2110688 史文天

网页抓包分析

（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（至少包含专业、学号、姓名）、自己的LOGO、自我介绍的音频信息。页面不要太复杂，包含要求的基本信息即可。

（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服务器的交互过程，并进行简单的分析说明。

（3）使用HTTP，不要使用HTTPS。

实验过程

搭建服务器

- 使用Node.js在本地部署服务器。

```
const express = require('express');
const app = express();
const path = require('path');
const os = require('os');

app.use(express.static(path.join(__dirname, './')));

const port = 6200;

app.listen(port, '0.0.0.0', () => {
    const networkInterfaces = os.networkInterfaces();
    let ipAddress = '127.0.0.1';

    // 遍历网络接口，找到非 localhost 下的 IPv4 地址

    Object.keys(networkInterfaces).forEach((interfaceName) => {
        networkInterfaces[interfaceName].forEach((networkInterface) => {
            if (!networkInterface.internal && networkInterface.family ===
'IPv4') {
                ipAddress = networkInterface.address;
            }
        });
    });
    console.log(`Availble on:`);
    console.log(`http://localhost:${port}`);
    console.log(`http://${ipAddress}:${port}`);
});
```

在文件夹下打开命令行窗口，运行Node命令即可部署服务器。

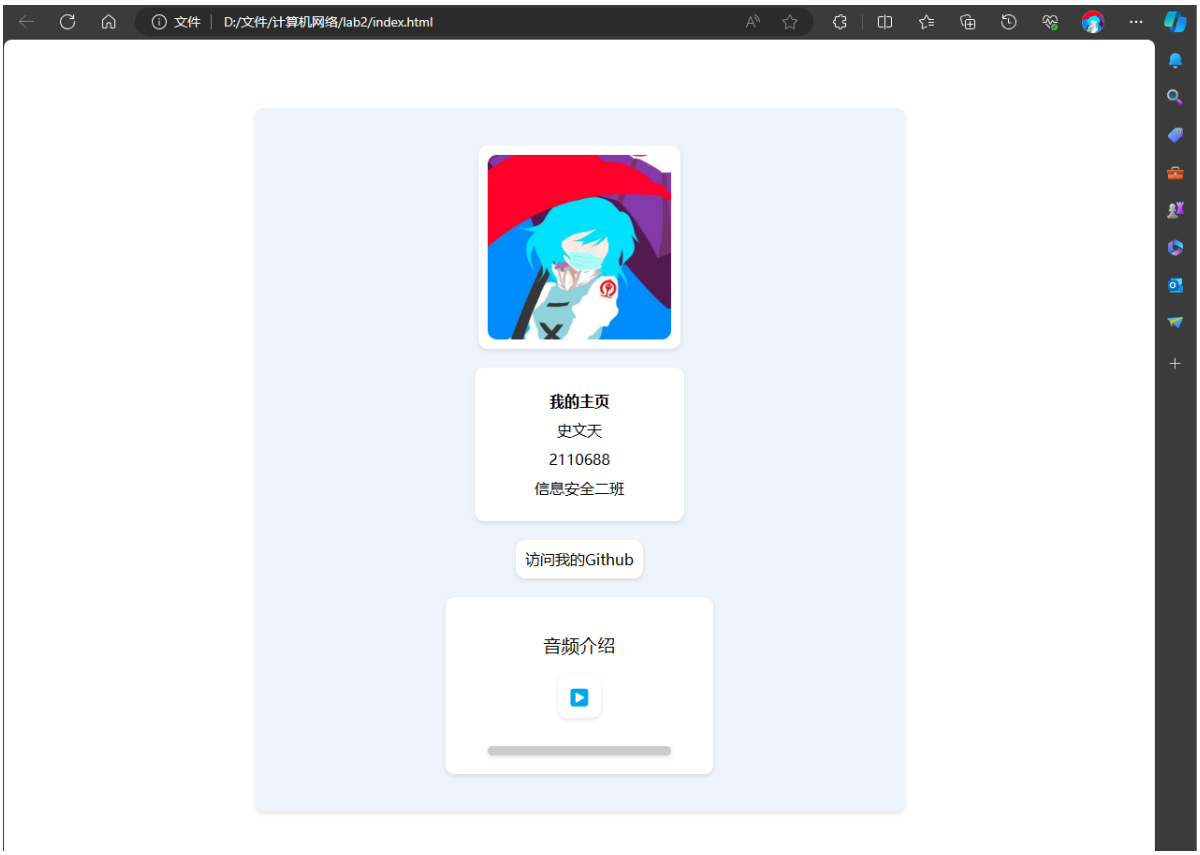
网页HTML设计

- 网页包括文本信息，logo，音频信息，使用JavaScript实现了音频播放部分。省略了控制风格的css部分，具体参考html文件。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>2110951自我介绍</title>
    <style>
    </style>
</head>

<body>
    <div id="app">
        <div class="container">
            <div class="logo">
                <img src="icon.jpg" alt="logo" class="icon">
            </div>
            <div class="info">
                <p class="title">我的主页</p>
                <p>史文天</p>
                <p>2110688</p>
                <p>信息安全二班</p>
            </div>
            <a class="github" href="https://github.com/FlandreScarlet36"
target="_blank">
                访问我的Github
            </a>
            <div class="audio">
                <p>音频介绍</p>
                <audio id="audioElement" src="audio.wav"></audio>
                <button id="playButton" onclick="togglePlay()"></button>
                <div class="progressBar">
                    <div class="progressBarFill" :style="{ width: progress +
'%' }"></div>
                </div>
            </div>
        </div>
    </div>
    <script>
        var audioElement = document.getElementById('audioElement');
        audioElement.addEventListener('timeupdate', updateProgressBar);
        audioElement.addEventListener('ended', resetAudio);

        var isPlaying = false;
        var playButton = document.getElementById('playButton');
        playButton.textContent = isPlaying ? '⏸' : '▶';
        var progress = 0;

        function togglePlay() {
            if (isPlaying) {
```

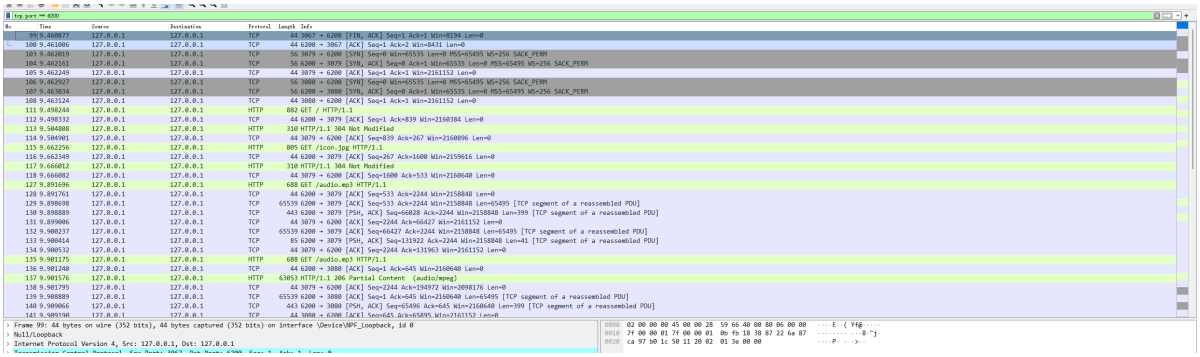
```
                audioElement.pause();
                playButton.textContent = '▶️';
            } else {
                audioElement.play();
                playButton.textContent = '⏸️';
            }
            isPlaying = !isPlaying;
        }

        function updateProgressBar() {
            var progressBarFill = document.querySelector('.progressBarFill');
            progress = (audioElement.currentTime / audioElement.duration) *
100;
            progressBarFill.style.width = progress + '%';
        }

        function resetAudio() {
            audioElement.currentTime = 0;
            progress = 0;
            isPlaying = false;
            playButton.textContent = '▶️';
        }
        var audioElement = new Audio('audio.wav');
        audioElement.addEventListener('timeupdate', updateProgressBar);
        audioElement.addEventListener('ended', resetAudio);
        var isPlaying = false;
        var progress = 0;
    </script>
</body>

</html>
```

效果展示



抓包分析

使用ip地址和tcp.port条件作为筛选器，只留下该网页与服务器的通讯记录。



三次握手分析

```
56 3080 → 6200 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
56 6200 → 3080 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
44 3080 → 6200 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
```

第一次握手由客户端向服务器发送请求，可以看到Seq=0，代表初次建立连接，这次握手将标志位SYN置为1，表示请求建立连接。

```
Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
```

第二次握手由服务器向客户端发送反馈，SYN和ACK为1，表示服务器同意进行连接。

```
Flags: 0x012 (SYN, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
```

第三次握手看到Seq=1，表示并非第一次建立连接。ACK为1，表示收到了服务器的回复。

```
Flags: 0x010 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
```

HTTP部分

```
HTTP      882 GET / HTTP/1.1
HTTP      310 HTTP/1.1 304 Not Modified
HTTP      805 GET /icon.jpg HTTP/1.1
HTTP      310 HTTP/1.1 304 Not Modified
HTTP      688 GET /audio.mp3 HTTP/1.1
HTTP      688 GET /audio.mp3 HTTP/1.1
HTTP    63053 HTTP/1.1 206 Partial Content  (audio/mpeg)
HTTP    63053 HTTP/1.1 206 Partial Content  (audio/mpeg)
```

第一个GET请求用于获取HTML界面。状态码304表示客户端有最新的缓存资源，并与服务器上的资源相同，因此服务器发送一个空包和304状态码，告诉客户端可以使用本地缓存资源，用于减少网络流量的浪费。

获取HTML界面后继续发送GET请求，获取icon.jpg，同样返回304状态码。

获取audio.MP3，返回206状态码，这种响应是在客户端表明自己只需要目标URL上的部分资源的时候返回的，是由于客户端在加载一个较大的文件，无法用一个包发送所有文件，因此采用断点续传发送。

四次挥手

```
44 6200 → 3079 [FIN, ACK] Seq=194972 Ack=2244 Win=2158848 Len=0
44 3079 → 6200 [ACK] Seq=2244 Ack=194973 Win=2098176 Len=0
44 6200 → 3080 [FIN, ACK] Seq=194440 Ack=645 Win=2160640 Len=0
44 3080 → 6200 [ACK] Seq=645 Ack=194441 Win=2098176 Len=0
```

第一次挥手客户端向服务器发送带有FIN和ACK标志位的包，FIN表示客户端要求关闭连接，ACK=2244表示确认了服务器发送的序号为2244的数据，查找后发现是audio.MP3的数据。

```
  688 GET /audio.mp3 HTTP/1.1
   44 6200 → 3079 [ACK] Seq=533 Ack=2244 Win=2158848 Len=0
65539 6200 → 3079 [ACK] Seq=533 Ack=2244 Win=2158848 Len=65495 [TCP segment of a reassembled PDU]
  443 6200 → 3079 [PSH, ACK] Seq=66028 Ack=2244 Win=2158848 Len=399 [TCP segment of a reassembled PDU]
   44 3079 → 6200 [ACK] Seq=2244 Ack=66427 Win=2161152 Len=0
65539 6200 → 3079 [ACK] Seq=66427 Ack=2244 Win=2158848 Len=65495 [TCP segment of a reassembled PDU]
   85 6200 → 3079 [PSH, ACK] Seq=131922 Ack=2244 Win=2158848 Len=41 [TCP segment of a reassembled PDU]
   44 3079 → 6200 [ACK] Seq=2244 Ack=131963 Win=2161152 Len=0
```

第二次挥手服务器向客户端发送带有ACK标志位的包，表示对客户端请求的确认。

第三次挥手客户端收到服务器的确认信息，发送带有FIN和ACK标志位的包，表示客户端准备关闭连接，在客户端发来ACK包后就会关闭连接。

第四次挥手服务器向客户端发送带有ACK标志位的包，关闭服务器连接，同时客户端服务器也准备关闭连接。

四次挥手后，等待2MSL后，客户端也关闭连接，TCP连接成功断开。