

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

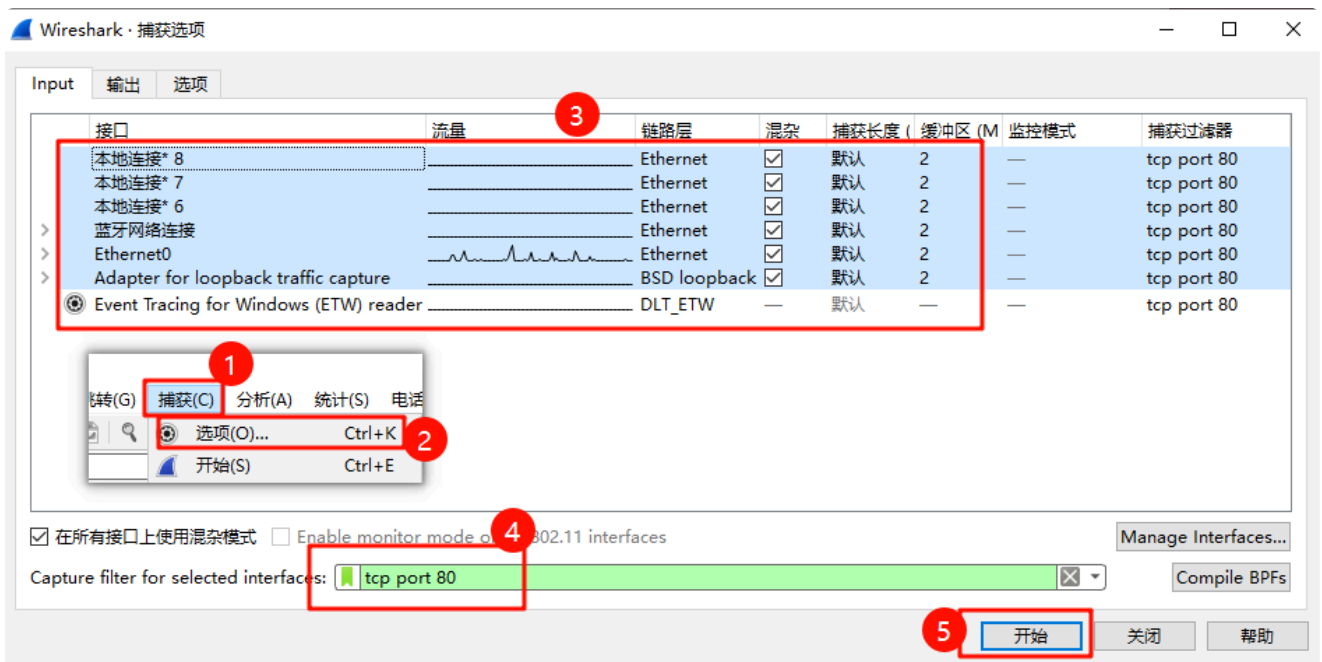


1.3. 清除cookie数据（访问记录）



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。

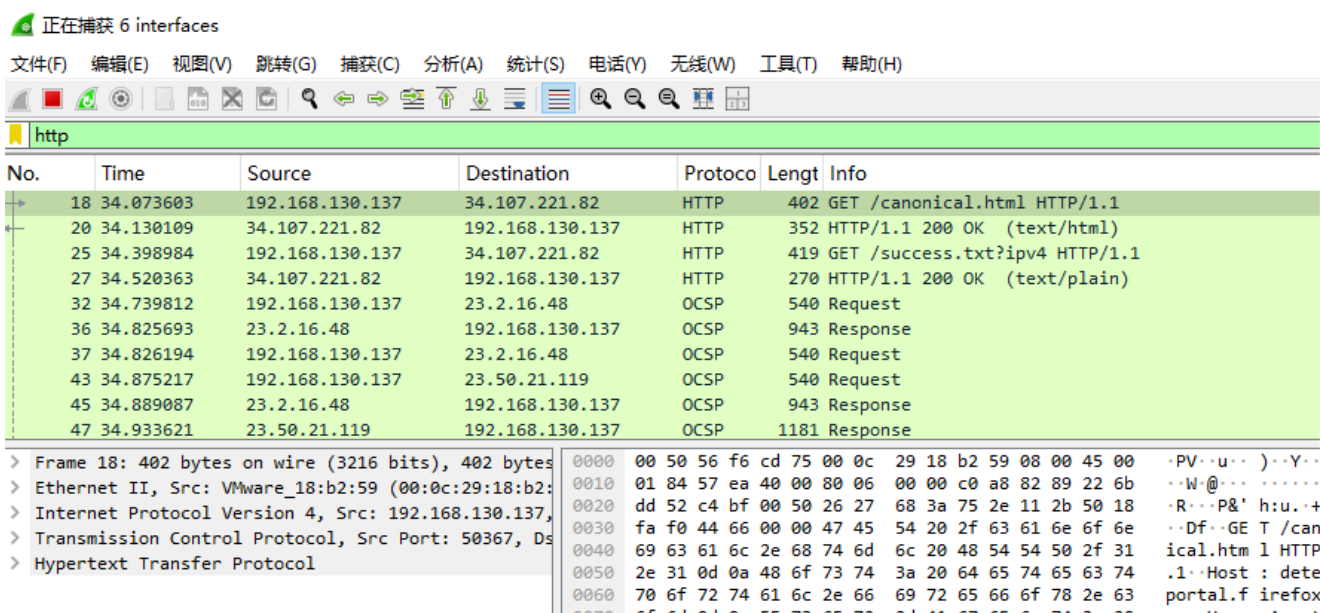


1.5. 抓包

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

正在捕获 6 interfaces

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(V) 无线(W) 工具(T) 帮助(H)

http

No.	Time	Source	Destination	Protocol	Length	Info
18	34.073603	192.168.130.137	34.107.221.82	HTTP	402	GET /canonical.html HTTP/1.1
20	34.130109	34.107.221.82	192.168.130.137	HTTP	352	HTTP/1.1 200 OK (text/html)
25	34.398984	192.168.130.137	34.107.221.82	HTTP	419	GET /success.txt?ip=4 HTTP/1.1
27	34.520363	34.107.221.82	192.168.130.137	HTTP	270	HTTP/1.1 200 OK (text/plain)
32	34.739812	192.168.130.137	23.2.16.48	OCSP	540	Request
36	34.825693	23.2.16.48	192.168.130.137	OCSP	943	Response
37	34.826194	192.168.130.137	23.2.16.48	OCSP	540	Request
43	34.875217	192.168.130.137	23.50.21.119	OCSP	540	Request
45	34.889087	23.2.16.48	192.168.130.137	OCSP	943	Response
47	34.933621	23.50.21.119	192.168.130.137	OCSP	1181	Response
49	35.153259	192.168.130.137	34.107.221.82	HTTP	419	GET /success.txt?ip=4 HTTP/1.1
51	35.208690	34.107.221.82	192.168.130.137	HTTP	270	HTTP/1.1 200 OK (text/plain)
54	35.234866	192.168.130.137	23.2.16.48	OCSP	540	Request
59	35.267460	192.168.130.137	203.208.41.2	OCSP	543	Request
65	35.275324	192.168.130.137	203.208.41.2	OCSP	543	Request
67	35.294351	23.2.16.48	192.168.130.137	OCSP	944	Response

Frame 18: 402 bytes on wire (3216 bits), 402 bytes captured (3216 bits) on interface

Ethernet II, Src: VMware_18:b2:59 (00:0c:29:18:b2:59), Dst: VMware_f6:cd:75 (00:0c:29:18:b2:59)

Internet Protocol Version 4, Src: 192.168.130.137, Dst: 34.107.221.82

Transmission Control Protocol, Src Port: 50367, Dst Port: 80, Seq: 1, Ack: 1, Len: 402

Hypertext Transfer Protocol

GET /canonical.html HTTP/1.1\r\n

Request Method: GET

Request URI: /canonical.html

Request Version: HTTP/1.1

Host: detectportal.firefox.com\r\n

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0\r\n

Accept: */*\r\n

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\n

Accept-Encoding: gzip, deflate\r\n

Cache-Control: no-cache\r\n

Pragma: no-cache\r\n

Connection: keep-alive\r\n

\r\n

[Response in frame: 20]

[Full request URI: http://detectportal.firefox.com/canonical.html]

0000 00 50 56 f6 cd 75 00 0c 29 18 b2 59 00 00 00 00

0010 01 84 57 ea 40 00 80 06 00 00 c0 a1 00 00 00 00

0020 dd 52 c4 bf 00 50 26 27 68 3a 75 2d 00 00 00 00

0030 fa f0 44 66 00 00 47 45 54 20 2f 6e 00 00 00 00

0040 69 63 61 6c 2e 68 74 6d 6c 20 48 5a 00 00 00 00

0050 2e 33 2c 65 6e 3b 71 3d 30 2e 32 00 00 00 00

0060 70 6f 72 74 61 6c 2e 68 69 72 65 6e 6f 78 2f 30

0070 6f 6d 0d 0a 55 73 65 72 2d 41 67 6e 6f 78 2f 30

0080 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 2d 00 00 00 00

0090 64 6f 77 73 20 4e 54 20 31 30 2e 30 2d 00 00 00 00

00a0 6e 36 34 3b 20 78 36 34 3b 20 72 70 2d 00 00 00 00

00b0 2e 30 29 20 47 65 63 6b 6f 2f 32 30 2d 00 00 00 00

00c0 30 31 20 46 69 72 65 66 6f 78 2f 30 2d 00 00 00 00

00d0 0d 0a 41 63 63 65 70 74 3a 20 2a 2d 00 00 00 00

00e0 63 63 65 70 74 2d 4c 61 6e 67 75 6e 6f 78 2f 30

00f0 7a 68 2d 43 4e 2c 7a 68 3b 71 3d 30 2e 37 2c 70

0100 68 2d 54 57 3b 71 3d 30 2e 37 2c 70 2d 00 00 00 00

0110 3b 71 3d 30 2e 35 2c 65 6e 2d 55 5a 2d 00 00 00 00

0120 2e 33 2c 65 6e 3b 71 3d 30 2e 32 00 00 00 00

0130 65 70 74 2d 45 6e 63 6f 64 69 6e 6f 78 2f 30

0140 69 70 2c 20 64 65 66 6c 61 74 65 00 00 00 00

0150 68 65 2d 43 6f 6e 74 72 6f 6c 3a 2d 00 00 00 00

0160 61 63 68 65 0d 0a 50 72 61 67 6d 6f 78 2f 30

0170 2d 63 61 63 68 65 0d 0a 43 6f 6e 6f 78 2f 30

0180 6f 6e 3a 20 6b 65 65 70 2d 61 6c 6e 6f 78 2f 30

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 Firefox/136.0\r\n
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\n
Accept-Encoding: gzip, deflate\r\n
Cache-Control: no-cache\r\n
Pragma: no-cache\r\n
Connection: keep-alive\r\n
\r\n

```

分析：

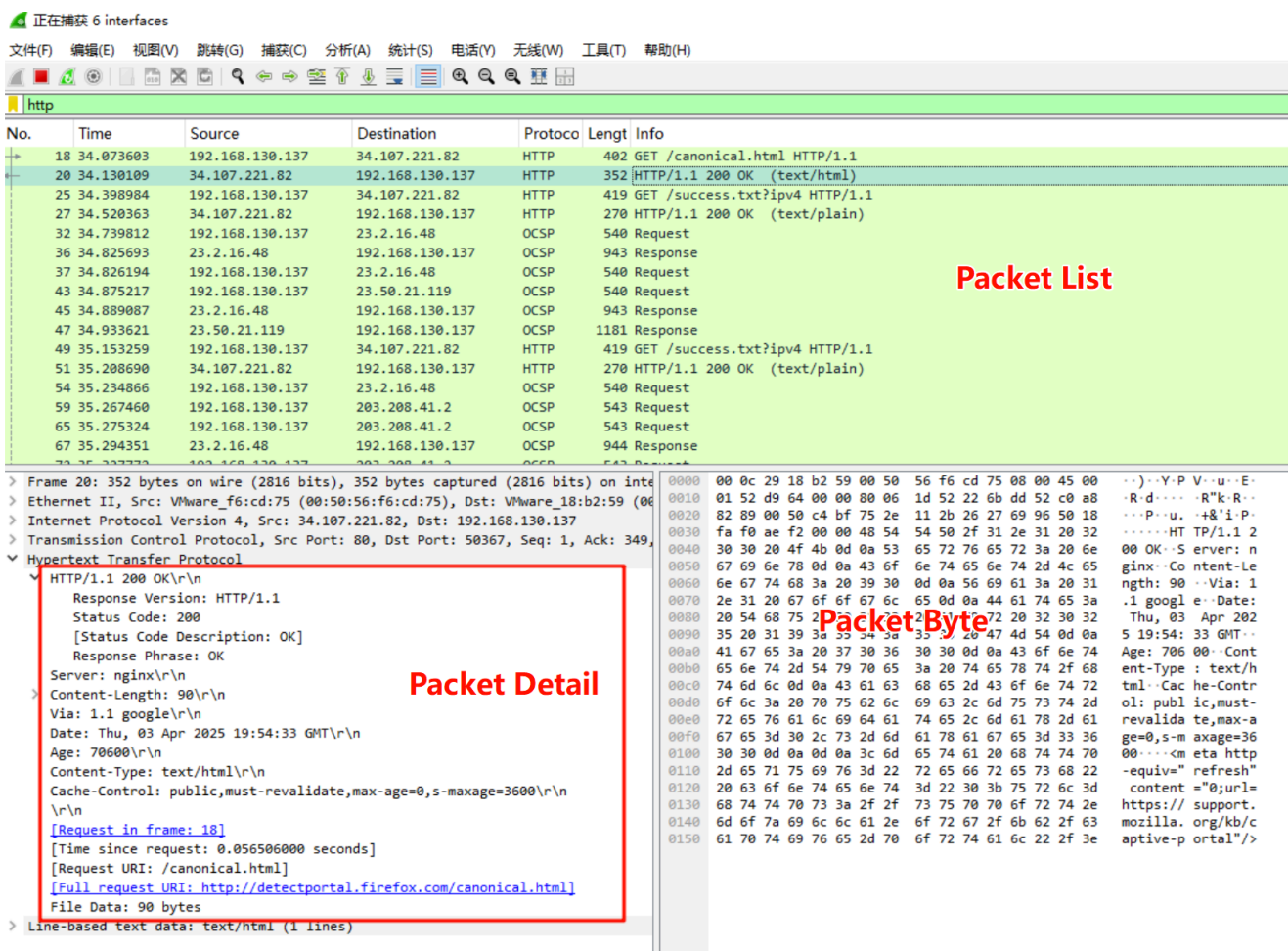
- 请求方法： GET
- 协议版本： HTTP/1.1
- Host： detectportal.firefox.com
- 客户端信息： Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
 - Windows 10系统， 64位架构
 - 使用Firefox浏览器， 版本号136.0
- 内容协商：

- 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。	<code>GET</code> ：请求获取指定资源。
Request URI	指定请求的资源路径。	<code>/canonical.html</code> ：目标资源路径。
Request Version	使用的HTTP协议版本。	<code>HTTP/1.1</code> ：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。	<code>detectportal.firefox.com</code> ：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。	<code>Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/201001</code> ：Windows 10系统，Firefox 136.0。
Accept	声明客户端可接受的响应内容类型。	<code>*/*</code> ：接受任意类型的内容。
Accept-Language	声明客户端优先接受的语言。	<code>zh-CN,zh;q=0.8...</code> ：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。	<code>gzip, deflate</code> ：支持gzip或deflate压缩。
Cache-Control	控制缓存行为。	<code>no-cache</code> ：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 <code>Cache-Control</code> 。	<code>no-cache</code> ：禁用缓存。
Connection	控制TCP连接是否保持活跃。	<code>keep-alive</code> ：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:

分析：

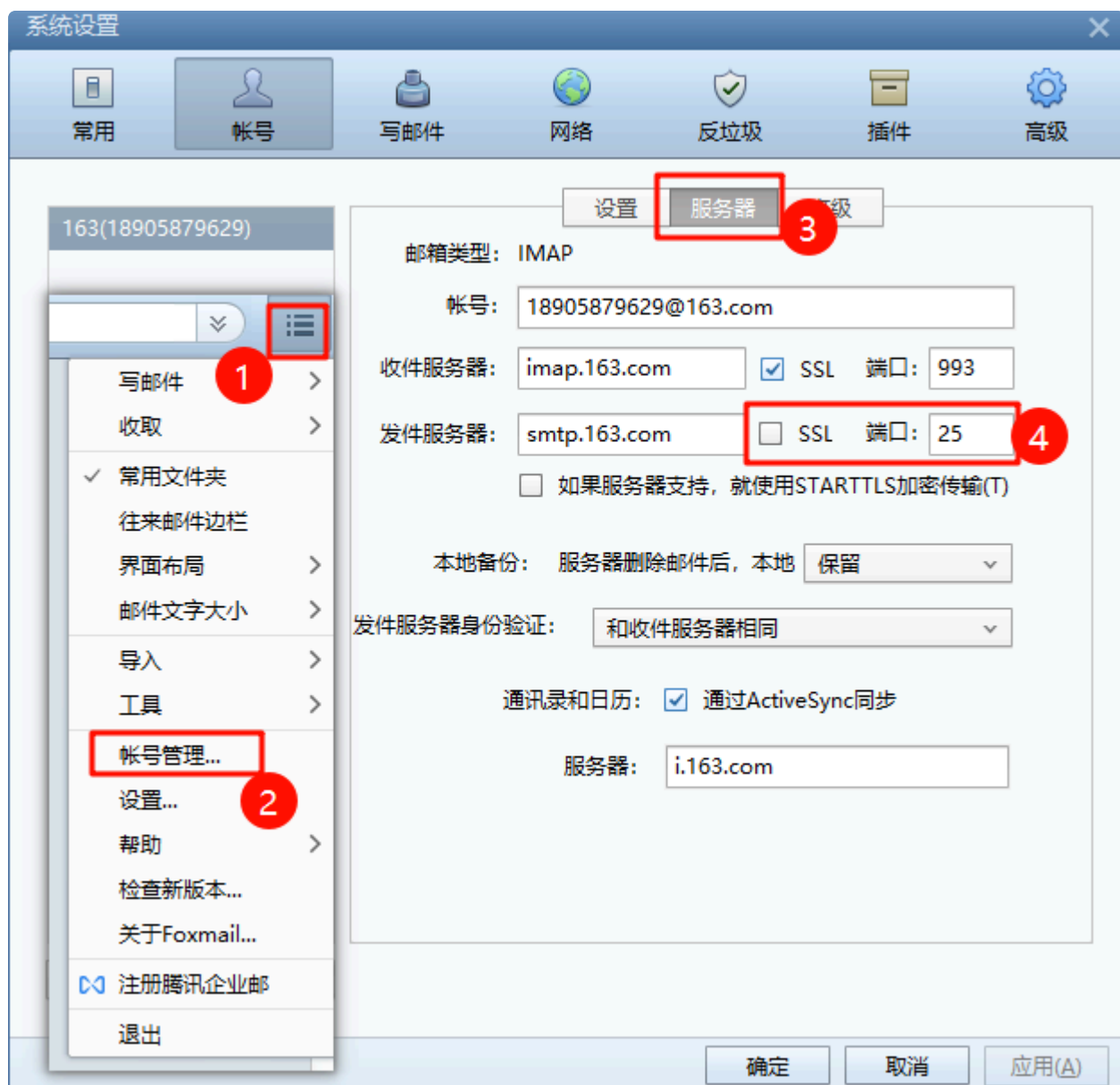
HTTP响应消息头字段总结

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。	<code>nginx</code> : 服务器使用Nginx处理请求。
Content-Length	声明响应体的字节大小。	<code>90</code> : 响应体长度为90字节。
Via	显示请求经过的代理或网关信息。	<code>1.1 google</code> : 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 <code>Thu, 03 Apr 2025 19:54:33 GMT</code>
Age	表示响应在缓存中已存储的时间(秒)。	<code>70600</code> : 响应已缓存约19.6小时。
Content-Type	声明响应体的媒体类型。	<code>text/html</code> : 内容为HTML格式。
Cache-Control	控制客户端和中间缓存的行为。	<code>public, must-revalidate, max-age=0, s-maxage=3600</code> : 公共缓存, 客户端立即失效, CDN缓存1小时。

2. SMTP协议实验

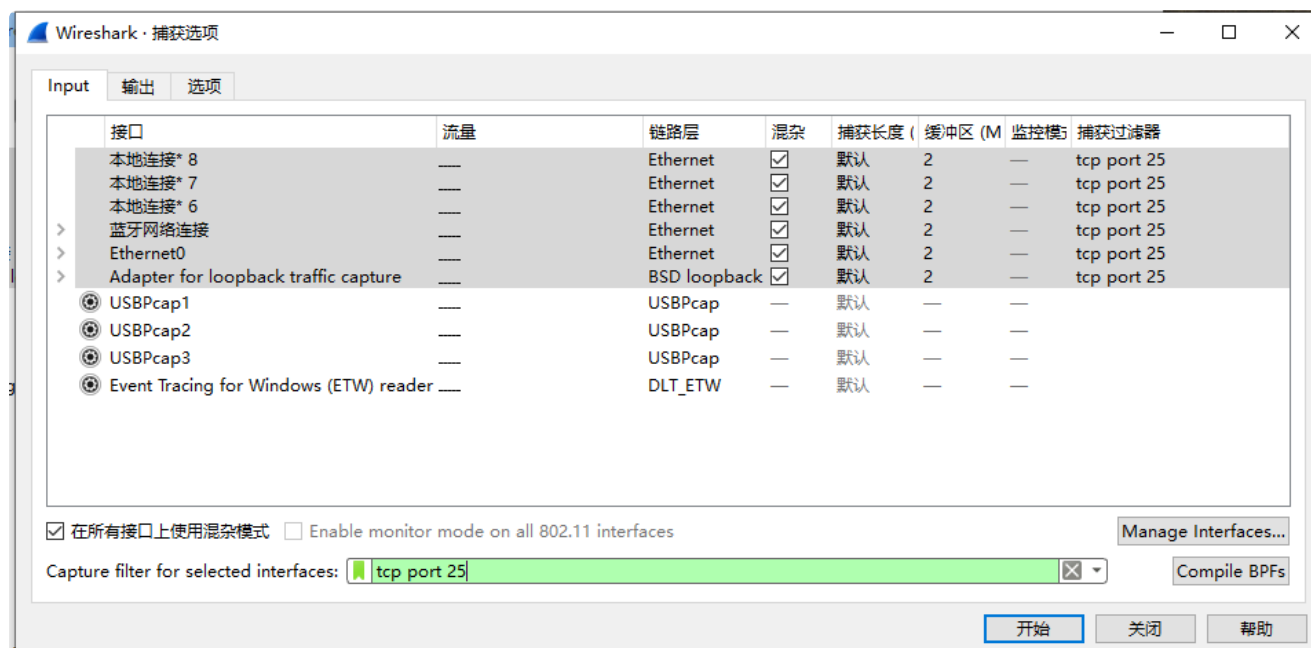
2.1. 下载Foxmail

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

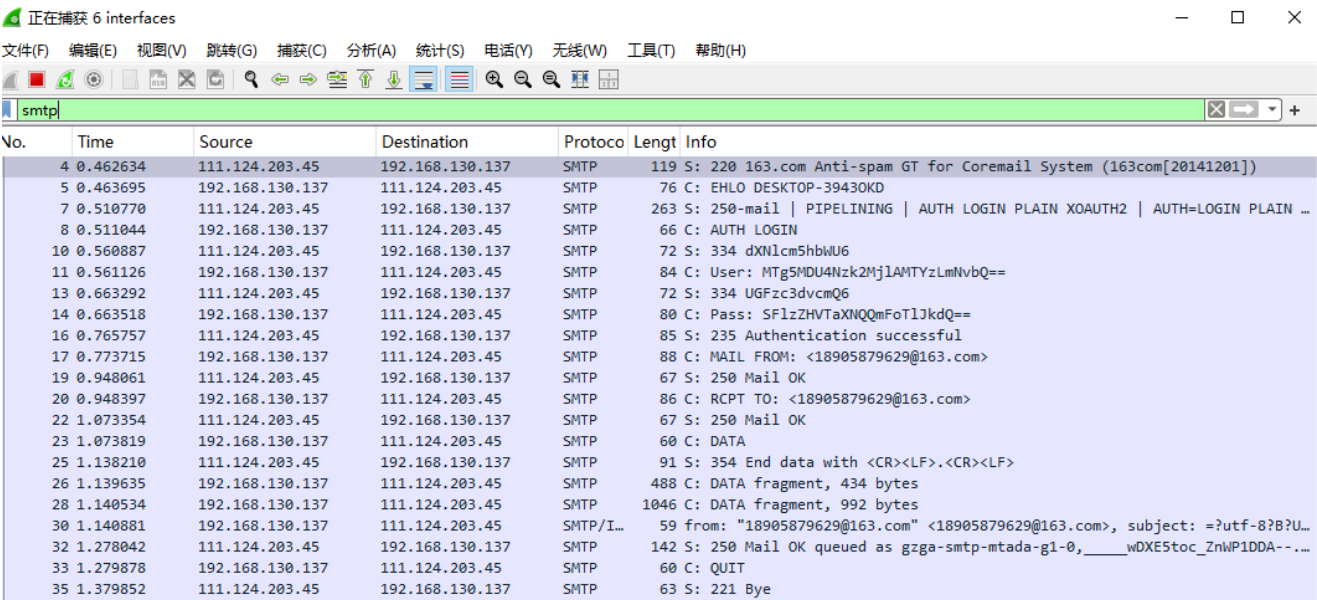
发送成功后停止截获
可以看到序号1-3为TCP的三次握手建立连接

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.130.137	111.124.203.45	TCP	66	49898 → 25 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000951	111.124.203.45	192.168.130.137	TCP	60	25 → 49898 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.001042	192.168.130.137	111.124.203.45	TCP	54	49898 → 25 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.462634	111.124.203.45	192.168.130.137	SMTP	119	S: 220 163.com Anti-spam GT for Coremail System (163com[20141201])
5	0.463695	192.168.130.137	111.124.203.45	SMTP	76	C: EHLO DESKTOP-3943OKD
6	0.463890	111.124.203.45	192.168.130.137	TCP	60	25 → 49898 [ACK] Seq=66 Ack=23 Win=64240 Len=0
7	0.510770	111.124.203.45	192.168.130.137	SMTP	263	S: 250-mail PIPELINING AUTH LOGIN PLAIN XOAUTH2 AUTH=LOGIN PLAIN ...
8	0.511044	192.168.130.137	111.124.203.45	SMTP	66	C: AUTH LOGIN
9	0.511220	111.124.203.45	192.168.130.137	TCP	60	25 → 49898 [ACK] Seq=275 Ack=35 Win=64240 Len=0
10	0.560887	111.124.203.45	192.168.130.137	SMTP	72	S: 334 dXNlcm5hbWU6

> Frame 4: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on Ethernet II, Src: VMware_f6:cd:75 (00:50:56:f6:cd:75), Dst: VMware_18:b2: Internet Protocol Version 4, Src: 111.124.203.45, Dst: 192.168.130.137 Transmission Control Protocol, Src Port: 25, Dst Port: 49898, Seq: 1, Ac Simple Mail Transfer Protocol

0000 00 0c 29 18 b2 59 00 50 56 f6 cd 75 08 00 45 00 00 69 71 24 00 00 80 06 4b 8f 6f 7c cb 2d c0 a8 82 89 00 19 c2 ea 18 05 3f 2c aa 00 4b 98 50 18 fa f0 be 4a 00 00 32 32 30 20 31 36 33 2e 63 6f 6d 20 41 6e 74 69 2d 73 70 61 6d 20 47 54 20 66 6f 72 20 43 6f 72 65 6d 61 69 6c 20 53 79 73 74 65 6d 20 28 31 36 33 63 6f 6d 5b 32 30 31 34 31

筛选SMTP



2.5. SMTP协议的分析

Protocol	Length	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: MTg5MDU4Nzk2MjIAMSjYzLmNvbQ==
SMTP	72 S:	334 UGFzc3dvcmQ6
SMTP	80 C:	Pass: SF1zZHVTaXNQOmFoTlJkdQ==
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'MTg5MDU4Nzk2MjIAMSjYzLmNvbQ=='))
5 print(base64.b64decode(b'SF1zZHVTaXNQOmFoTlJkdQ==`'))
6
```

问题 输出 调试控制台 终端 端口 筛选器

[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: MTg5MDU4Nzk2MjIAMSzYzLmNvbQ==`

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: SFlzZHVtZXNlcmFoTlJkdQ==`

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

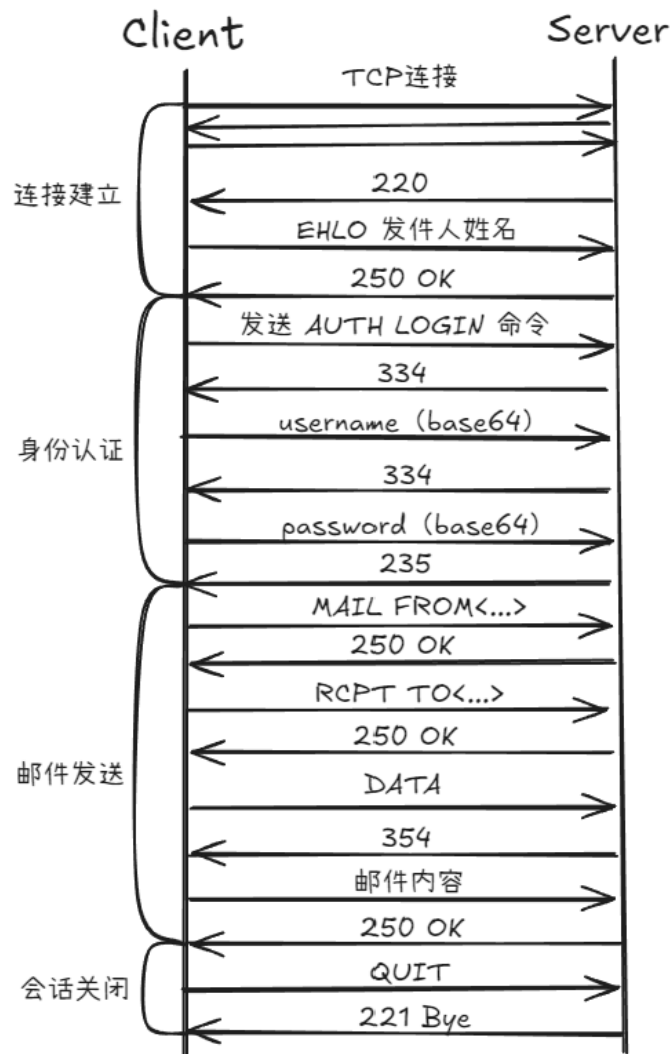
1. 客户端请求关闭SMTP会话:

`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`。

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation
C:\Users\admin>telnet smtp.163.com 25
```

将邮箱号和授权码转为base64编码：

```
smtp.py
1  import base64
2  print(base64.b64encode(b'18905879629@163.com'))
3  print(base64.b64encode(b'HYsduSisPBahNRdu'))
4
```

问题 输出 调试控制台 终端 端口

```
ect/smtp.py
b'MTg5MDU4Nzk2MjlAMTYzLmNvbQ=='
b'SFlzZHVTaXNQmFoTlJkdQ=='
```

依次输入

```
EHLO DESKTOP
AUTH LOGIN
MTg5MDU4Nzk2MjlAMTYzLmNvbQ== //18905879629@163.com的base64编码
SFlzZHVTaXNQmFoTlJkdQ== //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
```

```
Telnet smtp.163.com
EHLO DESKTOP
250-mail
250-PIPELINING
250-AUTH LOGIN PLAIN XOAUTH2
250-AUTH=LOGIN PLAIN XOAUTH2
250-coremail 1Uxr2xKj7kG0xki17xGrU7I0s8FY2U3Uj8Cz28x1UUUUU7Ic2I0Y2UFzzlgqUCa0xDrUUUUj
250-STARTTLS
250-ID
250 8BITMIME
AUTH LOGIN
334 dXNlcm5hbWU6
MTg5MDU4Nzk2MjIAMSzYzLmNvbQ==
334 UGFzc3dvcmQ6
SFlzZHVtYXN0QmFoTlJkdQ==
235 Authentication successful
MAIL FROM:<18905879629@163.com>
250 Mail OK
RCPT TO:<18905879629@163.com>
250 Mail OK
DATA
354 End data with <CR><LF>.<CR><LF>
To:18905879629@163.com
FROM:18905879629@163.com
Subject:Test

send a text
.
250 Mail OK queued as gzga-smtp-mtada-g1-2,_____wD3X7eCjPZnFN1+FQ--.56118S2 1744211158
```

邮箱的base64编码

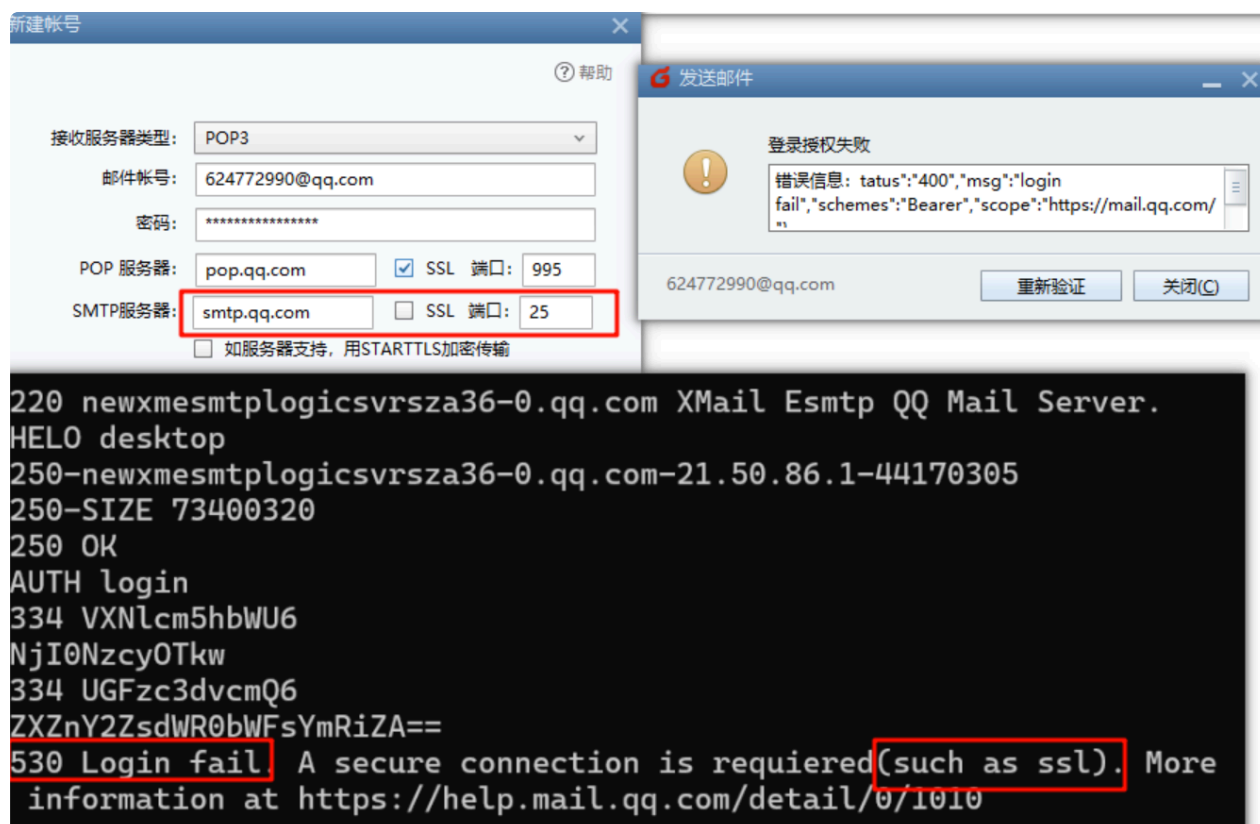
授权码的base64编码

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。





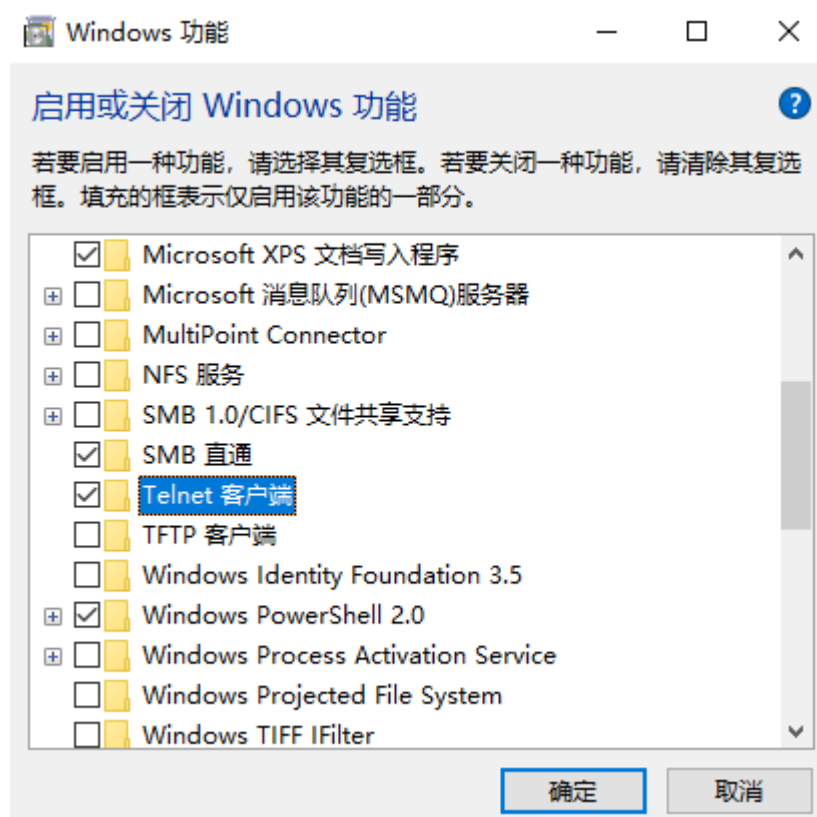
3. 问题：Telnet无法打开。

解决：“启用或关闭Windows功能”中，勾选Telnet客户端

```
C:\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation

C:\Users\admin>telnet smtp.qq.com 25
'telnet' is not recognized as an internal or external command,
operable program or batch file.
```



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。