

配置 Web 服务器实验报告

计算机网络课程实验报告 2

学院：网络空间安全学院

专业：密码科学与技术

学号：2112155

姓名：梁婧涵

实验要求

1. 搭建 Web 服务器（自由选择系统），并制作简单的 Web 页面，包含简单文本信息（包含专业、学号、姓名）、自己的 LOGO、自我介绍的音频信息。页面不要太复杂，追求的基本信息即可。
 2. 通过浏览器获取自己编写的 Web 页面，使用 Wireshark 捕获浏览器与 Web 服务器的过程，并进行简单的分析说明。
 3. 使用 HTTP，不要使用 HTTPS。
 4. 提交实验报告。
-

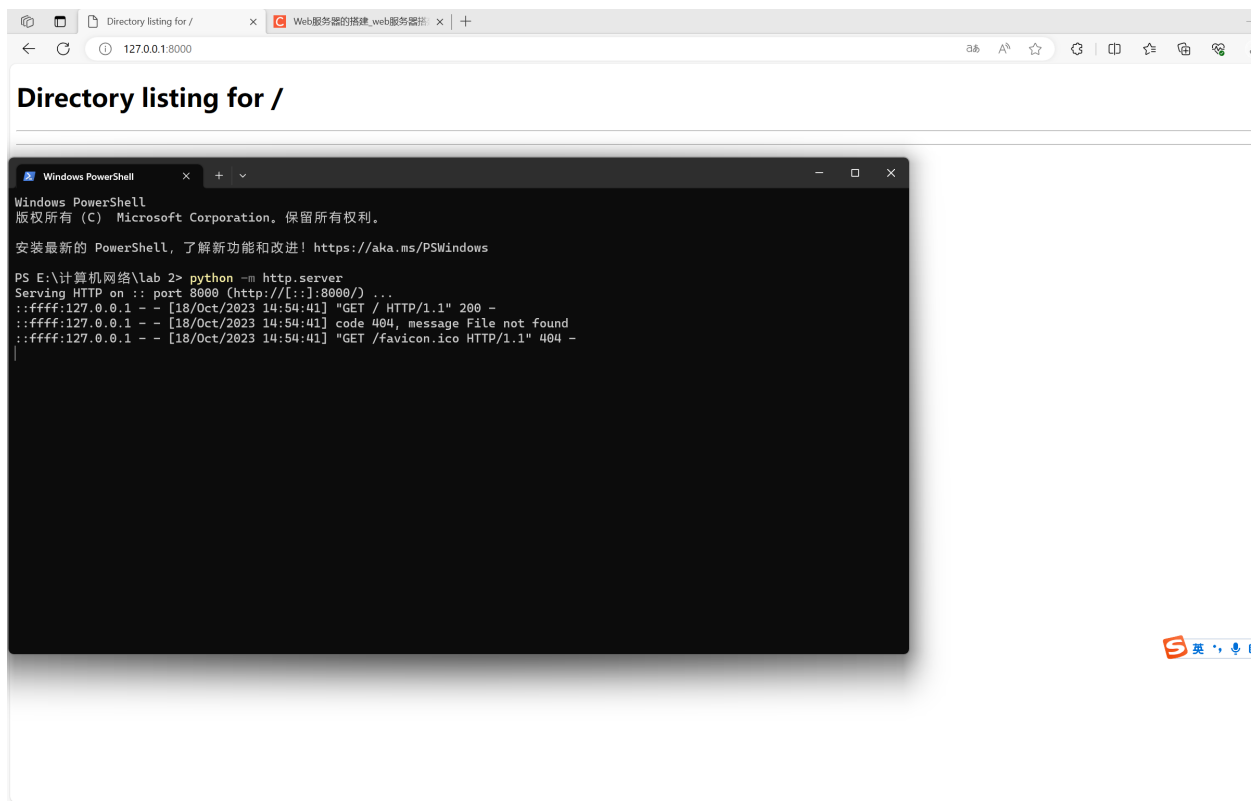
实验内容

1、Web 服务器搭建

- 1.1 使用 python 中的 web 模块快速搭建 web 服务器，在文件夹终端输入

Python

```
1 python -m http.server
```

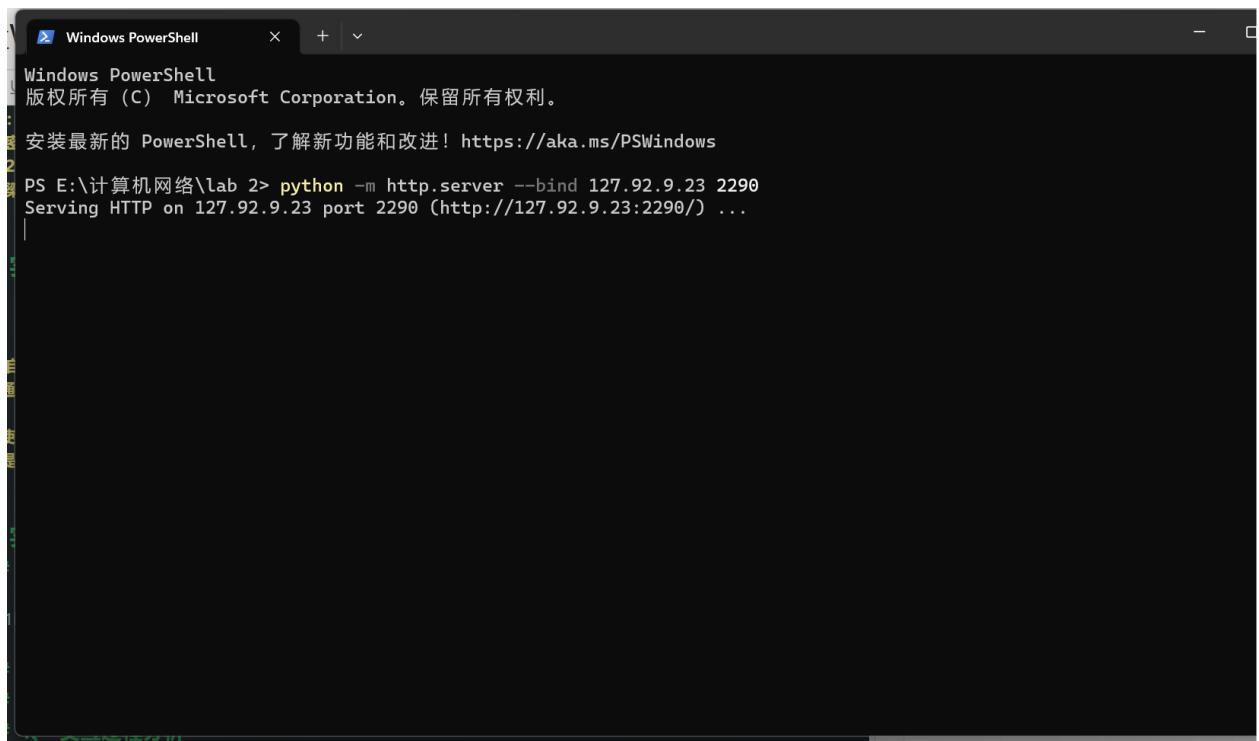


该ip地址是主机地址，端口号8000，浏览器中输入127.0.0.1:8000就可访问主页

1.2 使用bind规定了ip地址127.92.9.23，指定端口2290

Python

```
1 python -m http.server --bind 127.92.9.23 2290
```



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS E:\计算机网络\lab 2> python -m http.server --bind 127.92.9.23 2290
Serving HTTP on 127.92.9.23 port 2290 (http://127.92.9.23:2290/) ...
```

目的：抓包与主机区分开

在浏览器输入 <http://127.92.9.23:2290/lab2.html> 可访问网页

2、编写 Web 页面

编写 lab2.html，包含简单文本信息（至少包含专业、学号、姓名）、自己的 LOGO、自我介绍的音频信息

XML/HTML

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=
  0">
7     <title>我的个人网页</title>
8 </head>
9 <body leftmargin="140" rightmargin="140" bgcolor="white" text="black"
  </body>
10     <p align="right">welcome to my web</p>
11     <hr width="23%" align="right" color="white">
12     <h1><center>梁婧涵的个人网页</center></h1>
13     <h3>个人信息</h3>
14     <ul type="circle">
15         <li>姓名 : 梁婧涵</li>
16         <li>学号 : 2112155</li>
17         <li>专业 : 密码科学与技术</li>
18     </ul>
19     <hr color="white">
20     <hr color="white">
21     <p><center></center></p>
22 <audio controls="controls" loop="loop">
23     <source src="audio/个人介绍.mp3" type="audio/mp3"></source>
24 </audio>
25 </body>
26 </html>
```

3、Wireshark 捕获浏览器与 Web 服务器的交互过程

3.1 选择捕获选项进行捕获

交互动作在本机发生，回环操作，捕获时选择 Adapter for loop back traffic capture

3.2 交互

搭建好服务器在浏览器中打开网页实现交互，并用 wireshark 捕获

3.3 过滤获取目标信息

停止捕获，在过滤器命令行输入 ip.addr==127.92.9.23 过滤得到目标信息

4、交互过程分析

4.1 TCP 三次握手

16、17、18 行是捕获的 TCP 三次握手信息

The image shows a Wireshark capture of network traffic on a loopback adapter. The packet list shows several TCP and HTTP packets. Packets 16, 17, and 18 represent the TCP three-way handshake. Packet 19 is an HTTP GET request, and packet 21 is the corresponding HTTP response. The packet details pane for packet 19 is expanded, showing the Hypertext Transfer Protocol section. The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
13	1.251443	127.0.0.1	127.0.0.1	TCP	56	55395 →
14	1.954568	127.0.0.1	127.0.0.1	TCP	45	33210 →
15	1.954594	127.0.0.1	127.0.0.1	TCP	56	55393 →
16	2.553308	127.0.0.1	127.92.9.23	TCP	56	55550 →
17	2.553360	127.92.9.23	127.0.0.1	TCP	56	2290 → 5
18	2.553405	127.0.0.1	127.92.9.23	TCP	44	55550 →
19	2.626686	127.0.0.1	127.92.9.23	HTTP	830	GET /lab
20	2.626750	127.92.9.23	127.0.0.1	TCP	44	2290 → 5
21	2.628103	127.92.9.23	127.0.0.1	HTTP	148	HTTP/1.0
22	2.628124	127.0.0.1	127.92.9.23	TCP	44	55550 →

Frame 19: 830 bytes on wire (6640 bits), 830 bytes captured (6640 bits) on Adapter for loopback traffic capture, interface 0: Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.92.9.23

Transmission Control Protocol, Src Port: 55550, Dst Port: 2290

Hypertext Transfer Protocol

0000 02 00 00 00 45 00 03 3a 56 f2 40
0010 7f 00 00 01 7f 5c 09 17 d8 fe 08
0020 b7 33 50 4e 50 18 04 ff 07 ee 00
0030 2f 6c 61 62 32 2e 68 74 6d 6c 20
0040 31 2e 31 0d 0a 48 6f 73 74 3a 20
0050 32 2e 39 2e 32 33 3a 32 32 39 30
0060 6e 65 63 74 69 6f 6e 3a 20 6b 65
0070 69 76 65 0d 0a 43 61 63 68 65 2d
0080 6f 6c 3a 20 6d 61 78 2d 61 67 65
0090 65 63 2d 63 68 2d 75 61 3a 20 22
00a0 69 75 6d 22 3b 76 3d 22 31 31 38
00b0 69 63 72 6f 73 6f 66 74 20 45 64

1. 浏览器向 web 服务器发送 seq=0 的数据包，标志位 SYN=1，请求建立一个连接，浏览器进入 SYN_SENT 状态等待 web 服务器确认
2. web 服务器向浏览器发送 seq=0 的 SYN+ACK 响应数据包，标志位 SYN=1，web 服务器希望建立 TCP 连接。确认报文段 ACK=1，表示浏览器知道 web 接收了它的 SYN 包，进入 SYN_RECV。

3. 浏览器端向 web 发送 seq=1 的 ACK 响应数据包，标志位确认报文段 ACK=1，表示 web 端接收到 SYN 包。浏览器发出 ACK 报文后进入 ESTABLISHED，开始读写数据，web 收到来自浏览器的确认报文后进入 ESTABLISHED，进行数据读写。

实现了 TCP 的三次握手，之后进行数据传输

4.2 HTTP 请求

浏览器发给 HTTP 请求，第一个携带有效数据包。标志位 ACK 和 PUSH 两位是 1，ACK=1：TCP 规定在建立连接后所有传送的报文段都必须把 ACK 设置为 1；PUSH=1：报文段高优先级

1. 请求行

由请求方法字段 (Request Method)、URL 字段 (Request URI) 和 HTTP 协议版本字段 (RequestVersion) 3 个字段组成，用空格分隔

HTTP 协议的请求方法共八种：GET、POST、HEAD、PUT、DELETE、OPTIONS、TRACE、CONNECT。

- GET: 请求获取 Request-URI 所标识的资源。
- POST: 在 Request-URI 所标识的资源后附加新的数据。
- HEAD: 请求获取由 Request-URI 所标识的资源的响应消息报头。
- PUT: 请求服务器存储一个资源，并用 Request-URI 作为其标识。
- DELETE: 请求服务器删除 Request-URL 所标识的资源。
- OPTIONS: 请求查询服务器的性能，或者查询与资源相关的选项和需求。
- TRACE: 回显服务器收到的请求，主要用于测试或诊断。
- CONNECT: 保留将来使用。

2. 请求头

请求头由关键字 / 值对组成，每行一对，关键字和值用英文冒号 “:” 分隔。请求头通知服务器关于客户端请求的信息，包括 Host、Connection 等。请求头的最后会有一个空行，表示请求头结束

3. 请求体

请求体不在 GET 方法中使用，而在 POST 方法中使用。POST 方法适用于需要客户填写的场合，请求体就是用户填写的内容

本实验只需要向 web 端请求网页资源，GET 请求，没有请求体

分组 19 为例：

```
Wireshark - 分组 19 - Adapter for loopback traffic capture
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.92.9.23
> Transmission Control Protocol, Src Port: 55550, Dst Port: 2290, Seq: 1, Ack: 1, Len: 786
  > Hypertext Transfer Protocol
    > GET /lab2.html HTTP/1.1\r\n
      Host: 127.92.9.23:2290\r\n
      Connection: keep-alive\r\n
      Cache-Control: max-age=0\r\n
      sec-ch-ua: "Chromium";v="118", "Microsoft Edge";v="118", "Not=A?Brand";v="99"\r\n
      sec-ch-ua-mobile: ?0\r\n
      sec-ch-ua-platform: "Windows"\r\n
      Upgrade-Insecure-Requests: 1\r\n
      User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 Edg/118.0.2088.
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
      Sec-Fetch-Site: none\r\n
      Sec-Fetch-Mode: navigate\r\n
      Sec-Fetch-User: ?1\r\n
      Sec-Fetch-Dest: document\r\n
      Accept-Encoding: gzip, deflate, br\r\n
      Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
      If-Modified-Since: Wed, 18 Oct 2023 10:37:50 GMT\r\n
      \r\n
      [Full request URI: http://127.92.9.23:2290/lab2.html]
      [HTTP request 1/1]
      [Response in frame: 21]
```

请求行：GET /lab2.html HTTP/1.1，请求方法字段：GET，URL 字段：/lab2.html（请 lab2.html），HTTP 协议版本字段：HTTP/1.1\r\n（协议版本 1.1）

请求头：剩余部分到 \r\n 空行

4.3 HTTP 响应

Web 发给浏览器的 HTTP 响应数据包，标志位 ACK 和 PUSH 两位为 1。接下来以第 2 组为例分析 HTTP 响应的内容。HTTP 响应分为响应行，响应头和响应体三部分：

1. 状态行

HTTP 协议版本、状态码、状态码描述三部分组成。其中 HTTP 协议版本与 HTTP 请求一状态码由三位数字构成的，用来标识服务端对客户端这次请求的处理结果。状态码与状态述是一一对应的，常见的状态码及其状态描述有：

- 200 ok, 表示访问成功
- 404 NOT FOUND, 标识请求的 URL 路径没有对应的资源。比如在访问不存在路径时出现 404 状态码
- 403 Forbidden, 表示禁止访问，有可能时权限不够，也有可能是没有登陆
- 405 Method Not Allowed, 方法不支持，由于前后端约定不同导致
- 500 Internal Server Error, 表示服务器出现内部错误，一般是服务器的代码执行过程到了一些特殊情况 (服务器异常崩溃)，比较少见
- 504 Gateway Timeout, 表示超时，服务器负载比较大的时候，就可能会导致出现超情况
- 302 Move temporarily, 临时重定向

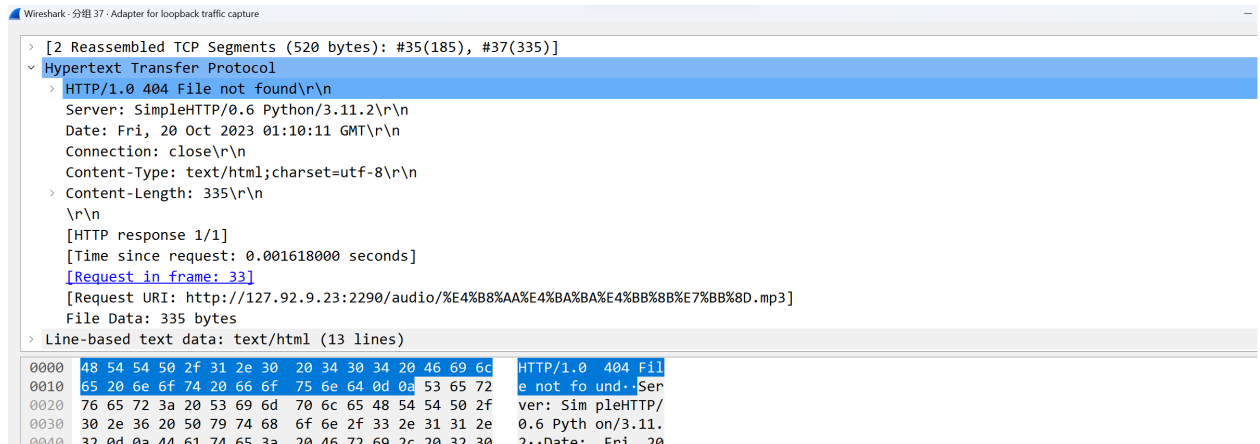
- 301 Moved Permanently, 永久重定向

2. 响应头

向客户端提供额外信息，如发送响应对象，响应者功能，响应相关的特殊指令，空行表示头结束

3. 响应体

服务器返回客户端的文本信息



响应行：HTTP/1.0 404 File not found（版本 1.0，标识请求的 URL 路径没有对应的资

响应头：

- Server: 服务器的信息，基于 Python3.11.2 的 SimpleHTTP 建立的服务器
- Date: 响应产生的时间
- Connect-type: 指定返回的数据类型，这里 text/html 代表返回 HTML 文档
- Content-length: 响应内容的长度（字节数）

响应体：html 文档

4.4 TCP 四次挥手

1. 浏览器向 Web 服务器发送序列号 seq=105 的 FIN 数据包，标志位 FIN=1，ACK=1，FIN=1 为关闭连接，数据包请求释放连接，关闭浏览器到 Web 浏览器的数据传送，浏览器进入 FIN_WAIT_1 状态
2. Web 服务器向浏览器发送序列号 seq=105, ACK=106（上一条是浏览器向 Web 服务器序列号 +1）的 ACK 数据包，标志位中 ACK 为 1 表示确认，该数据包用来告诉浏览器 Web 收到了释放连接的请求，并对该条请求进行两确认，Web 服务器进入 CLOSE_WAIT 状态

- Web 服务器向浏览器发送序列号 seq=787,ACK=106 的 FIN 数据包，标志位中 FIN ACK 为 1。Web 服务器向浏览器发送 FIN 报文，表示释放连接，关闭 Web 服务器至器的数据传送，Web 服务器进入 LAST_ACK 状态
- 浏览器发送序列号 seq=504,ACK=788（上一数据包序列号 +1）的 ACK 数据包，标中 ACK 为 1 表示确认。该数据包为浏览器收到 Web 端的请求释放连接报文发送的确认信息，Web 服务器进入 CLOSED 状态

挥手完成，web 服务器与浏览器连接断开

23,24,25,26 为例：

*Adapter for loopback traffic capture						
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)						
ip.addr==127.92.9.23						
No.	Time	Source	Destination	Protocol	Length	Info
19	2.626686	127.0.0.1	127.92.9.23	HTTP	830	GET /lab2.html HTTP/1.1
20	2.626750	127.92.9.23	127.0.0.1	TCP	44	2290 → 55550 [ACK] Seq=1 Ack=787 Win=2160384 Len=0
21	2.628103	127.92.9.23	127.0.0.1	HTTP	148	HTTP/1.0 304 Not Modified
22	2.628134	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [ACK] Seq=787 Ack=105 Win=327168 Len=0
23	2.628236	127.92.9.23	127.0.0.1	TCP	44	2290 → 55550 [FIN, ACK] Seq=105 Ack=787 Win=2160384 Len=0
24	2.628258	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [ACK] Seq=787 Ack=106 Win=327168 Len=0
25	2.628753	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [FIN, ACK] Seq=787 Ack=106 Win=327168 Len=0
26	2.628815	127.92.9.23	127.0.0.1	TCP	44	2290 → 55550 [ACK] Seq=106 Ack=788 Win=2160384 Len=0
27	2.636165	127.0.0.1	127.92.9.23	TCP	56	55552 → 2290 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER
28	2.636218	127.92.9.23	127.0.0.1	TCP	56	2290 → 55552 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=2
29	2.636289	127.0.0.1	127.92.9.23	TCP	44	55552 → 2290 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
30	2.636518	127.0.0.1	127.92.9.23	TCP	56	55553 → 2290 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER
31	2.636552	127.92.9.23	127.0.0.1	TCP	56	2290 → 55553 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=2
32	2.636585	127.0.0.1	127.92.9.23	TCP	44	55553 → 2290 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
33	2.733286	127.0.0.1	127.92.9.23	HTTP	686	GET /audio/%E4%B8%AA%E4%BA%BA%E4%BB%8B%E7%BB%8D.mp3 HTTP/1.1

4.5 其他

4.5.1 HTTP 报文

在过滤器中输入 http，筛选出所有的 HTTP 交互数据观察网页相关的数据传输

*Adapter for loopback traffic capture						
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)						
http						
No.	Time	Source	Destination	Protocol	Length	Info
6	0.737130	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback=e HT
8	0.737301	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
19	2.626686	127.0.0.1	127.92.9.23	HTTP	830	GET /lab2.html HTTP/1.1
21	2.628103	127.92.9.23	127.0.0.1	HTTP	148	HTTP/1.0 304 Not Modified
33	2.733286	127.0.0.1	127.92.9.23	HTTP	686	GET /audio/%E4%B8%AA%E4%BA%BA%E4%BB%8B%E7%BB%8D.mp3 HTTP/1.1
37	2.734904	127.92.9.23	127.0.0.1	HTTP	379	HTTP/1.0 404 File not found (text/html)
54	5.753061	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback=e HT
56	5.753446	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
105	10.769485	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback=e HT
107	10.769681	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
124	15.774847	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback=e HT
126	15.775002	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
145	21.205205	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback=e HT
147	21.205462	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
179	26.215512	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback=e HT

> Frame 21: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on

> Null/Loopback

> Internet Protocol Version 4, Src: 127.92.9.23, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 2290, Dst Port: 55550, Seq: 1, Ack

0000 02 00 00 00 45 00 00 90 2f d4 40 00 80 06 00 00 ...E... /
0010 7f 5c 09 17 7f 00 00 01 08 f2 d8 fe b7 33 50 4e
0020 19 e5 88 9a 50 18 20 f7 85 f7 00 00 48 54 54 50 ...P...
0030 2f 31 2e 30 20 33 30 34 20 4e 6f 74 20 4d 6f 64 .../1.0 304

在交互完 html 文档后，浏览器应该是对 html 文件进行了解析。文档中涉及到的还有一些和音频，浏览器也需要获取这些资源，于是向 Web 发送了获取图片的 GET 请求和获取音

的 GET 请求，Web 也做出了响应。

浏览器发送完所有图片的 GET 请求，Web 依次对图片请求进行响应；在所有完成所有 GET 图片的响应后，浏览器才会发出对于音频的 GET 请求，Web 再进行响应。

4.5.2 TCP 报文

浏览器发出 HTTP 请求后，Web 会发回一个 ACK 报文表示确认。在浏览器发出请求到服务器做出响应，Web 会向浏览器发送 PSH 报文传输数据。数据传输过程是发送方每一次 write，都会将这一次的数据打包成一个或多个 TCP 报文段，并将最后一个 TCP 报文段标志 PSH。只有当接收方收到包含了 PSH 标志的报文才会将接收缓冲区中的所有数据推送给进程，保证了数据的完整性。（如果发送缓冲区满了也会把缓冲区数据打包发送，同理接收缓冲区满了也会推送给应用进程）

No.	Time	Source	Destination	Protocol	Length	Info
16	2.553308	127.0.0.1	127.92.9.23	TCP	56	55550 → 2290 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER
17	2.553360	127.92.9.23	127.0.0.1	TCP	56	2290 → 55550 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=2
18	2.553405	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [ACK] Seq=1 Ack=1 Win=327424 Len=0
19	2.626686	127.0.0.1	127.92.9.23	HTTP	830	GET /lab2.html HTTP/1.1
20	2.626750	127.92.9.23	127.0.0.1	TCP	44	2290 → 55550 [ACK] Seq=1 Ack=787 Win=2160384 Len=0
21	2.628103	127.92.9.23	127.0.0.1	HTTP	148	HTTP/1.0 304 Not Modified
22	2.628134	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [ACK] Seq=787 Ack=105 Win=327168 Len=0
23	2.628236	127.92.9.23	127.0.0.1	TCP	44	2290 → 55550 [FIN, ACK] Seq=105 Ack=787 Win=2160384 Len=0
24	2.628258	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [ACK] Seq=787 Ack=106 Win=327168 Len=0
25	2.628753	127.0.0.1	127.92.9.23	TCP	44	55550 → 2290 [FIN, ACK] Seq=787 Ack=106 Win=327168 Len=0
26	2.628815	127.92.9.23	127.0.0.1	TCP	44	2290 → 55550 [ACK] Seq=106 Ack=788 Win=2160384 Len=0
27	2.636165	127.0.0.1	127.92.9.23	TCP	56	55552 → 2290 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER
28	2.636218	127.92.9.23	127.0.0.1	TCP	56	2290 → 55552 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=2
29	2.636289	127.0.0.1	127.92.9.23	TCP	44	55552 → 2290 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
30	2.636518	127.0.0.1	127.92.9.23	TCP	56	55553 → 2290 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER
31	2.636552	127.92.9.23	127.0.0.1	TCP	56	2290 → 55553 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=2
32	2.636585	127.0.0.1	127.92.9.23	TCP	44	55553 → 2290 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
33	2.733286	127.0.0.1	127.92.9.23	HTTP	686	GET /audio/%E4%B8%AA%E4%BA%BA%E4%BB%8B%E7%BB%8D.mp3 HTTP/1.1
34	2.733349	127.92.9.23	127.0.0.1	TCP	44	2290 → 55552 [ACK] Seq=1 Ack=643 Win=2160640 Len=0
35	2.734812	127.92.9.23	127.0.0.1	TCP	229	2290 → 55552 [PSH, ACK] Seq=1 Ack=643 Win=2160640 Len=185 [TCP seg
36	2.734855	127.0.0.1	127.92.9.23	TCP	44	55552 → 2290 [ACK] Seq=643 Ack=186 Win=2160896 Len=0
37	2.734904	127.92.9.23	127.0.0.1	HTTP	379	HTTP/1.0 404 File not found (text/html)
38	2.734922	127.0.0.1	127.92.9.23	TCP	44	55552 → 2290 [ACK] Seq=643 Ack=521 Win=2160640 Len=0
39	2.734989	127.92.9.23	127.0.0.1	TCP	44	2290 → 55552 [FIN, ACK] Seq=521 Ack=643 Win=2160640 Len=0
40	2.735011	127.0.0.1	127.92.9.23	TCP	44	55552 → 2290 [ACK] Seq=643 Ack=522 Win=2160640 Len=0
41	2.735540	127.0.0.1	127.92.9.23	TCP	44	55552 → 2290 [FIN, ACK] Seq=643 Ack=522 Win=2160640 Len=0
42	2.735601	127.92.9.23	127.0.0.1	TCP	44	2290 → 55552 [ACK] Seq=522 Ack=644 Win=2160640 Len=0
43	2.988618	192.168.80.1	192.168.80.1	ICMP	128	Destination unreachable (Host unreachable)
44	3.439758	127.0.0.1	127.0.0.1	TCP	45	55137 → 33210 [ACK] Seq=1 Ack=1 Win=8310 Len=1
45	3.439796	127.0.0.1	127.0.0.1	TCP	56	33210 → 55137 [ACK] Seq=1 Ack=2 Win=8434 Len=0 SLE=1 SRE=2
46	3.439805	127.0.0.1	127.0.0.1	TCP	45	55436 → 33210 [ACK] Seq=1 Ack=1 Win=8418 Len=1

4.5.3 问题

抓包前如果已经通过 Web 服务器访问过了网页，在加载时会出现 GET 图片状态码为 304 问题，是浏览器缓存导致的，浏览器将同一个链接认为是相同的请求，所以便没有往 Web 发送该请求以致于设置失败。刷新网页即可

