request中的Content部分找到，并根据submit的内容格式分解出相应的登录名和密码，比较后组装相应的响应消息；

否则，组装404的响应消息。

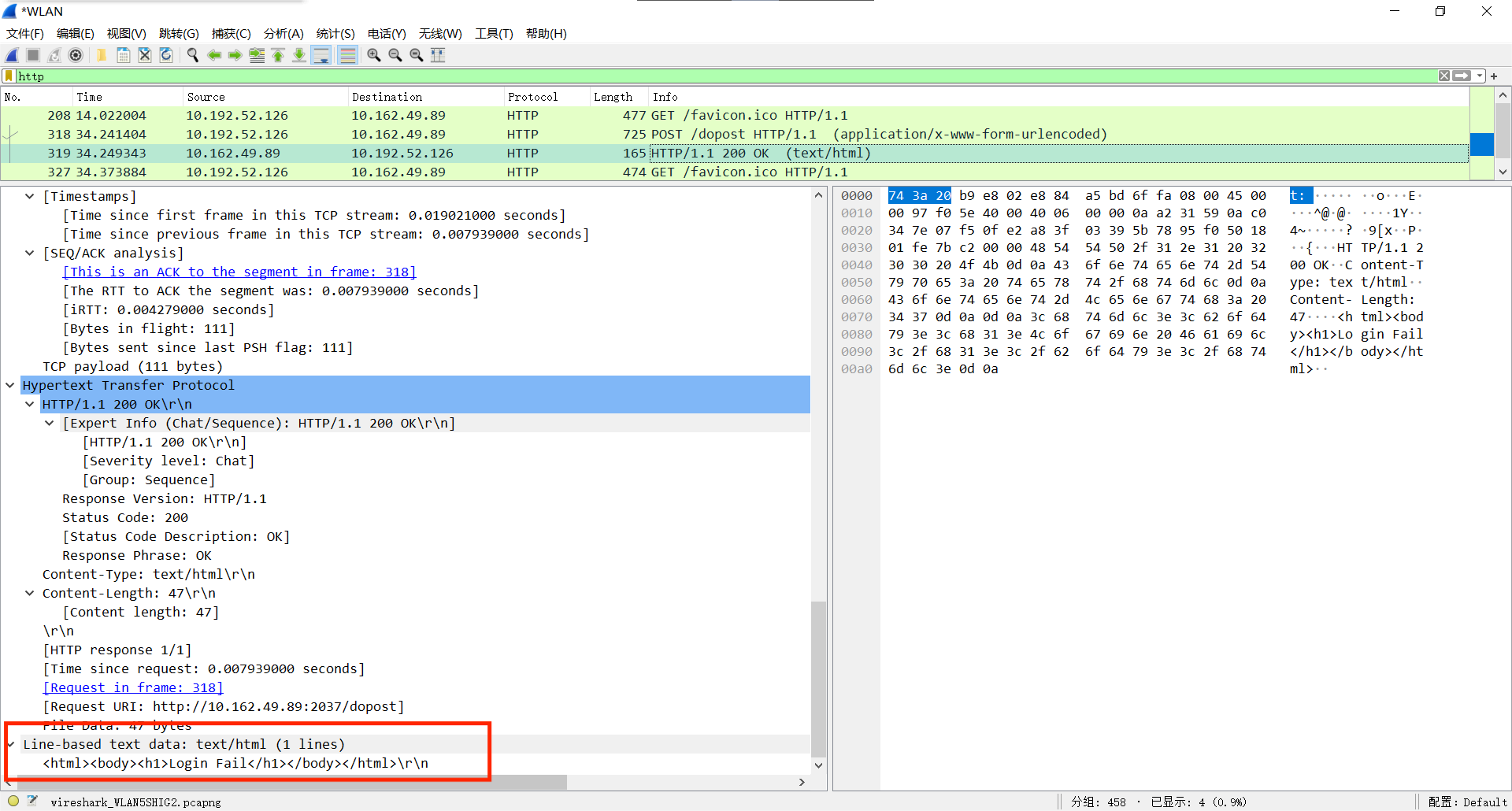
Wireshark抓取的数据包截图（HTTP协议部分）



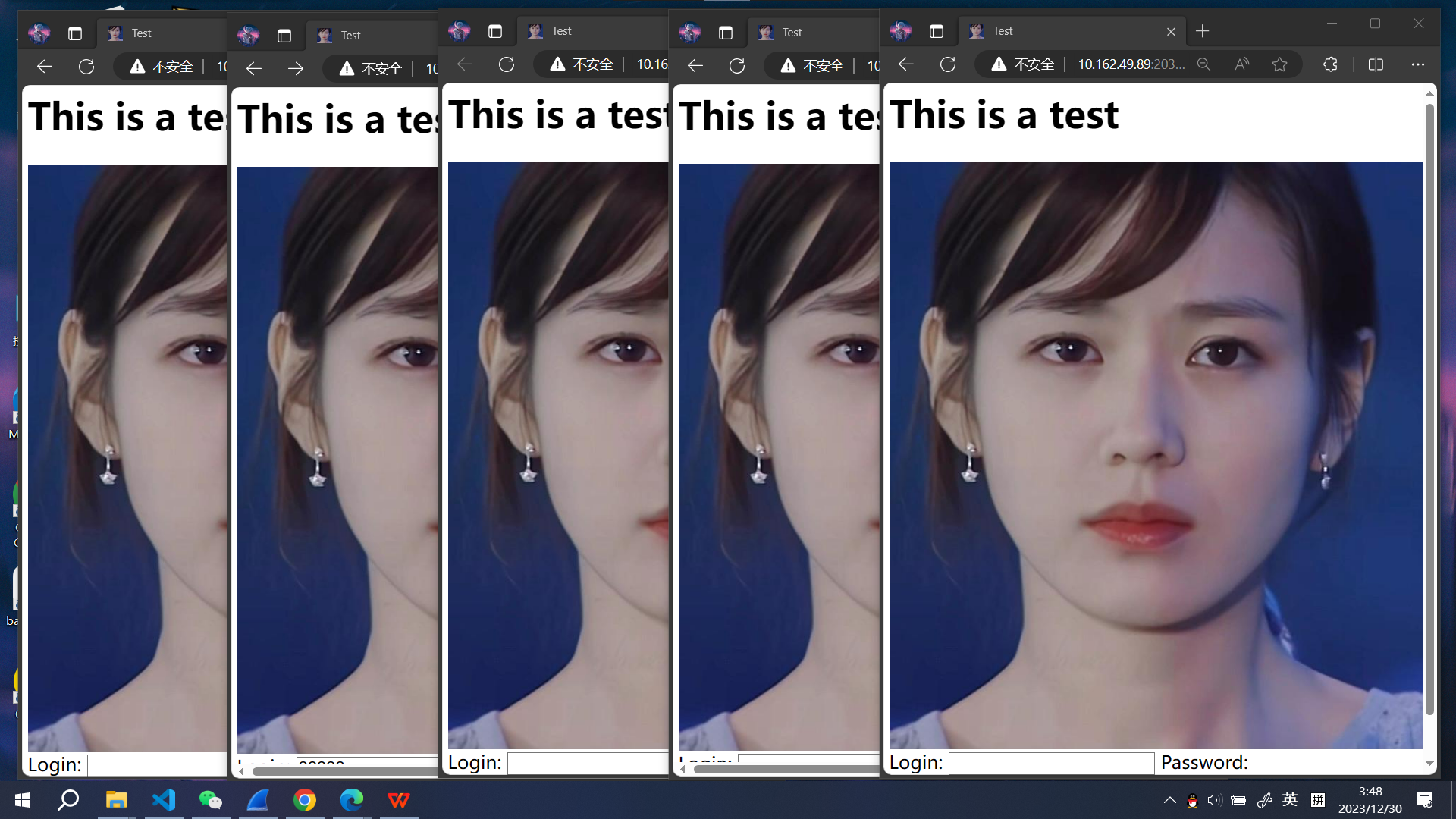
* 浏览器输入错误的登录名或密码，点击登录按钮（login）后的显示截图。



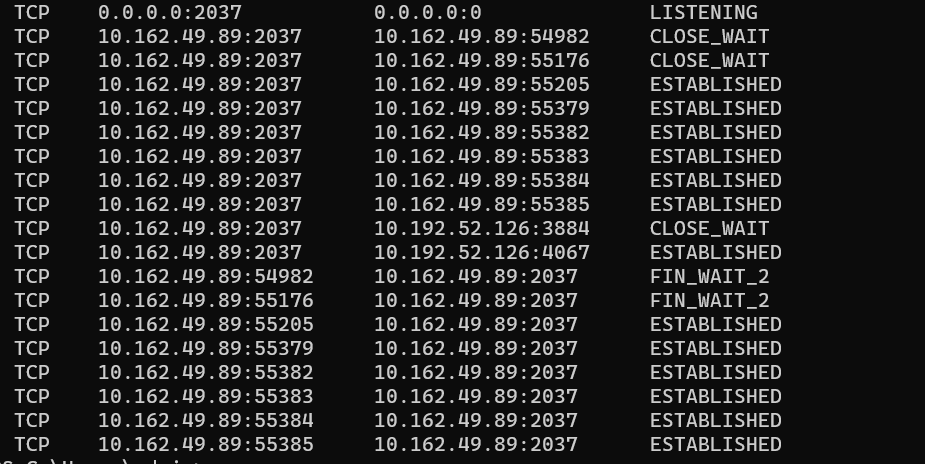
* Wireshark抓取的数据包截图（HTTP协议部分）



* 多个浏览器同时访问包含图片的HTML文件时，浏览器的显示内容截图（将浏览器窗口缩小并列）



* 多个浏览器同时访问包含图片的HTML文件时，使用netstat –an显示服务器的TCP连接（截取与服务器监听端口相关的）



# 实验结果与分析

* HTTP协议是怎样对头部和体部进行分隔的？

在HTTP协议中，头部（Header）和主体（Body）之间通过一个空行来进行分隔。这个空行是由回车符（CR，ASCII码为13）和换行符（LF，ASCII码为10）组成的，即"\r\n"。

对于头部中的请求（应答）行、请求（应答）头等部分，也用“\r\n”分隔，因此头部和主体之间实际上由“\r\n\r\n”分隔。

* 浏览器是根据文件的扩展名还是根据头部的哪个字段判断文件类型的？

根据头部的Content-Type字段判断文件类型。

* HTTP协议的头部是不是一定是文本格式？体部呢？

头部是纯文本格式，但是体部不一定，也可以有“.jpg”格式，需要根据Content-Type来确定。

* POST方法传递的数据是放在头部还是体部？两个字段是用什么符号连接起来的？

放在体部；

两个字段是使用“&”连接的。

# 讨论、心得

建议参考操作系统的实验设计，对整体代码给出一个框架，而不是全部由学生自己写。