南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

计算机网络课程报告

配置 Web 服务器，编写简单页面，分析交互过程

学号：2013018

姓名：许健

年级：2020 级

专业：信息安全

2022 年 10 月 28 日

目录

一、 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 (至少包含专业、学号、姓名) 和自己的 LOGO
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. 提交实验报告

二、 Web 服务器搭建、编写 Web 页面

(一) Web 服务器搭建

本次实验采用 phpstudy_Pro 搭建网站，是之前实习实训所使用的工具。配置如图1所示，提供服务的 Port 为 8888。

图 1: phpstudy 配置

(二) 编写 Web 页面

编写的 index.html 文档包含简单的个人信息和 LOGO。在 edge 浏览器中启动，页面如图2所示。

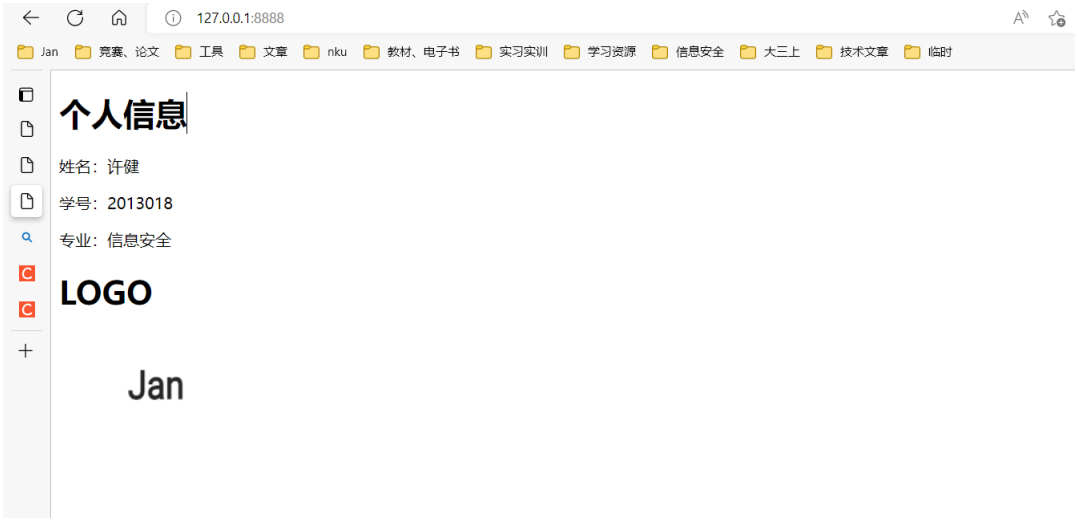

图 2: 实验网站

三、 Wireshark 捕获浏览器与 Web 服务器的交互过程

(一) 捕获情况

打开 Wireshark，然后在浏览器中键入网址 127.0.0.1:8888，这时交互的数据包会被捕获。我们可以使用 Wireshark 设置过滤器。过滤 port 为 8888 的数据包。

捕获结果如图3所示, 包括 TCP 数据包和 HTTP 应用报文，源 IP 和目的 IP 均为 127.0.0.1，服务器使用的是 8888 端口，客户端使用的是 6091 端口。

56	5.710952	127.0.0.1	127.0.0.1	TCP	56	6091 → 8888 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
57	5.711008	127.0.0.1	127.0.0.1	TCP	56	8888 → 6091 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
58	5.711069	127.0.0.1	127.0.0.1	TCP	44	6091 → 8888 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
59	5.711293	127.0.0.1	127.0.0.1	HTTP	1043	GET / HTTP/1.1
60	5.711308	127.0.0.1	127.0.0.1	TCP	44	8888 → 6091 [ACK] Seq=1 Ack=1000 Win=2619648 Len=0
61	5.711911	127.0.0.1	127.0.0.1	HTTP	276	HTTP/1.1 304 Not Modified
62	5.711943	127.0.0.1	127.0.0.1	TCP	44	6091 → 8888 [ACK] Seq=1000 Ack=233 Win=2619392 Len=0
63	5.746421	127.0.0.1	127.0.0.1	HTTP	961	GET /LOGO.png HTTP/1.1
64	5.746465	127.0.0.1	127.0.0.1	TCP	44	8888 → 6091 [ACK] Seq=233 Ack=1917 Win=2618880 Len=0
65	5.746891	127.0.0.1	127.0.0.1	HTTP	275	HTTP/1.1 304 Not Modified
66	5.746944	127.0.0.1	127.0.0.1	TCP	44	6091 → 8888 [ACK] Seq=1917 Ack=464 Win=2619136 Len=0
80	6.138036	127.0.0.1	127.0.0.1	TCP	56	6096 → 8888 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
81	6.138089	127.0.0.1	127.0.0.1	TCP	56	8888 → 6096 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
82	6.138147	127.0.0.1	127.0.0.1	TCP	44	6096 → 8888 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
103	10.756732	127.0.0.1	127.0.0.1	TCP	44	8888 → 6091 [FIN, ACK] Seq=464 Ack=1917 Win=2618880 Len=0
104	10.756774	127.0.0.1	127.0.0.1	TCP	44	6091 → 8888 [ACK] Seq=1917 Ack=465 Win=2619136 Len=0
161	22.555982	127.0.0.1	127.0.0.1	TCP	44	6091 → 8888 [FIN, ACK] Seq=1917 Ack=465 Win=2619136 Len=0
162	22.556012	127.0.0.1	127.0.0.1	TCP	44	8888 → 6091 [ACK] Seq=465 Ack=1918 Win=2618880 Len=0

图 3: Wireshark 捕获结果

(二) 分析交互过程

1. TCP 建立连接

刚开始需要 TCP 三报文握手建立连接，在网上找了一张形象的图表示 TCP 建立连接的过程，与上文捕获报文对比发现正好对照，至此客户端的 6091 端口和服务端的 8888 端口建立连接。

第 1 次握手建立连接时，客户端向服务器发送 SYN 报文（SEQ=x，SYN=1），并进入 SYN_SENT 状态，等待服务器确认。

第 2 次握手实际上是分两部分来完成的，即 SYN+ACK（请求和确认）报文。服务器收到了客户端的请求，向客户端回复一个确认信息（ACK=x+1）。服务器再向客户端发送一个 SYN 包（SEQ=y）建立连接的请求，此时服务器进入 SYN_RECV 状态。

第 3 次握手，是客户端收到服务器的回复（SYN+ACK 报文）。此时，客户端也要向服务器发送确认包（ACK）。此包发送完毕客户端和服务器进入 ESTABLISHED 状态，完成 3 次握手。

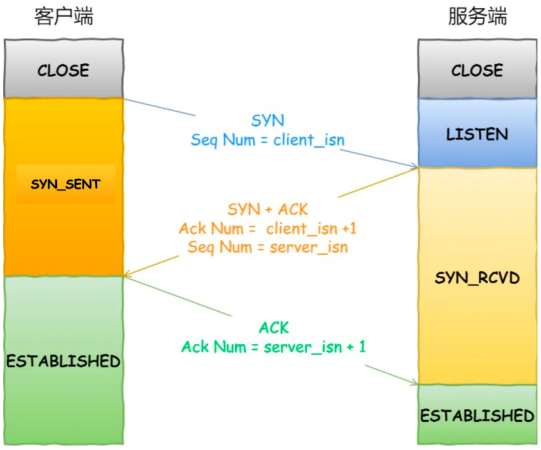

图 4: TCP 建立连接

2. HTTP 请求

客户端向服务端发送 GET 的 HTTP 请求，查看报文的具体内容，包括的信息有 HTTP 版本、操作系统、Cookie 信息、If-Modified-since 字段、Purpose 等等。

图 5: GET 请求报文

服务器端接收客户端的 GET 请求，并回复 304 Not-Modified，说明服务端已经执行了 GET，但文件未变化。

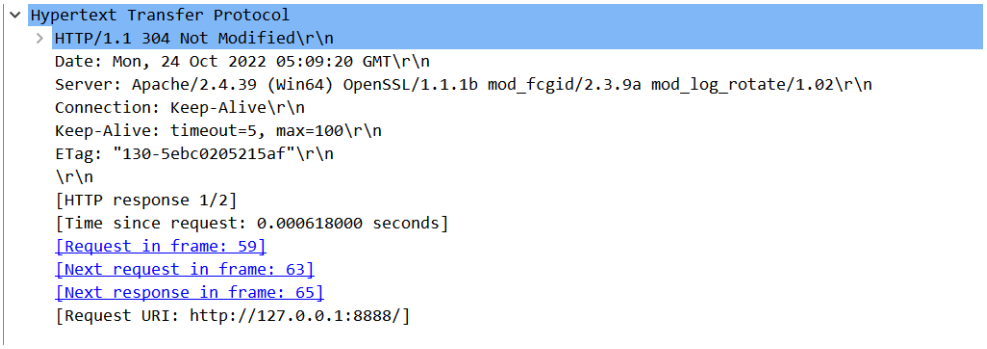

图 6: 服务器端回复报文

然后客户端接着请求 LOGO.png 图片信息，服务端接着回复 304 NOt-Modified，至此 Web

服务端与浏览器的交互结束。

在双方传递报文数据的过程中，TCP 会采用累计确认机制，其中使用到了 Seq 序列号和 Ack 确认号字段，说明 TCP 提供的是可靠传输服务。

3. TCP 释放连接

最后，TCP 四报文挥手释放连接，在网上找了一张形象的图表示 TCP 释放连接的过程，与上文捕获报文对比发现正好对照。在断开连接之前客户端和服务器都处于 ESTABLISHED 状态，双方都可以主动断开连接，以客户端主动断开连接为优。

第一次挥手：客户端打算断开连接，向服务器发送 FIN 报文 (FIN 标记位被设置为 1，1 表示为 FIN，0 表示不是)，FIN 报文中会指定一个序列号，之后客户端进入 FIN_WAIT_1 状态。也就是客户端发出连接释放报文段 (FIN 报文)，指定序列号 seq = u，主动关闭 TCP 连接，等待服务器的确认。

第二次挥手：服务器收到连接释放报文段 (FIN 报文) 后，就向客户端发送 ACK 应答报文，以客户端的 FIN 报文的序列号 seq+1 作为 ACK 应答报文段的确认序列号 ack = seq+1 = u + 1。接着服务器进入 CLOSE_WAIT(等待关闭) 状态，此时的 TCP 处于半关闭状态 (下面会说什么是半关闭状态)，客户端到服务器的连接释放。客户端收到来自服务器的 ACK 应答报文段后，进入 FIN_WAIT_2 状态。

第三次握手：服务器也打算断开连接，向客户端发送连接释放 (FIN) 报文段，之后服务器进入 LASK_ACK(最后确认) 状态，等待客户端的确认。服务器的连接释放 (FIN) 报文段的 FIN=1，ACK=1，序列号 seq=m，确认序列号 ack=u+1。

第四次握手：客户端收到来自服务器的连接释放 (FIN) 报文段后，会向服务器发送一个 ACK 应答报文段，以连接释放 (FIN) 报文段的确认序号 ack 作为 ACK 应答报文段的序列号 seq，以连接释放 (FIN) 报文段的序列号 seq+1 作为确认序号 ack。

之后客户端进入 TIME_WAIT(时间等待) 状态，服务器收到 ACK 应答报文段后，服务器就进入 CLOSE(关闭) 状态，到此服务器的连接已经完成关闭。

客户端处于 TIME_WAIT 状态时，此时的 TCP 还未释放掉，需要等待 2MSL 后，客户端才进入 CLOSE 状态。

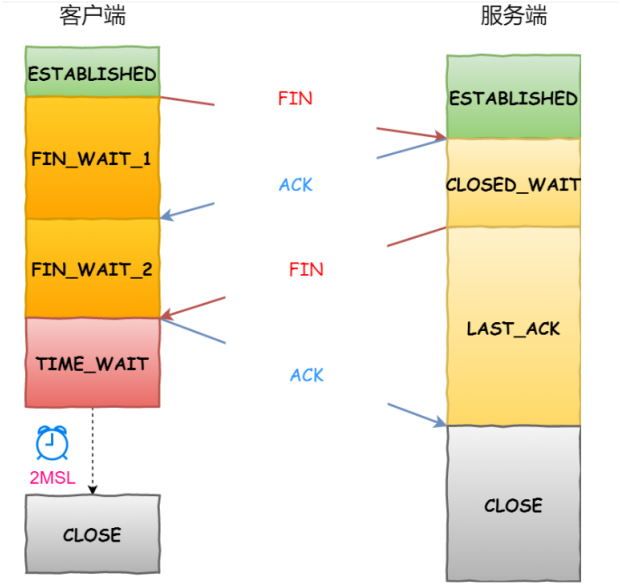

图 7: TCP 释放连接