华东师范大学数据科学与工程学院计算机网络实验报告

课程名称:	计算机网络与编程	指导教师:	张召
姓　　名:	孙嘉	学　　号:	10235501445
年　　级:	2023 级	实践名称:	协作控制

目录

1 题目要求

实验目的

- 熟悉 HTTP、HTTPS、SMTP 和 POP3 协议的工作原理。
- 了解这些协议在实际网络中的运行过程。

实验任务

- 使用 Wireshark 分析 HTTP、HTTPS、SMTP 和 POP3 协议的数据传输过程。

实验内容

HTTP/HTTPS

- 分析 HTTP 协议的无状态特性及其安全局限。
- 对比 HTTPS 如何通过 SSL/TLS 加密解决 HTTP 的安全问题。

- 捕获并分析 HTTP GET 和 POST 请求及其响应报文。

SMTP/POP3

- 发送和接收邮件以捕获 SMTP 和 POP3 协议数据包。
- 分析邮件发送（SMTP）和接收（POP3）过程中涉及的数据包结构。

实践操作

- 配置 Wireshark 捕获 HTTP、HTTPS、SMTP 和 POP3 数据包。
- 利用工具追踪流查看具体交互内容，并理解 Base64 编码的邮件数据。

2 项目实现思路

2.1 Task 1 - HTTP 抓包

2.1.1 准备工作

确保抓包数据的准确性和有效性，需执行以下步骤：

1. 清空 Web 浏览器缓存，以避免使用缓存数据。
2. 启动 Wireshark 抓包工具，设置为捕获数据包模式，等待数据捕获。

2.1.2 抓取 HTTP 请求网络包

1. 访问目标网站：在浏览器中输入 `http://fishros.com` 并访问。
2. Wireshark 捕获过程中的 HTTP 请求数据包。

2.2 Task 2 - HTTP 请求与响应分析

2.2.1 抓包筛选与准备

1. 清除浏览器缓存，确保数据新鲜。
2. 启动 Wireshark，开始捕获网络包。
3. 访问特定网址，刷新页面以生成 HTTP 请求。
4. 应用过滤器（如 'http'），筛选出 HTTP 通信数据。

2.2.2 标识与结构分析

1. 选取 HTTP 请求，分析其组成部分：
    - 请求方法（GET/POST 等）
    - 请求 URL
    - 协议版本
    - 请求头（User-Agent, Accept 等）
    - 请求正文（POST 请求常见）
2. 对应 HTTP 响应的识别与比较：
    - 状态码
    - 响应头
    - 响应正文内容

2.3 Task 3 - POST 与 GET 请求的对比

2.3.1 抓包区分不同请求类型

1. 观察日常浏览，特别注意表单提交触发的 POST 请求。
2. 实际操作于登录或注册页面，捕获 POST 数据包。

2.3.2 请求与响应特点分析

- GET：与 URL 参数，无请求正文。
- POST：请求正文含数据，Content-Length 与 Content-Type。
- 响应差异：Content-Type 依请求逻辑变化。

2.4 Task 4 - 邮件传输协议分析（SMTP 与 POP3）

2.4.1 邮件发送与接收的抓包操作

1. 配置 Foxmail 邮箱，验证邮件收发功能。
2. 发送邮件时捕获 SMTP 数据包。
3. 收取邮件时捕获 POP3 数据包。

2.4.2 SMTP 与 POP3 协议交互分析

- SMTP 命令与响应：EHLO, MAIL FROM, RCPT TO, 250 响应等。
- POP3 命令与响应：USER, PASS, STAT 等及其功能解析。

2.5 Task 5 - SMTP 交互过程深入解析

2.5.1 代表性 SMTP 交互选例

1. 选择完整的邮件发送过程数据包。

2.5.2 交互报文详细解读

1. 解析每一步的命令与响应：
   - EHLO 的认证与服务器认可。
   - MAIL FROM 与 RCPT TO 设定邮件路径。
   - 数据传输控制标志。
2. 关键环节分析确保邮件传输安全与正确。

3 功能实现情况

3.1 task1

3.1.1 抓取 HTTP 请求数据包

完成准备工作后，使用 Wireshark 成功捕获访问 http://fishros.com 的 HTTP 请求数据包，验证了抓包机制的有效性。

3.1.2 请求包的详细分析

HTTP 请求的各个组成部分：

请求头部分析

- **Host:** drive.wps.cn — 目标服务器。
- **Connection:** Keep-Alive — 维持连接。
- **Accept-Encoding:** gzip, deflate — 支持的压缩格式。
- **Accept-Language:** zh-CN,en,* — 语言偏好。
- **User-Agent:** Mozilla/5.0 — 客户端信息。

请求正文特点 GET 请求正文为空，所有参数通过 URL 传递。

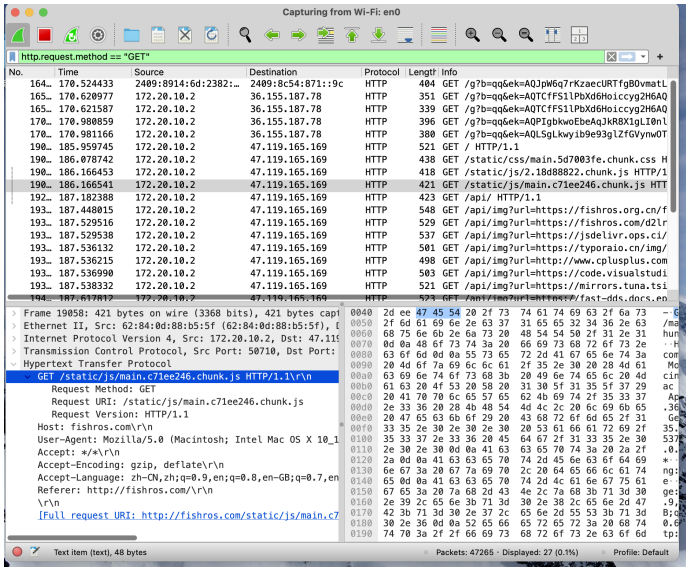

图 1: 请求包

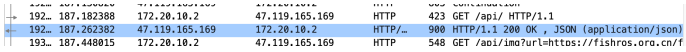

图 2: 响应报文

3.2 task2

3.2.1 状态行与基本响应信息

协议版本 HTTP/1.1

状态码 200

状态码描述 OK

此响应表明服务器成功处理请求，资源已正常返回。

3.2.2 响应头部详情

Server nginx/1.18.0 — 服务器软件及版本，使用高性能的 *Nginx*。

Date Mon, 07 Apr 2025 02:08:29 GMT — 响应生成时间，采用 *GMT*。

Content-Type application/json; charset=utf-8 — 数据类型为 *JSON*，字符编码 *UTF-8*。

Content-Length 1984 — 响应正文长度，便于客户端处理。

Connection keep-alive — 保持连接，优化后续请求。

Content-Encoding gzip — 数据已使用 *gzip* 压缩，提高传输效率。

Vary Accept-Encoding — 缓存策略考虑客户端编码偏好。

3.2.3 响应正文概览

响应正文以 JSON 格式提供，长度为 1984 字节，经过 gzip 压缩，确保高效传输且需按 UTF-8 解码。在对比

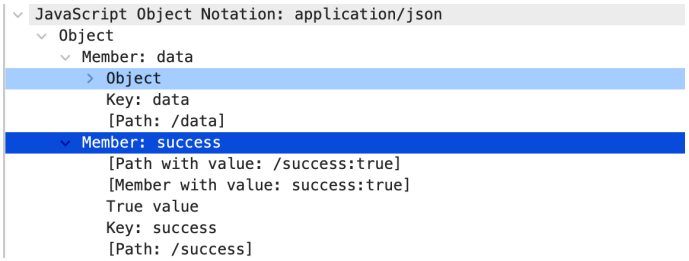

图 3: 响应正文

分析 HTTP 响应报文和请求报文的组成和区别时，我们可以从以下几个关键方面进行：

3.2.4 请求报文组成与特点：

1. **请求行**：包含请求方法（如 GET、POST）、请求的 URL 和使用的 HTTP 协议版本。在您的案例中，虽然没有直接提供请求行内容，但通常 GET 请求会指定访问的 URL，如 'http://fishros.com'，并使用 HTTP/1.1 或更高版本。

2. **请求头（Request Headers）**：

- **Host**: 指示目标服务器的域名，例如 'drive.wps.cn'，但在访问 'http://fishros.com'时，实际 Host 应与目标一致。
- **Connection**: 表明连接的处理方式，如 'Keep-Alive'用于维持连接不关闭。
- **Accept-Encoding**: 客户端支持的压缩格式，如 'gzip, deflate'。
- **Accept-Language**: 客户端首选的语言，这里为 'zh-CN,en,*'。
- **User-Agent**: 描述客户端软件信息，如浏览器类型和版本。

3. **请求正文（Request Body）**：对于 GET 请求，正文通常是空的，参数通过 URL 传递。POST 等其他类型请求则可能包含数据。

3.2.5 响应报文组成与特点：

1. **状态行**：包括 HTTP 协议版本、状态码和状态码描述。例如，'HTTP/1.1 200 OK'表明请求成功。

2. **响应头（Response Headers）**：

- **Server**: 服务器软件及其版本，如 'nginx/1.18.0'。
- **Date**: 响应生成的时间，遵循 GMT 时间标准。
- **Content-Type**: 指示响应数据的类型，这里是 'application/json; charset=utf-8'，意味着数据是 JSON 格式且编码为 UTF-8。

- **Content-Length**: 响应正文的未压缩字节数，这里是 1984 字节。
- **Connection**: 同样可能是 'keep-alive'，表明连接可以复用。
- **Content-Encoding**: 如果数据被压缩，如 'gzip'，表示响应正文是压缩过的。
- **Vary**: 指示响应依据哪些请求头来变化，这里为 'Accept-Encoding'，说明服务器根据客户端接受的编码来调整响应。

3. **响应正文 (Response Body)**：实际的数据部分，以指定的格式（如 JSON）发送，且在本例中经过 gzip 压缩，长度为 1984 字节，需解压后按 UTF-8 编码读取。

3.2.6 区别总结：

- **目的**：请求报文是客户端向服务器提出请求，而响应报文是服务器对请求的回应。
- **结构**：请求报文包含请求行、请求头和可能的请求正文；响应报文有状态行、响应头和响应正文。
- **正文内容**：请求正文在 GET 请求中通常为空，而在 POST 等请求中携带数据；响应正文则包含服务器返回的数据，如 HTML、JSON 或其他格式。
- **编码和压缩**：响应报文可能包含关于数据编码和压缩的信息（如 Content-Encoding），而请求报文则表达客户端的接受偏好。
- **状态码**：响应报文独有的，提供了请求是否成功的明确指示，而请求报文没有类似的状态指示。

3.3 task3

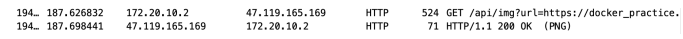

图 4: GET

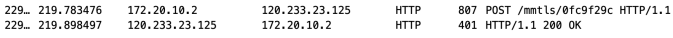

图 5: POST

3.3.1 GET 与 POST 请求报文的参数位置与安全性

在 HTTP 请求中, GET 和 POST 方法在处理参数上存在显著差异。GET 请求将参数直接附在 URL 之后, 如 "GET /api/img?url=https://moveit.ros.org/assets/images/roadmap.png HTTP/1.1", 其中参数 "url=https://-moveit.ros.org/assets/images/roadmap.png" 直观可见于地址栏，这导致 GET 请求的数据不安全，易被缓存和记录。相反, POST 请求的参数隐藏在请求正文内, 如 "POST /mmtls/0fc9f29c HTTP/1.1" 示例, 其 "Content-Length: 519" 表明数据在正文且不显示于 URL，提高了数据传输的隐私性和安全性。

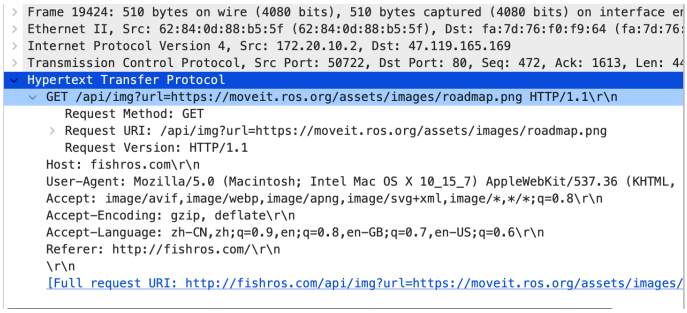

图 6: GET 正文

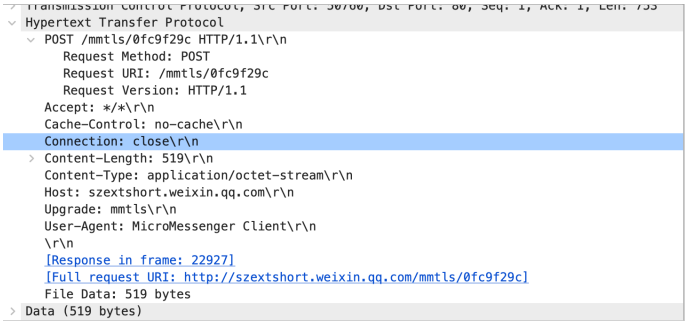

图 7: POST 正文

3.3.2 数据长度限制与适用场景

GET 方法受限于 URL 长度，因此不适合传输大量数据，适用于查询参数简单的情况。POST 方法则无此限制，适合文件上传、表单大量数据提交等场景，确保了大数据量传输的可能性。

3.3.3 请求报文格式的对比

GET 请求通常不包含请求正文，信息集中于请求行和头部，简化了报文结构。POST 请求则必须有正文，且通过“Content-Length”字段指定正文大小，确保服务器正确处理数据。

3.3.4 响应报文的侧重点与头部差异

响应报文根据请求方法的不同，其内容和头部信息各有侧重。GET 响应通常直接返回请求的资源，如图片或 HTML 页面，其“Content-Type”指示资源类型。POST 响应则侧重于操作反馈，可能包括成功状态或错误信息，头部信息可能涉及数据处理的状态指示，但这些差异取决于应用逻辑和服务器配置，不是固定不变的。

3.4 task4

POP 数据包组成分析

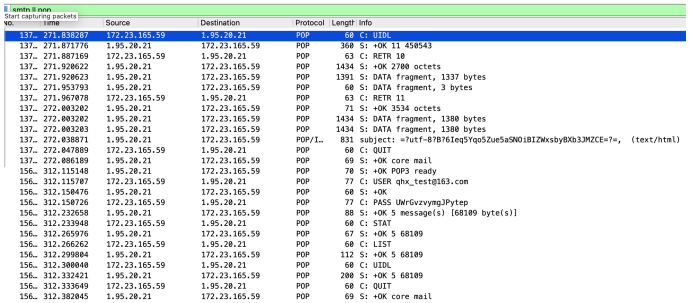

图 8: SMTP||POP

3.4.1 链路层

实际网络通信中，链路层负责将数据包封装成帧，通过源 MAC 地址和目的 MAC 地址在本地网络中传输数据，是数据包在物理网络中传输的基础。

3.4.2 网络层（IPv4）

从图中源 IP 地址（如 172.23.165.59）和目的 IP 地址（如 1.95.20.21）可知，网络层为数据包提供了在网络中的逻辑寻址功能。IPv4 首部包含版本、首部长度、区分服务字段、总长度、标识、标志、片偏移、生存时间、协议、首部校验和、源 IP 地址和目的 IP 地址等信息，用于确定数据包的传输路径，确保数据包能从源主机到达目的主机。

3.4.3 传输层（TCP）

POP3 基于 TCP 协议，传输层负责在源主机和目的主机的应用程序之间提供端到端的通信。TCP 通过序列号、确认号、窗口大小、校验和、标志位（如 SYN、ACK、FIN 等）来保障数据的可靠传输，建立和维护连接，处理流量控制和错误恢复等。

3.4.4 应用层（POP3）

- **连接与欢迎**：服务器发送 “+OK POP3 ready”（如 156... 312.115148 这一行），表示服务器已准备好接受客户端连接，标志着 POP3 会话开始。
- **用户认证**：客户端发送 “USER qhx_test@163.com”（156... 312.115707）提供用户名，服务器回应 “+OK”（156... 312.150476）；接着客户端发送 “PASS UWrGvzvyngJPyptep”（156... 312.150726）提交密码，服务器验证通过后回复 “+OK 5 message(s) [68109 byte(s)]”（156... 312.232658），告知客户端邮箱中的邮件数量和总字节数。
- **邮件信息获取**：
  - 客户端发送 “STAT” 命令（156... 312.233948）获取邮件统计信息，服务器回应 “+OK 5 68109”（156... 312.265976），分别表示邮件数量和总字节数。

  - 客户端发送“LIST”命令（156... 312.266262）获取邮件列表，服务器回复“+OK 5 68109”（156... 312.299804）。
  - 客户端发送“UIDL”命令（137... 271.838287 以及 156... 312.300040）获取邮件唯一标识符列表，服务器进行相应回复（如 137... 271.871776 和 156... 312.332421）。

- **邮件内容获取**：客户端发送“RETR 10”（137... 271.887169）和“RETR 11”（137... 271.967078）命令分别获取编号为 10 和 11 的邮件内容，服务器先回复邮件大小（如“+OK 2700 octets”（137... 271.920622）和“+OK 3534 octets”（137... 272.003202）），然后分片段发送邮件数据（如“DATA fragment, 1337 bytes”（137... 271.920622）等）。

- **连接关闭**：客户端发送“QUIT”命令（137... 272.047889 和 156... 312.333649）请求断开连接，服务器回应“+OK core mail”（137... 272.086189 和 156... 312.382045），结束 POP3 会话。

SMTP 数据包组成分析

3.4.5 链路层

同样负责将数据包封装成帧，利用 MAC 地址在本地网络传输数据。

3.4.6 网络层（IPv4）

通过源 IP 和目的 IP 地址为数据包在网络中寻址，与 POP3 数据包的网络层功能一致。

3.4.7 传输层（TCP）

SMTP 基于 TCP 协议，通过 TCP 的序列号、确认号、标志位等机制保证邮件传输过程中数据的可靠传输，建立和维护客户端与服务器之间的连接。

3.4.8 应用层（SMTP）

- **连接建立**：客户端与服务器通过三次握手建立 TCP 连接，之后服务器以“220”等状态码响应客户端请求，客户端发送“EHLO”命令标识身份。

- **认证**：客户端发起“AUTH LOGIN”等认证请求，发送经 Base64 编码的用户名和密码，服务器进行验证并返回相应状态码。

- **邮件参数设置**：客户端使用“MAIL FROM”指定发件人，“RCPT TO”指定收件人，服务器确认操作是否成功并回复状态码。

- **邮件内容传输**：客户端发送“DATA”命令，服务器告知结束标识，客户端传输邮件内容，服务器确认接收并排队。

- **连接关闭**：客户端发送“QUIT”命令，服务器回应“221 Bye”，双方通过四次挥手关闭连接。

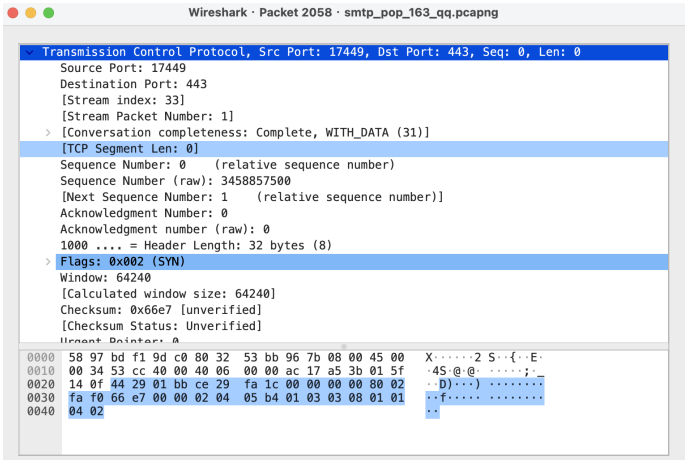

图 9: SMTP||POP

3.5 task5

SMTP 协议报文交换分析

3.5.1 TCP 连接建立

TCP 连接通过三次握手完成。从 Wireshark 抓包数据（以序号 2058 的数据包为例）可知，客户端（源 IP 172.23.165.59）使用源端口 17449 向目的端口 443（目的 IP 1.95.20.15）发送 TCP 包。此数据包标志位 Flags 为 0x002（SYN），表示客户端发起连接请求，序号 Sequence Number 为 0（相对值），此时客户端进入 SYN-SENT 状态。

服务器收到请求后，回应 SYN-ACK 包（如序号 2060 的数据包），Seq 为 0，Ack（确认号）为 1（确认收到客户端 SYN 包并将序列号加 1），服务器进入 SYN-RECV 状态。

客户端收到 SYN-ACK 包后，发送 ACK 包（如序号 2061 的数据包），Seq 为 1，Ack 为 1，表示客户端确认收到服务器的 SYN-ACK 包，双方建立连接，进入 ESTABLISHED 状态。

3.5.2 SMTP 命令交互

TCP 连接建立后，SMTP 客户端和服务器进行命令与响应交互：

- **EHLO 命令**：客户端发送 “EHLO LAPTOP-GIH06R83”（对应实验报告中包 2331），用于向服务器标识自身主机名。服务器回应 “250-mail | PIPELINING | AUTH LOGIN PLAIN XOAUTH2 | AUTH=LOGIN PLAIN XOAUTH2”（对应包 2334），表明服务器就绪，并告知支持的功能与认证方式。这类似于客户端发送 “EHLO LAPTOP - 4HJ99PRJ”，服务器返回支持的功能选项（如最大邮件大小、是否支持 STARTTLS 加密等）的情况。
- **AUTH XOAUTH2 命令**：客户端发起 “AUTH LOGIN” 认证请求（对应包 2335），随后发送经 Base64 编码的用户名 “User: MTg2MjUxNzEzODY@MTYzLmNvbQ==”（对应包 2340）和密码 “Pass: QlFxVmd-wZU15VlRoTm5tUW==”（对应包 2343）。服务器最终以 “235 Authentication successful”（对应包 2347）

确认认证成功。类似地，客户端尝试使用 OAuth 2.0 认证机制登录，发送 “AUTH XOAUTH2...” 命令，服务器回复 Base64 编码信息指示认证状态，同样是客户端进行认证操作，服务器回应认证结果。

3.5.3 TCP 连接关闭

通信结束时，双方按 TCP 四次挥手断开连接：

- 服务器发起关闭请求，向客户端发送 FIN, ACK 包（如序号 1616 的数据包），序列号 Seq=4425，确认号 Ack=5181，表示服务器完成数据发送，请求关闭连接，此时服务器进入 FIN-WAIT-1 状态。
- 客户端确认请求，收到服务器的 FIN 包后，回复 ACK 包（如序号 1685 的数据包），序列号 Seq=4681，确认号 Ack=19706，客户端进入 CLOSE-WAIT 状态，服务器收到该 ACK 后进入 FIN-WAIT-2 状态。
- 客户端发起关闭请求，客户端完成自身数据发送后，向服务器发送 FIN, ACK 包（如序号 1687 的数据包），序列号 Seq=20328，确认号 Ack=4789，客户端进入 LAST-ACK 状态。
- 服务器确认关闭，服务器收到客户端的 FIN 包后，回复 ACK 包（如序号 1684 的数据包），序列号 Seq=4681，确认号 Ack=19694，服务器进入 CLOSED 状态，客户端收到该 ACK 后也进入 CLOSED 状态，连接完全关闭。

4 总结

本次实验围绕 HTTP、HTTPS、SMTP 和 POP3 协议展开，借助 Wireshark 工具进行分析，让我收获了丰富的知识与实践经验。

在实验过程中，我深入理解了 HTTP 和 HTTPS 协议的原理及运行机制。HTTP 基于 TCP 协议，无状态的特点使其高效但需借助 Cookie 等技术处理状态问题。HTTPS 通过 SSL/TLS 协议加密数据，解决了 HTTP 存在的安全风险，保障敏感数据传输安全。分析 HTTP 报文时，我熟悉了请求报文和响应报文的结构，各部分协同完成客户端与服务器的交互。

GET 和 POST 方法是 HTTP 协议的重要组成部分。通过抓包对比，我掌握了它们在请求和响应报文上的差异。GET 用于获取资源，参数在 URL 中，安全性较低且数据量受限；POST 用于提交数据，数据在请求正文，更适合大量数据传输和对安全性要求高的场景。

HTTPS 协议分析中，配置相关参数后成功抓取数据包，深入研究 TLS 握手过程，理解其在建立安全连接、保障数据加密传输方面的关键作用。

SMTP 和 POP3 协议的分析实验，使我熟悉了邮件传输和接收的工作机制。通过配置邮箱账户收发邮件，抓取数据包，直观看到协议在客户端和服务器间的交互，对邮件系统原理有了更清晰的认识。

实验中遇到抓包不准确和协议分析困难等问题，经过重新配置、查阅资料和请教老师得以解决，这提升了我的实践和解决问题的能力。

此次实验激发了我对网络协议的学习兴趣。未来，我将深入学习 HTTP/3、QUIC 等网络协议的高级特性，加强网络安全研究，为开发更优质的网络应用程序努力，注重理论与实践结合，提升网络技术综合素养。