实验一：应用层协议消息的捕获和解析

姓名	学号	班级	专业
王何佳	2023211603	2023211804	网络空间安全

一、实验内容

1. 使用 Wireshark 软件捕获 HTTP 消息，分析其消息头，理解 HTTP 的通信原理。
2. 使用 Wireshark 软件捕获一次从客户端发送 Email 的过程，分析 SMTP 消息，理解 Email 系统中发送邮件的通信原理。
3. 使用 Telnet 软件访问 Email 服务器，输入 SMTP 命令与 Email 服务器交互，理解 SMTP 的通信过程和 Base64 编码的概念。

二、实验环境

Windows10虚拟机：安装了Wireshark、Foxmail、Firefox

三、实验步骤

1. HTTP协议实验

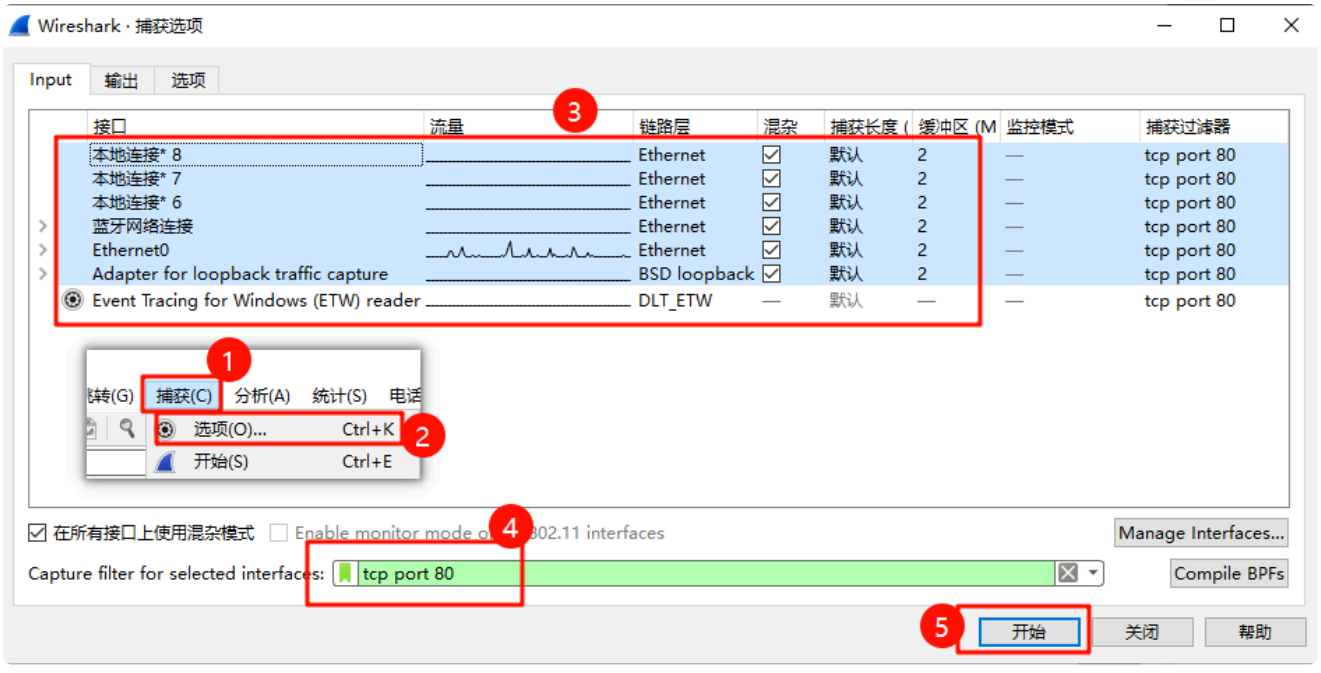
1.1. 安装wireshark

1.2. 确保计算机已连接到网络

1.3. 清除cookie数据（访问记录）

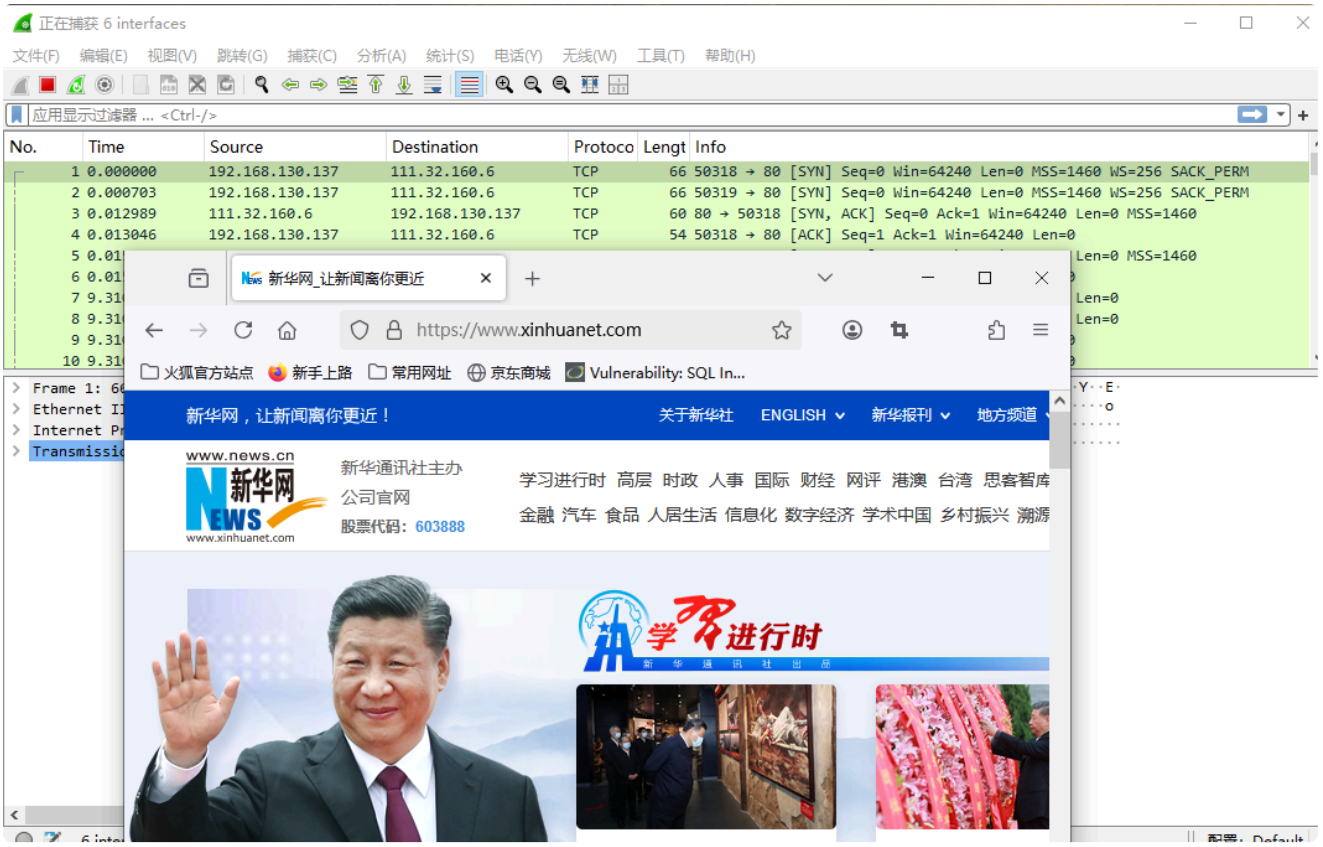
1.4. 运行 wireshark 并设置捕获条件

捕获->选项，选中所有活跃的网卡，设置捕获协议和端口号`tcp port 80`，开始抓包。

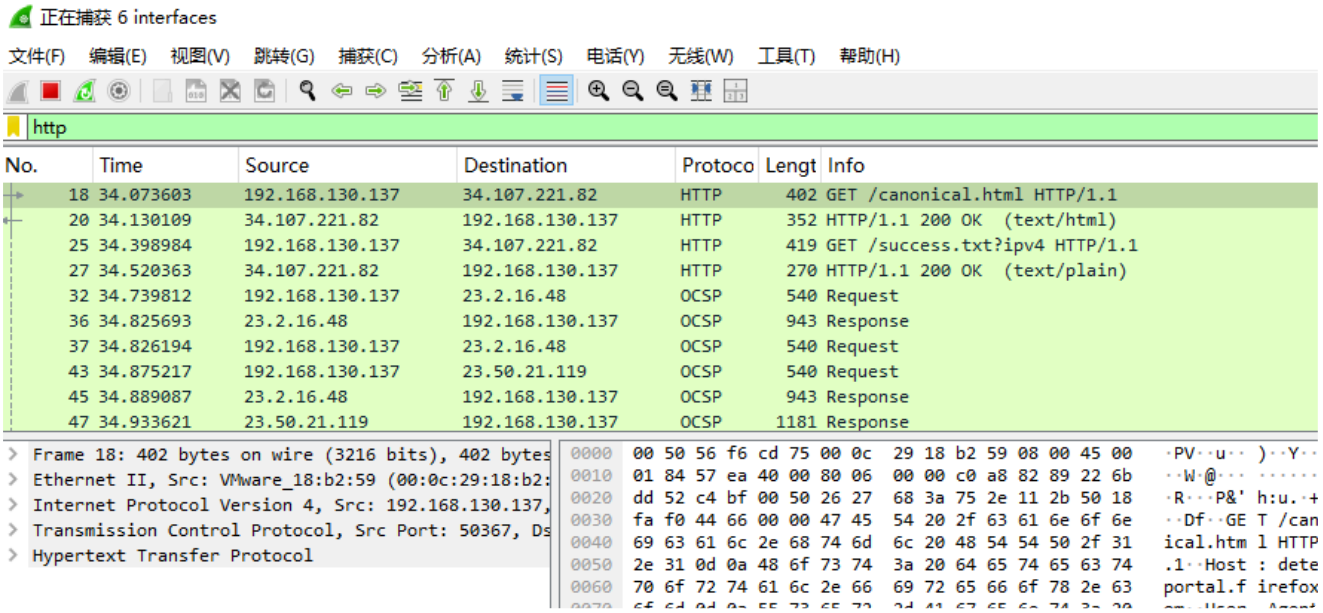


1.5. 抓包

访问`www.xinhuanet.com`，网页全部显示后停止捕获。

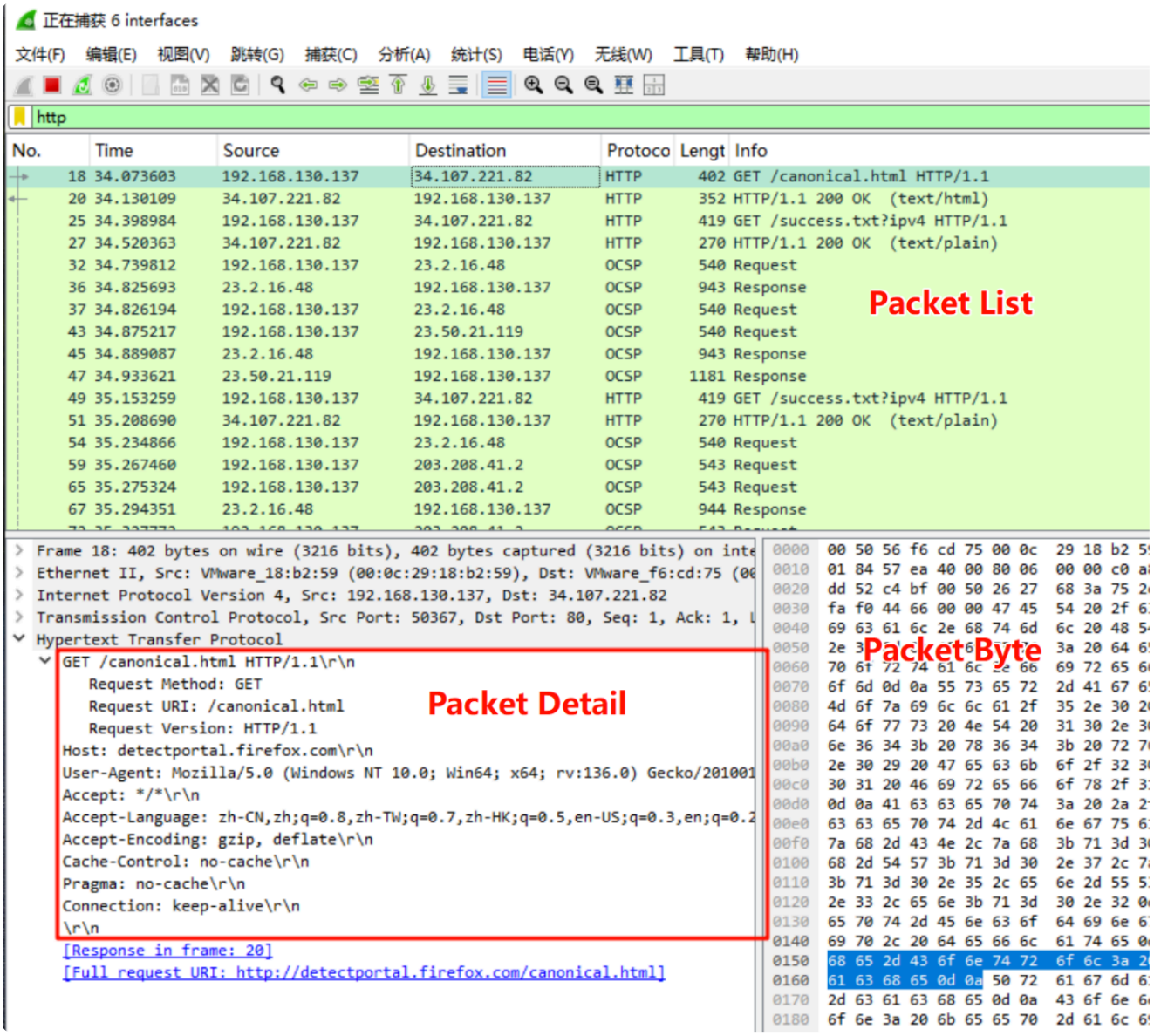


筛选http



1.6. 消息头分析

1.6.1. 请求消息分析



图中消息头部分内容如下：

```
GET /canonical.html HTTP/1.1\r\n
Host: detectportal.firefox.com\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2\r
Connection: keep-alive\r\n
```

分析：

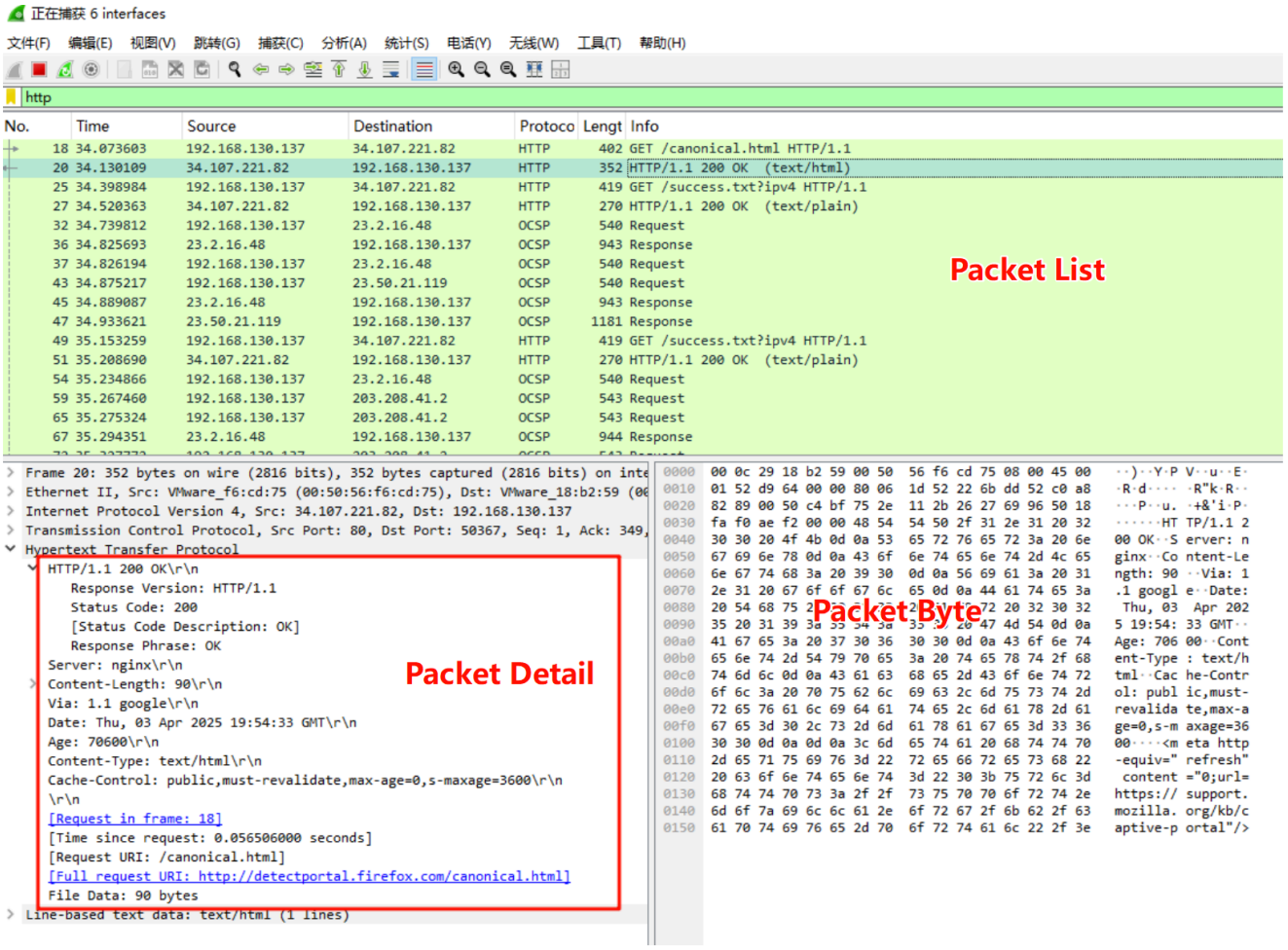
- 请求方法：`GET`
- 协议版本：`HTTP/1.1`
- Host：`detectportal.firefox.com`
- 客户端信息：`Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0`
  - Windows 10系统，64位架构
  - 使用Firefox浏览器，版本号136.0
- 内容协商：

    - 客户端接收任何类型的响应内容
    - 语言优先级：简体中文>中文>台湾繁体>香港繁体>美式英语>英语
- 连接管理：`keep-alive`，客户端希望保持TCP连接（HTTP/1.1默认）

HTTP请求消息头字段列表总结

字段名称	功能	现有值及含义
Request Method	定义客户端请求的操作类型。	`GET`：请求获取指定资源。
Request URI	指定请求的资源路径。	`/canonical.html`：目标资源路径。
Request Version	使用的HTTP协议版本。	`HTTP/1.1`：支持持久连接和分块传输。
Host	指定目标服务器的主机名和端口。	`detectportal.firefox.com`：目标主机地址。
User-Agent	标识客户端类型、操作系统和浏览器版本。	`Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/201001`：Windows 10系统，Firefox 136.0。
Accept	声明客户端可接受的响应内容类型。	`*/*`：接受任意类型的内容。
Accept-Language	声明客户端优先接受的语言。	`zh-CN,zh;q=0.8...`：优先级为简体中文 > 中文 > 台湾繁体 > 香港繁体 > 英语。
Accept-Encoding	声明客户端支持的压缩算法。	`gzip, deflate`：支持gzip或deflate压缩。
Cache-Control	控制缓存行为。	`no-cache`：要求服务器提供最新内容，不使用缓存。
Pragma	旧版HTTP字段，用于向后兼容，功能类似`Cache-Control`。	`no-cache`：禁用缓存。
Connection	控制TCP连接是否保持活跃。	`keep-alive`：保持连接复用。

1.6.2. 响应消息分析



图中消息头部分内容如下：

```
HTTP/1.1 200 OK
Server:nginx
Content-Length:90
Content-Type:text/html
```

分析：

- 协议版本：`HTTP/1.1`
- 状态码：`200 OK`，表示资源已正常返回
- 服务器使用`nginx`，响应经过Google代理服务器
- 内容类型：`text/html`，响应体长度为90字节

HTTP响应消息头字段总结

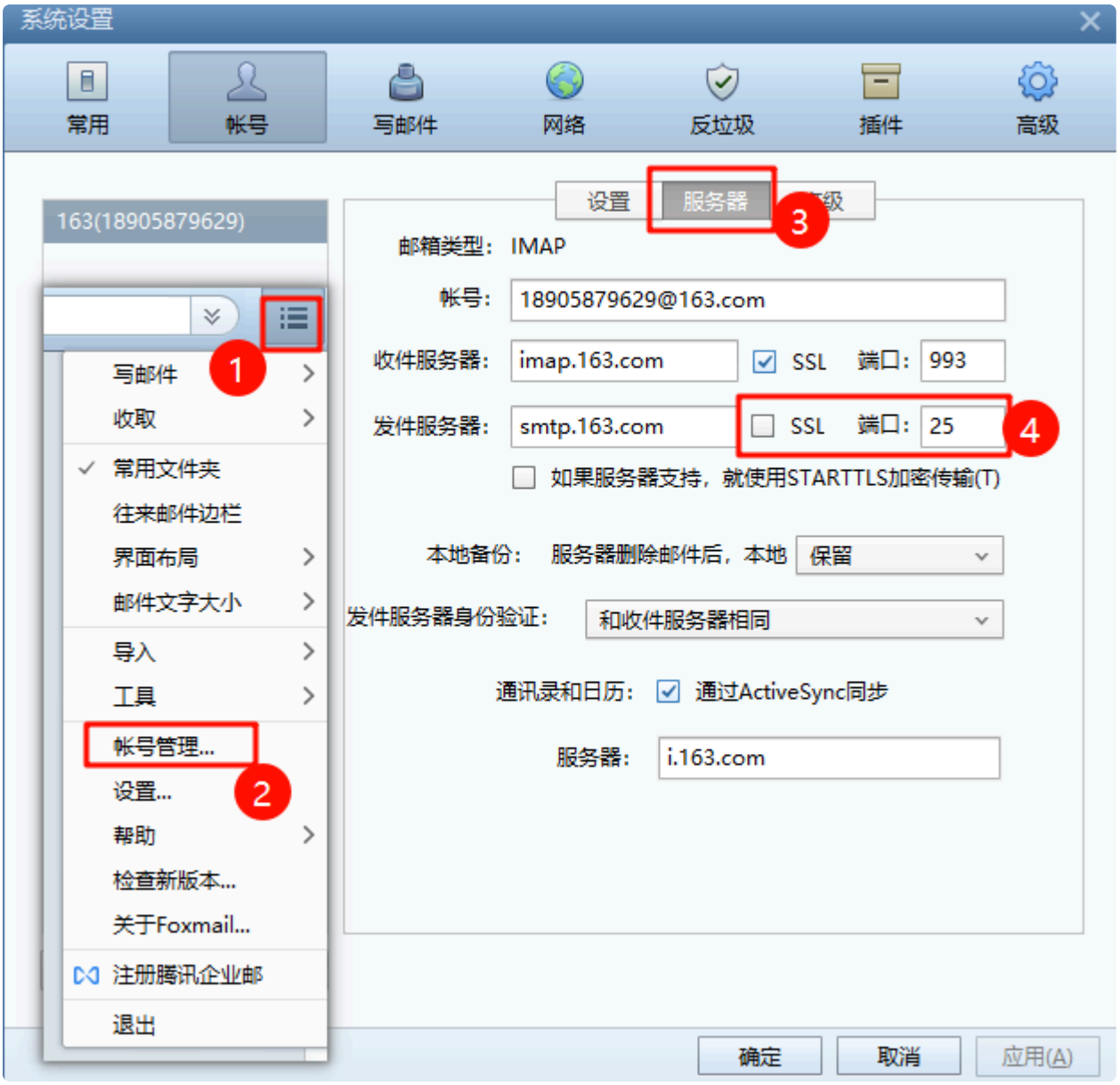
字段名称	功能	现有值及含义
Response Version	使用的HTTP协议版本。	`HTTP/1.1`：支持持久连接和分块传输。
Status Code	表示请求处理结果的三位数字代码。	`200 OK`：请求成功，资源已返回。

字段名称	功能	现有值及含义
Server	标识服务器软件类型及版本。	`nginx`：服务器使用Nginx处理请求。
Content-Length	声明响应体的字节大小。	`90`：响应体长度为90字节。
Via	显示请求经过的代理或网关信息。	`1.1 google`：响应经过Google代理服务器（如CDN）。
Date	响应生成的日期和时间。	时间为`Thu, 03 Apr 2025 19:54:33 GMT`
Age	表示响应在缓存中已存储的时间（秒）。	`70600`：响应已缓存约19.6小时。
Content-Type	声明响应体的媒体类型。	`text/html`：内容为HTML格式。
Cache-Control	控制客户端和中间缓存的行为。	`public,must-revalidate,max-age=0,s-maxage=3600`：公共缓存，客户端立即失效，CDN缓存1小时。

2. SMTP协议实验

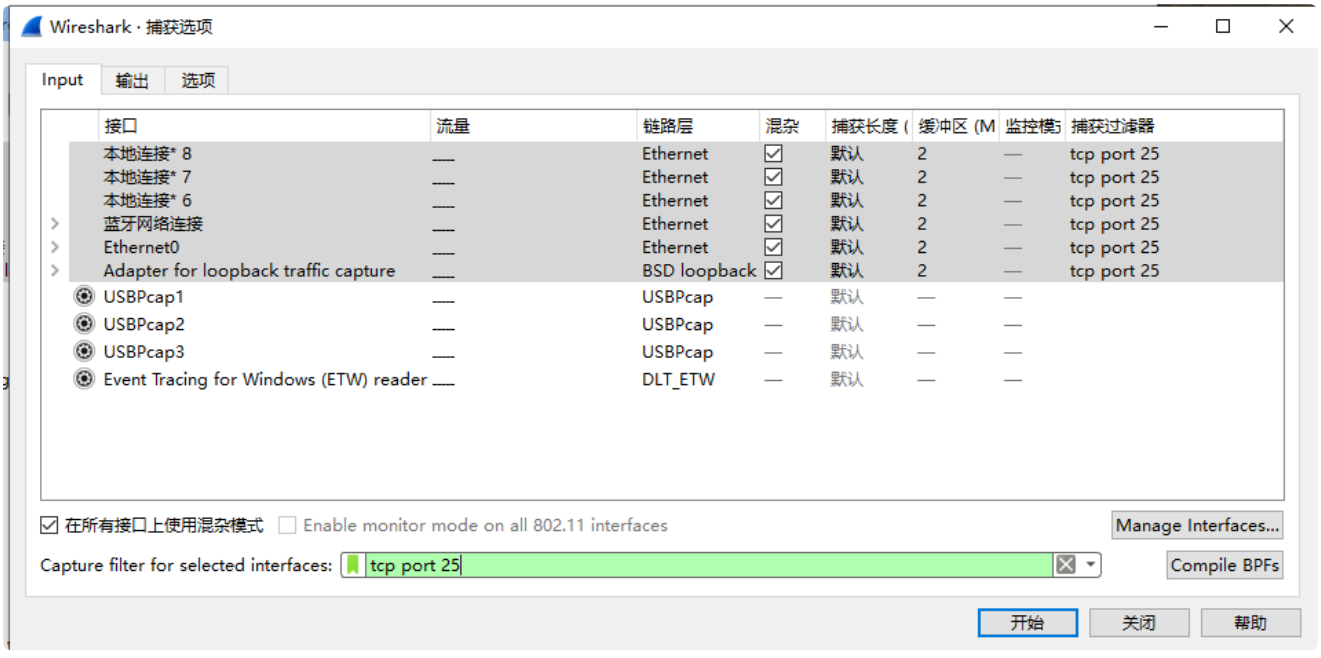
2.1. 下载Foxmail

配置：发件服务器不勾选SSL，端口为25。



2.2. 运行wireshark并设置捕获条件

捕获->选项，选中所有活跃的网卡，设置捕获协议和端口号`tcp port 25`，开始抓包。

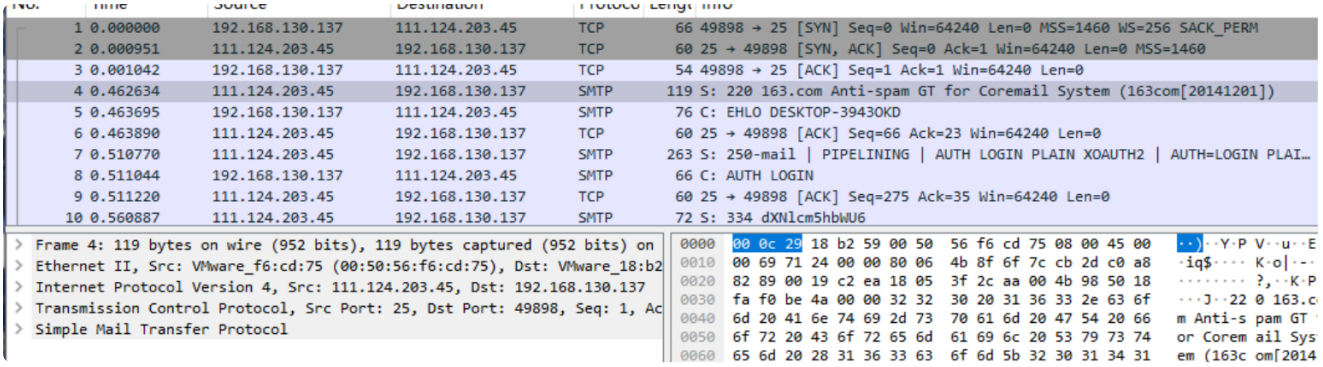


2.3. 使用Foxmail发送邮件

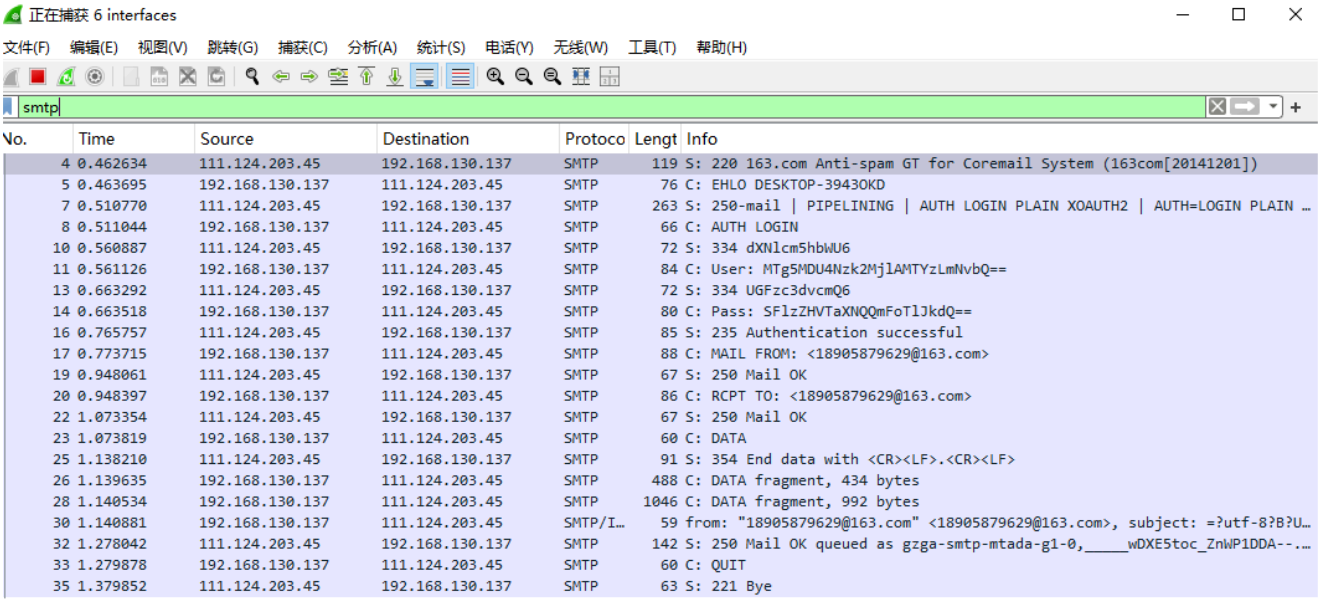
2.4. 抓包

发送成功后停止截获

可以看到序号1-3为TCP的三次握手建立连接



筛选SMTP



2.5. SMTP协议的分析

Protoco	Lengt	Info
SMTP	119	S: 220 163.com Anti-spam GT for Coremail System (163com[20141201])
SMTP	76	C: EHLO DESKTOP-3943OKD
SMTP	263	S: 250-mail | PIPELINING | AUTH LOGIN PLAIN XOAUTH2 | AUTH=LOGIN PLAIN
SMTP	66	C: AUTH LOGIN
SMTP	72	S: 334 dXNlcm5hbWU6
SMTP	84	C: User: MTg5MDU4Nzk2MjlAMTYzLmNvbQ==
SMTP	72	S: 334 UGFzc3dvcmQ6
SMTP	80	C: Pass: SFlzZHVTaXNQQmFoTlJkdQ==
SMTP	85	S: 235 Authentication successful
SMTP	88	C: MAIL FROM: <18905879629@163.com>
SMTP	67	S: 250 Mail OK
SMTP	86	C: RCPT TO: <18905879629@163.com>
SMTP	67	S: 250 Mail OK
SMTP	60	C: DATA
SMTP	91	S: 354 End data with <CR><LF>.<CR><LF>
SMTP	488	C: DATA fragment, 434 bytes
SMTP	1046	C: DATA fragment, 992 bytes
SMTP/I…	59	from: "18905879629@163.com" <18905879629@163.com>, subject: =?utf-8?B?
SMTP	142	S: 250 Mail OK queued as gzga-smtp-mtada-g1-0,_____wDXE5toc_ZnWP1DDA--
SMTP	60	C: QUIT
SMTP	63	S: 221 Bye

对图中base64编码的字符进行解码：

smtp.py

```
1  import base64
2  print(base64.b64decode(b'dXNlcm5hbWU6'))
3  print(base64.b64decode(b'UGFzc3dvcmQ6'))
4  print(base64.b64decode(b'MTg5MDU4Nzk2MjlAMTYzLmNvbQ=='))
5  print(base64.b64decode(b'SFlzZHVTaXNQQmFoTlJkdQ==`'))
6
```

问题 输出 调试控制台 终端 端口 筛选器 Coc

```
[Running] python -u "d:\VSCode\Py_Project\smtp.py"
b'username:'
b'Password:'
b'18905879629@163.com'
b'HYsduSisPBahNRdu'
```

连接建立

1. **SMTP连接建立**：
   `220 163.com Anti-spam GT for Coremail System (163com[20141201])`
2. **客户端发送EHLO命令**：
   `EHLO DESKTOP-3943OKD`
3. **服务器响应EHLO命令**：
   `250`表示连接成功

身份认证

1. **客户端请求使用LOGIN方式进行身份验证**:
   `AUTH LOGIN`
2. **服务器请求用户名**:
   `334 dXNlcm5hbWU6`
   解码后为`username:`
3. **客户端发送用户名**:
   `User: MTg5MDU4Nzk2MjlAMTYzLmNvbQ==`
   解码后为`18905879629@163.com`
4. **服务器请求密码**:
   `334 UGFzc3dvcmQ6`
   解码后为`Password:`
5. **客户端发送密码**: （此处为授权码）
   `Pass: SFlzZHVTaXNQQmFoTlJkdQ==`
   解码后为`HYsduSisPBahNRdu`
6. **服务器确认身份验证成功**:
   `235 Authentication successful`

邮件发送

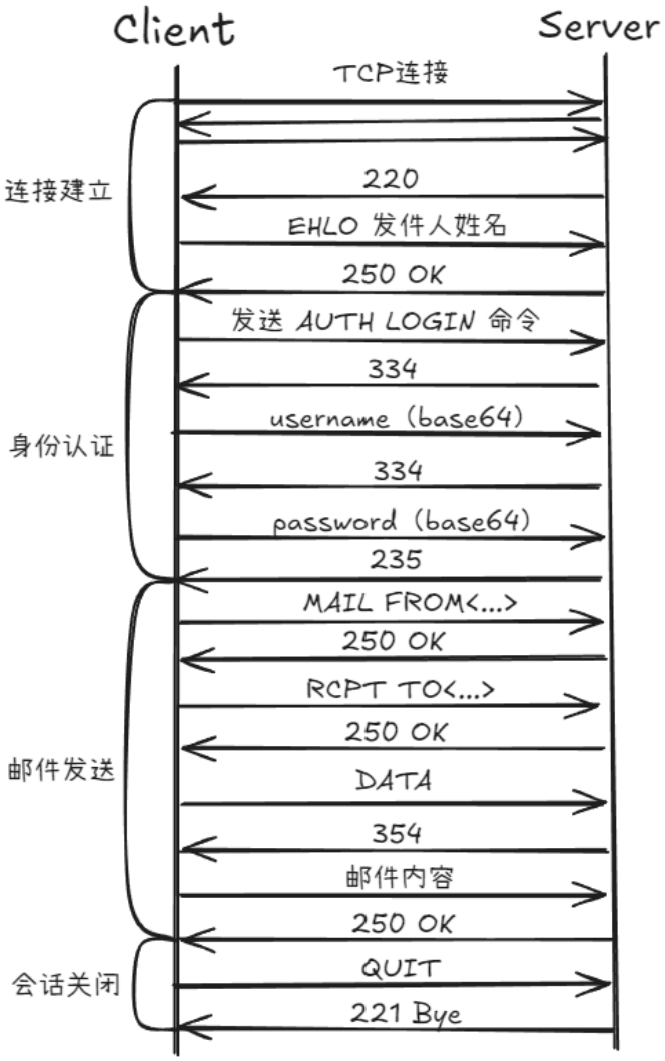
1. **客户端指定发件人邮箱地址**:
   `MAIL FROM: <18905879629@163.com>`
2. **服务器确认发件人地址有效**:
   `250 Mail OK`
3. **客户端指定收件人邮箱地址**:
   `RCPT TO: <18905879629@163.com>`
4. **服务器确认收件人地址有效**:
   `250 Mail OK`
5. **客户端请求发送邮件内容**:
   `DATA`
6. **服务器确认可以发送邮件内容，并提示以`<CR><LF>.<CR><LF>`结束邮件内容**:
   `354 End data with <CR><LF>.<CR><LF>`
7. **客户端发送邮件内容，分两个片段，总大小为434 + 992 = 1426字节**:
   `DATA fragment, 434 bytes`
   `DATA fragment, 992 bytes`
8. **服务器确认邮件已成功接收并排队发送**:
   `250 Mail OK queued as ggzga-smtp-mtada-g1-0`

会话关闭

1. **客户端请求关闭SMTP会话**:
   `QUIT`

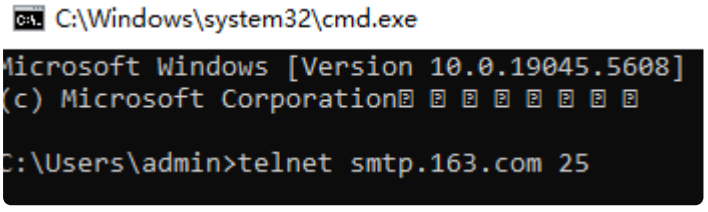
2. **服务器确认会话已关闭**：
   `221 Bye`

通信过程



3. Telnet交互实验

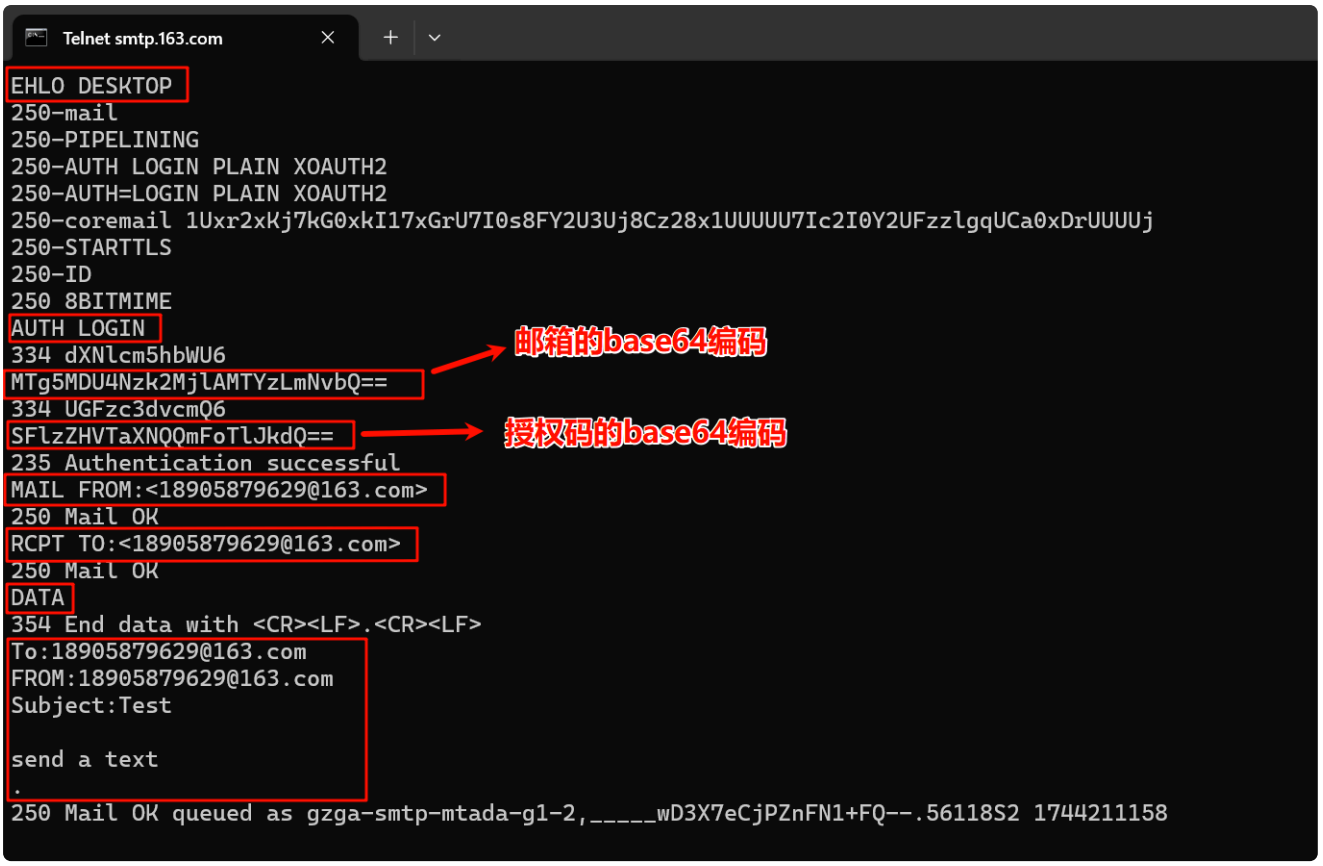
打开cmd，输入`telnet smtp.163.com 25`。



将邮箱号和授权码转为base64编码：

依次输入

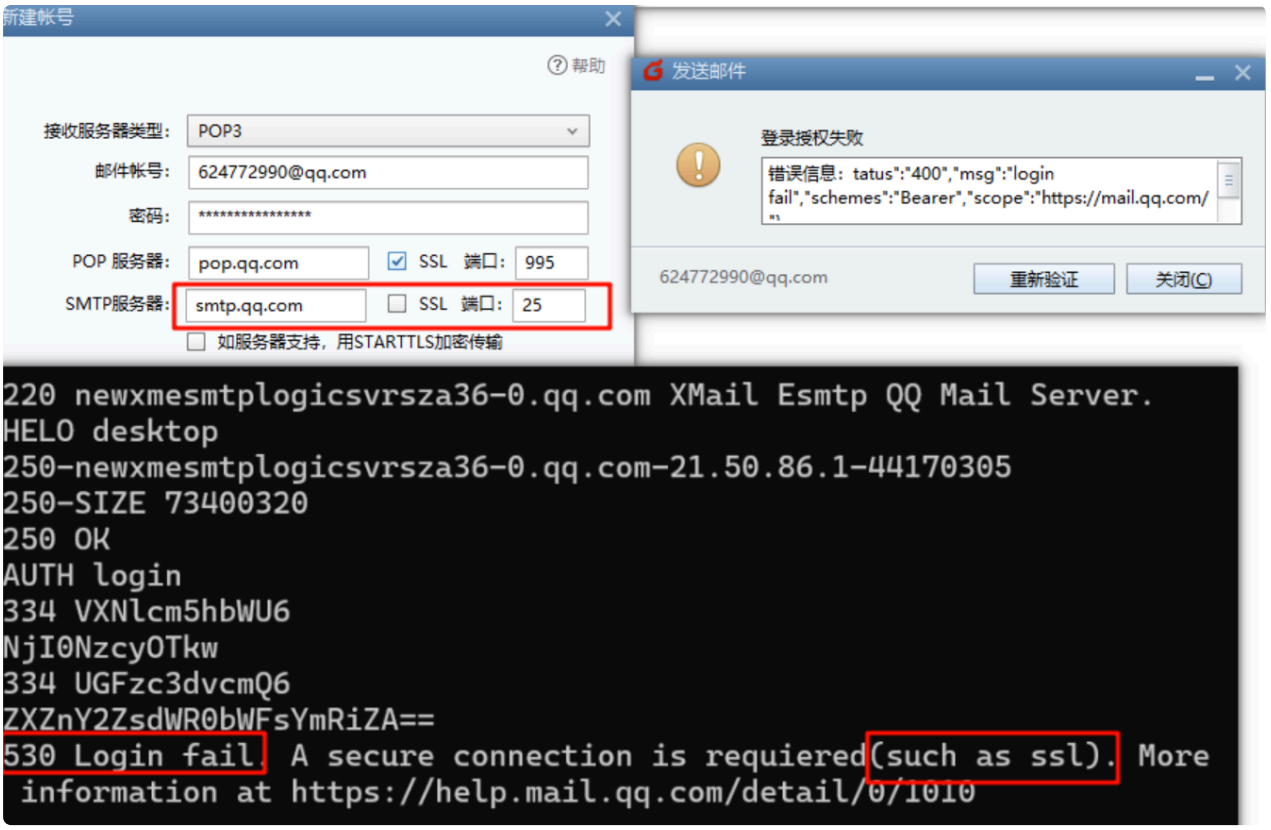
```
EHLO DESKTOP
AUTH LOGIN
MTg5MDU4Nzk2MjlAMTYzLmNvbQ==      //18905879629@163.com的base64编码
SFlzZHVTaXNQQmFoTlJkdQ==          //HYsduSisPBahNRdu的base64编码
MAIL FROM:<18905879629@163.com>   //发件人
RCPT TO:<18905879629@163.com>     //收件人
DATA
To:18905879629@163.com
FROM:18905879629@163.com
Subject:Test

send a text

.
QUIT
```



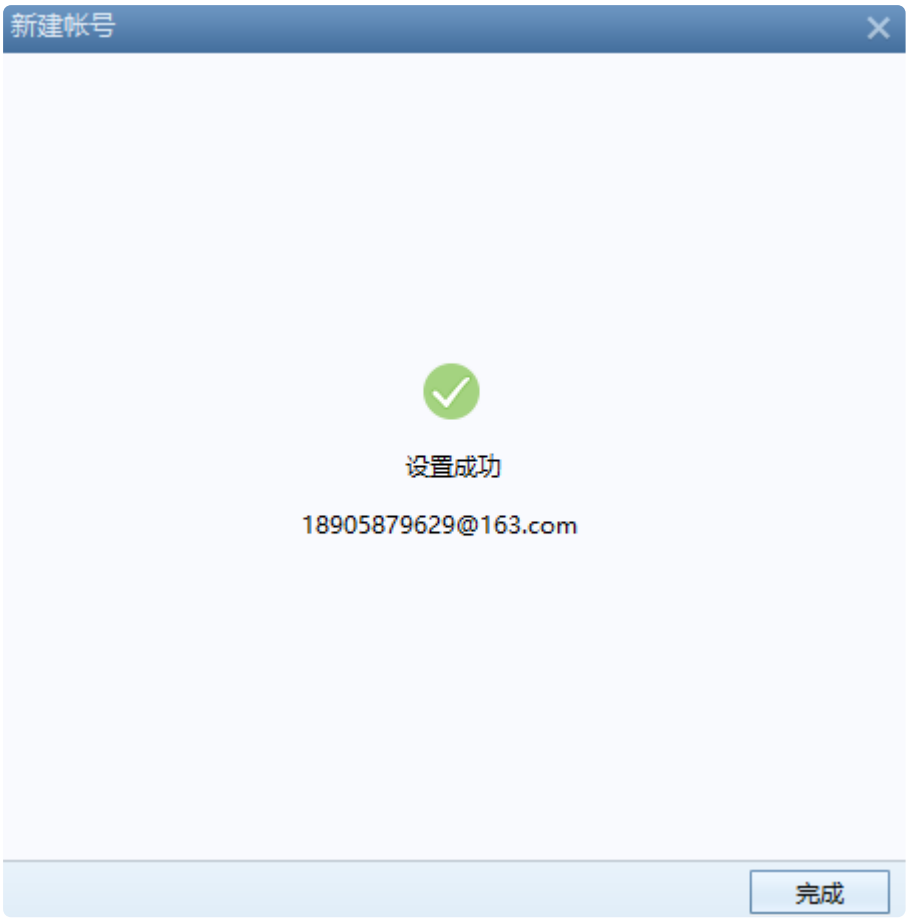
成功收到邮件

四、问题与解决

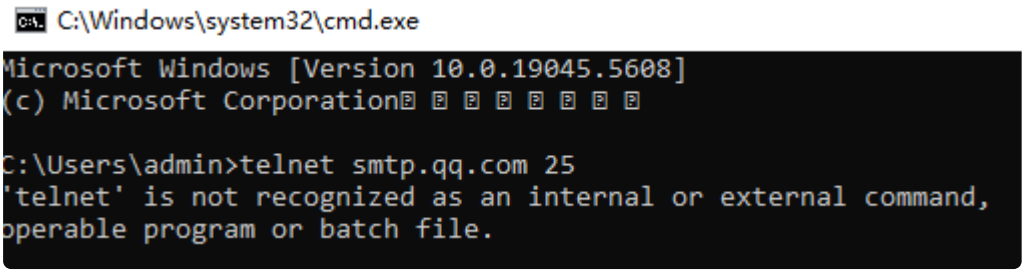
1. 问题：QQ邮箱强制要求SSL加密，不勾选无法成功发送邮件，也无法用telnet连接。
   解决：换用163邮箱

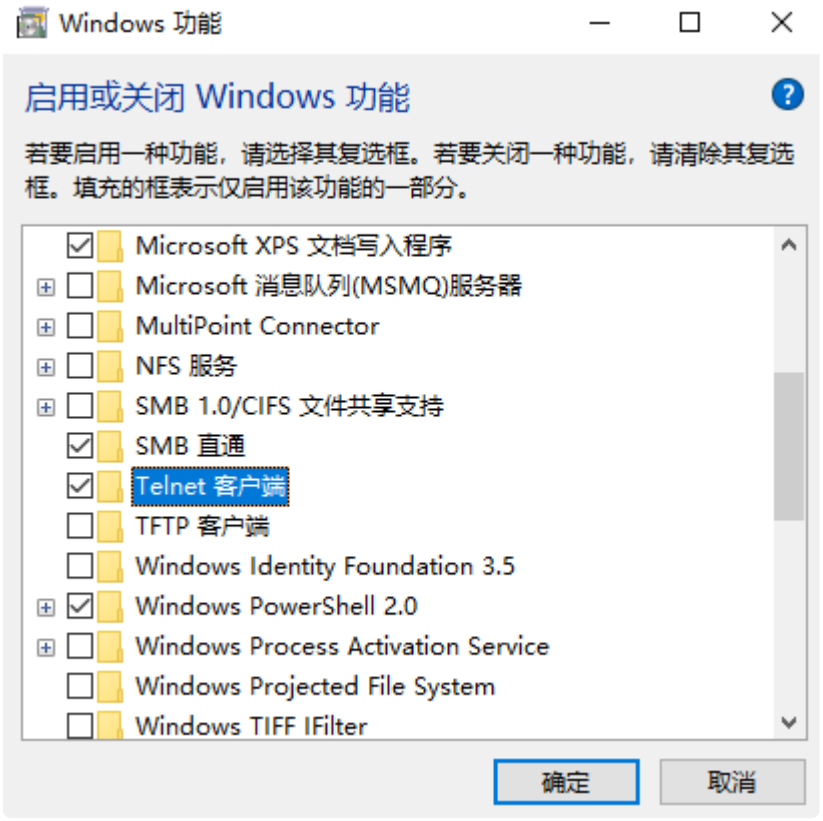


2. 问题：Foxmail 新建账号登不进去。
   解决：为邮箱开启IMAP/SMTP服务，并在“密码”框填授权码，而不是邮箱密码。



3. 问题：Telnet无法打开。
   解决：“启用或关闭Windows功能”中，勾选Telnet客户端





五、心得与体会

通过本次实验，我对应用层协议的工作原理有了更深入的理解，并掌握了使用工具进行协议分析与交互的实践技能。

通过Wireshark捕获HTTP请求与响应消息，我直观地观察到浏览器与服务器之间的通信过程。例如，请求头中的`User-Agent`和`Accept-Language`字段让我认识到客户端如何声明自身能力与偏好；响应头中的`Content-Type`和`Cache-Control`则揭示了服务器如何控制资源类型与缓存策略。此外，通过分析`keep-alive`机制，我理解了HTTP/1.1相较于旧版本在连接复用上的优化。这些实践让我对Web通信的细节有了更具体的认识。

在SMTP实验中，我通过Foxmail发送邮件并抓包分析，完整地观察到邮件从身份认证到内容传输的全过程。例如，`AUTH LOGIN`命令触发的Base64编码身份验证让我意识到协议设计中安全性实现的重要性；`DATA`命令的交互则展示了邮件内容的分段传输机制。同时，通过手动解码Base64字段，我进一步巩固了编码原理的实际应用场景。

通过Telnet直接与SMTP服务器交互，我亲身体验了协议命令的逐条执行过程。手动输入`EHLO`、`MAIL FROM`、`RCPT TO`等命令，并观察服务器的响应，让我对SMTP的“请求-响应”模式有了更直观的理解。此外，Base64编码的实际操作也加深了我对编码技术必要性的认识。

Wireshark的过滤功能（如`tcp port 80`或`tcp port 25`）帮助我快速定位目标协议数据包，而Foxmail和Telnet的配合使用则让我熟悉了不同工具在协议分析中的互补作用。这些技能为后续网络协议的学习与实践打下了坚实基础。

本次实验不仅让我掌握了Wireshark、Telnet等工具的使用技巧，更重要的是培养了我从协议层面对网络通信进行分析的能力。这些经验对我后续学习网络安全、协议逆向等课程具有重要意义，同时也让我认识到协议设计中的安全性与效率权衡，为未来参与实际项目积累了宝贵的实践经验。