

实验二：使用 Wireshark 软件分析 HTTP 协议

学号：16307130194，姓名：陈中钰

1 分析其中一条请求报文

1.1 查看当前网络的 IP 地址

当前连接的网络为 WLAN。接着通过 `ipconfig` 命令，查看 WLAN 信息，如 Figure 1 所示，可知 WLAN 的 IP 地址为 192.168.1.108。由于当前连接的网络为 WLAN，因此该 IP 地址就是当前网络的 IP 地址。

无线局域网适配器 WLAN:

```
连接特定的 DNS 后缀 . . . . . : DHCP HOST
本地链接 IPv6 地址. . . . . : fe80::b9e4:e950:ff39:a706%12
IPv4 地址 . . . . . : 192.168.1.108
子网掩码 . . . . . : 255.255.255.0
默认网关. . . . . : 192.168.1.1
```

Figure 1: 查看 WLAN 的 IP 地址

1.2 捕获分组

打开 Wireshark，设置显示过滤器为 `ip.addr == 192.168.1.108 and http`，该 IP 地址为 1.1 获得的当前网络的 IP 地址。由于当前连接的网络为 WLAN，于是选择要捕获的网络为 WLAN。点击开始捕获分组按钮，打开 Chrome 浏览器访问 www.163.com，在网页加载完后点击停止捕获分组按钮。接下来分析捕获的分组。

1.3 请求报文选取

从封包列表中找出 Source 为当前网络 IP 地址、Destination 为目标网站 IP 地址、Protocol 为 HTTP 的封包，这就是需要分析的请求报文。从符合要求的封包中选择了一条，并查看封包详细信息中的 Hypertext Transfer Protocol (HTTP) 部分，如 Figure 2 所示。

1.4 浏览器可以接受哪些类型的文件？

如 Figure 2 所示，Accept 字段的内容就是浏览器接受的文件类型，包括 MIME 类型和子类型，其中 text 类有 html 子类型、application 类有 xhtml 和 xml 子类型、image 类有 jxr 子类型。

1.5 可以接受哪些编码的文件？

如 Figure 2 所示，Accept-Encoding 字段的内容就是接受的编码格式，包括 gzip、deflate 编码格式。



Figure 2: 请求报文的 HTTP 部分

1.6 除以上回答过的字段，头部还包含哪些字段？它们的含义是什么？

如 Figure 2 所示，除了上述 Accept、Accept-Encoding 字段外，请求头部还有其他字段。Accept-Language 字段表示接受的语言，在这里是中文；User-Agent 字段表示用户代理的字符串值，内容如 Figure 2 所示；Host 字段表示服务器域名和 TCP 端口号，在这里服务器域名是 www.163.com，而由于使用的是服务请求的标准端口号 8080，因此端口号省略了；Connection 字段设置当前连接，在这里为 Keep-Alive，表示需要建立 HTTP 长连接。

在请求头部的前面是请求行，包括了 Request Method 为 GET、Request URI 为 /、Request Version 为 HTTP/1.1。

在请求头部的后面是回车符和换行符，而再后面就是请求正文。

2 分析其中一条响应报文

2.1 响应报文选取

从封包列表找出 Source 为目标网站 IP 地址、Destination 为当前网络 IP 地址、Protocol 为 HTTP 的封包，这就是需要分析的响应报文。从符合要求的封包中选择了一条，并查看封包详细信息中的 Hypertext Transfer Protocol (HTTP) 部分，如 Figure 3 所示。

2.2 服务器返回了什么类型的文件？

如 Figure 3 所示，Content-Type 字段表示返回的文件类型，包括 MIME 类型和子类型，在这里是 application 类的 octet-stream 子类型（不间断的字节流）文件。

2.3 服务器返回了多少字节的内容？

如 Figure 3 所示，Content-Length 字段表示服务器返回内容的大小，单位是字节。另外，File Data 字段也表示服务器返回内容的大小，并且包含了单位为字节。在这里服务器返回了 197 字节的内容。

2.4 尝试从响应报文字段确定所访问的网站使用的是何种服务器软件？

如 Figure 3 所示，Server 字段表示服务器软件名称，因此所访问的网站使用的服务器软件是 nginx。

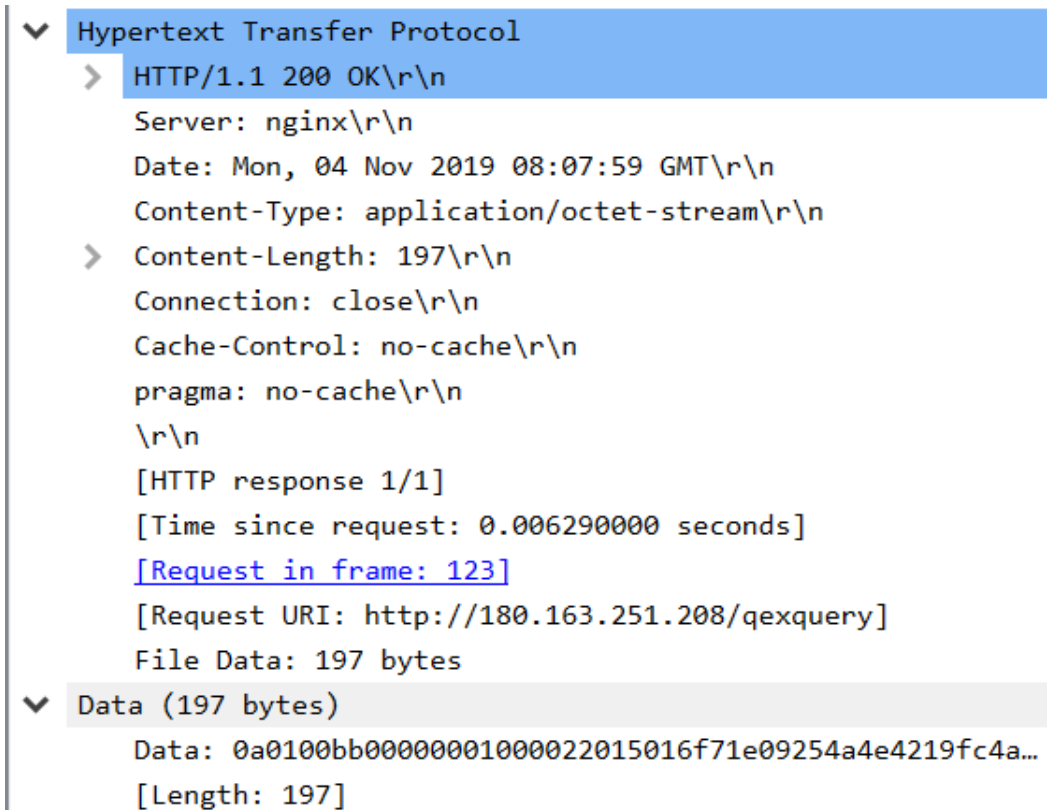


Figure 3: 响应报文的 HTTP 部分

3 访问一个内容丰富的网站（大段文字、多图），加载完毕后刷新页面，分析在此期间捕获的数据包

3.1 捕获分组

按照1.2中的设置来准备捕获分组，接着点击开始捕获分组，访问www.news.sh.cn，加载完毕后刷新页面，然后停止捕获分组。接下来分析在此期间捕获的数据包。

3.2 获取的页面的最后修改时间（last-modified）是什么时候？

按照2.1中的要求找到一条响应报文，该响应报文返回了 gif 类型文件，查看它的封包详细信息中的 HTTP 部分，如 Figure 4 中所示。HTTP 响应头部中的 Last-Modified 字段表示该 gif 类型文件的最后修改时间，为 Sat, 08 Apr 2017 19:13:10 GMT。通过查看返回其他文件的响应报文，可以发现不同的文件有着不同的最后修改时间。

3.3 分析显示信息为 Continuation or non-HTTP traffic 的数据包，此种类型的数据包是什么含义？其最大有效载荷为多少字节？为什么？

按照2.1中的要求找到响应报文，而且 Info 信息栏的内容为 Continuation。如 Figure 5 所示，可以找到一连串符合上述要求的响应报文。这是因为当客户端向服务端请求一个数据量很大的 HTTP 对象时，数据大小超过了一个 HTTP 响应报文的最大小时，需要把对象分成多个响应报文进行发送，而中间的响应报文只需要传输数据，不需要包含状态行、头部行等其他信息。在 Figure 5 中，Info 栏内容为 Continuation 的响应报文就是这种只需要传输数据的数据包。

通过查看这种数据包的封包详细信息的 HTTP 部分，如 Figure 5 所示，可以看到 HTTP 响应报文只包括了 524 字节的数据，所以它的最大有效载荷就是 524 字节。另外，HTTP 响应报文的大小限制是由服务器设置的。

```

▼ Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Server: NWS_CDN_P1\r\n
    Connection: keep-alive\r\n
    Date: Tue, 05 Nov 2019 02:03:28 GMT\r\n
    Cache-Control: max-age=259200\r\n
    Expires: Fri, 08 Nov 2019 02:03:28 GMT\r\n
    Last-Modified: Sat, 08 Apr 2017 19:13:10 GMT\r\n
    Content-Type: image/gif\r\n
  > Content-Length: 1410\r\n
    X-NWS-LOG-UUID: 9998040275169185433 2107abdde38741480fc0afbef76a41bf\r\n
    Vary: Origin\r\n
    X-Cache-Lookup: Hit From Disktank3\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.034874000 seconds]
    [Request in frame: 587]
    [Next request in frame: 2397]
    [Next response in frame: 2400]
    [Request URI: http://pub.idqqimg.com/qconn/wpa/button/button_old_81.gif]
    File Data: 1410 bytes

```

Figure 4: 返回 gif 类型文件的响应报文的 HTTP 部分

3.4 分析 HTTP 状态码为 304 的数据包，解释其含义？

在客户端向服务端请求一个对象的时候，如果发现自己曾经缓存过这个对象，那么在请求头部中会包含 If-Modified-Since 字段，这个字段就是已经缓存的对象的 Last-Modified 字段。这时候服务端收到的请求报文包含 If-Modified-Since 字段，会把该字段和目标对象的 Last-Modified 字段进行比较，如果的 Last-Modified 字段比 If-Modified-Since 字段的时间更晚，说明该对象在服务端更新过，而客户端缓存的对象已经过期了，于是服务端会返回 HTTP 状

690	3.004442	192.168.1.105	159.138.29.18	HTTP	433	GET /i/bxj.jpg HTTP/1.1
750	3.057618	159.138.29.18	192.168.1.105	HTTP	590	Continuation
751	3.057618	159.138.29.18	192.168.1.105	HTTP	590	Continuation
752	3.057619	159.138.29.18	192.168.1.105	HTTP	590	Continuation
753	3.057619	159.138.29.18	192.168.1.105	HTTP	590	Continuation
758	3.059799	159.138.29.18	192.168.1.105	HTTP	590	Continuation
759	3.059799	159.138.29.18	192.168.1.105	HTTP	590	Continuation
826	3.096942	159.138.29.18	192.168.1.105	HTTP	590	Continuation
827	3.096942	159.138.29.18	192.168.1.105	HTTP	590	Continuation
828	3.096942	159.138.29.18	192.168.1.105	HTTP	590	Continuation
829	3.096943	159.138.29.18	192.168.1.105	HTTP	590	Continuation
830	3.096943	159.138.29.18	192.168.1.105	HTTP	590	Continuation
832	3.096943	159.138.29.18	192.168.1.105	HTTP	590	Continuation
833	3.096944	159.138.29.18	192.168.1.105	HTTP	590	Continuation
855	3.099892	159.138.29.18	192.168.1.105	HTTP	590	Continuation
856	3.099893	159.138.29.18	192.168.1.105	HTTP	590	Continuation


```

<
  > Frame 758: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface 0
  > Ethernet II, Src: Tp-LinkT_58:c3:89 (f4:83:cd:58:c3:89), Dst: IntelCor_5f:31:1a (ac:2b:6e:5f:31:1a)
  > Internet Protocol Version 4, Src: 159.138.29.18, Dst: 192.168.1.105
  > Transmission Control Protocol, Src Port: 80, Dst Port: 60741, Seq: 25969, Ack: 1093, Len: 624
  ▼ Hypertext Transfer Protocol
    ▼ Data (524 bytes)
      Data: f9c7fa57fae7fb77fc07fc98fd29fdbafe4bfedcfff6dffff...
      [Length: 524]

```

Figure 5: 显示信息为 Continuation 的响应报文

状态码 200，并返回对象的数据，接着客户端会更新对应的缓存。但是如果 Last-Modified 字段的时间并没有比 If-Modified-Since 字段的时间更晚，那么说明客户端缓存的对象仍是最新版本，因此服务端返回 HTTP 状态码 304，而且不返回对象的数据，客户端可以直接使用该对象的缓存。通过这种缓存机制，可以使网站的二次访问速度更快，并减少带宽的消耗。

3.5 页面上的内容（HTML 文档、图片、Javascript 脚本、CSS 文件等）是否来自于同一个服务器？判断的依据是什么？若不是，这样做的好处是什么？

找到返回了不同文件的响应报文，查看它们的 Source，发现 Source 的 IP 地址并不一定相同。如 Figure 6 所示，No.456 响应报文返回的是 png 类型文件，服务器 IP 地址为 159.138.29.18，而 No.596 响应报文返回的是 gif 类型文件，服务器 IP 地址为 113.105.165.74，说明了两份文件并不是来自于同一个服务器。所以，不同的文件是来自于不同的服务器的，因为不同文件的 Source 的 IP 地址并不相同。

把不同的文件放在不同的服务器中，可以分散服务器的压力，提高网页加载速度，同时也可以更有条理地分类整理文件。

No.	Time	Source	Destination	Protocol	Length	Info
438	2.547558	192.168.1.105	159.138.29.18	HTTP	435	GET /i/jbhnk.jpg HTTP/1.1
456	2.549547	159.138.29.18	192.168.1.105	HTTP	374	HTTP/1.1 200 OK (PNG)
464	2.551757	192.168.1.105	159.138.29.18	HTTP	432	GET /i/xj.jpg HTTP/1.1
471	2.568324	192.168.1.105	14.215.158.24	HTTP	531	GET /pa?p=1:191646616:8 HTTP/1.1
494	2.608588	14.215.158.24	192.168.1.105	HTTP	467	HTTP/1.1 301 Moved Permanently (text/html)
587	2.817170	192.168.1.105	113.105.165.74	HTTP	451	GET /qconn/wpa/button/button_old_81.gif HTTP/1.1
596	2.852044	113.105.165.74	192.168.1.105	HTTP	1464	HTTP/1.1 200 OK (GIF89a)
612	2.932496	159.138.29.18	192.168.1.105	HTTP	590	Continuation

Figure 6: 不同的文件来自不同的服务器

3.6 利用浏览器的控制台调试工具的 Network 页面帮助理解上一小问？

打开 Chrome 浏览器的控制台调试工具的 Network 页面，并加载网站。如 Figure 7 所示，可以发现，由于不同的文件来自不同的服务器，所以不同服务器的文件可以并行地加载，使网站加载速度大大提升，并且降低了单个服务器的压力。

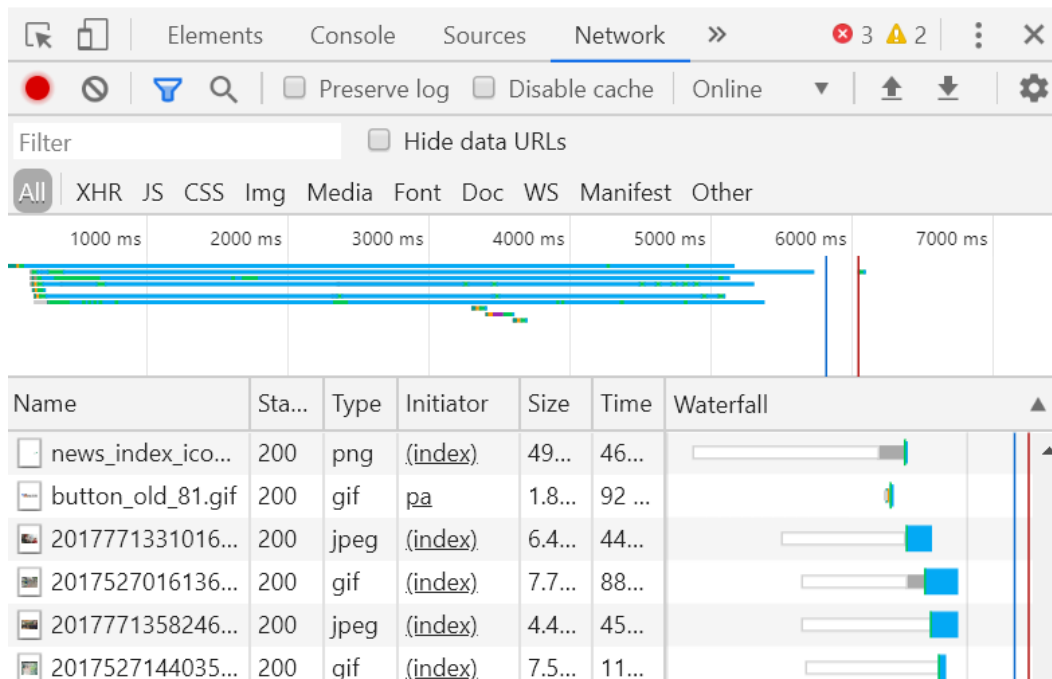


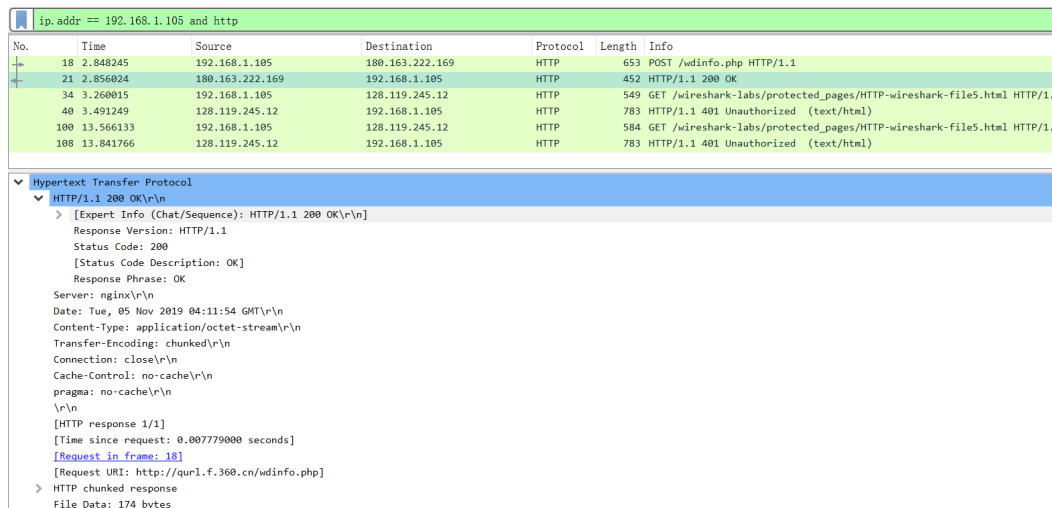
Figure 7: Chrome 浏览器的控制台调试工具的 Network 页面

4 问题 4

访问http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html，在弹出的对话框中输入用户名 abc 和密码 123，然后进行分析。

4.1 服务器对最初的 HTTP 消息的响应（状态码和短语）是什么？

按照2.1中的设置来捕获分组，结果如 Figure 8所示。其中，No.21 封包就是服务器对最初的 HTTP 消息的响应。通过观察它的封包详细信息的 HTTP 部分，可以发现它的状态码是 200，短语是 OK。



No.	Time	Source	Destination	Protocol	Length	Info
18	2.848245	192.168.1.105	180.163.222.169	HTTP	653	POST /wdinfo.php HTTP/1.1
21	2.856024	180.163.222.169	192.168.1.105	HTTP	452	HTTP/1.1 200 OK
34	3.260015	192.168.1.105	128.119.245.12	HTTP	549	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
40	3.491249	128.119.245.12	192.168.1.105	HTTP	783	HTTP/1.1 401 Unauthorized (text/html)
100	13.566133	192.168.1.105	128.119.245.12	HTTP	584	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
108	13.841766	128.119.245.12	192.168.1.105	HTTP	783	HTTP/1.1 401 Unauthorized (text/html)

Hypertext Transfer Protocol	
HTTP/1.1 200 OK\r\n	
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]	
Response Version: HTTP/1.1	
Status Code: 200	
[Status Code Description: OK]	
Response Phrase: OK	
Server: nginx\r\n	
Date: Tue, 05 Nov 2019 04:11:54 GMT\r\n	
Content-Type: application/octet-stream\r\n	
Transfer-Encoding: chunked\r\n	
Connection: close\r\n	
Cache-Control: no-cache\r\n	
pragma: no-cache\r\n	
\r\n	
[HTTP response 1/1]	
[Time since request: 0.007779000 seconds]	
[Request in frame: 18]	
[Request URI: http://qr1.f.360.cn/wdinfo.php]	
> HTTP chunked response	
File Data: 174 bytes	

Figure 8: 最初的 HTTP 响应报文

4.2 与前一部分实验相比，在这个响应消息中出现了什么新的字段？

如 Figure 8所示，查看这个响应报文的封包详细信息的 HTTP 部分。和 Figure 3中的响应报文相比，可以发现新字段 Transfer-Encoding，其内容为 chunked，这表示设置传输实体的编码格式为 chunked。

4.3 当浏览器第二次发送 HTTP GET 消息时，有什么新的字段被包含在 HTTP GET 消息中？

如 Figure 9所示，第二次发送的 HTTP GET 消息就是 No.100 封包，从图中还可以看到它的封包详细信息的 HTTP 部分。和 Figure 2中的请求报文相比，新的字段是 Authorization、Upgrade-Insecure-Requests 字段；和第一次发送的 HTTP GET 消息相比，新的字段只有 Authorization。其中，Authorization 是 HTTP 身份验证的凭证，内容就是所输入的用户名 abc 和密码 123. 另外，Upgrade-Insecure-Requests 字段的内容为 1，表示服务器可以处理 HTTPS 协议。

5 问题 5

访问<https://www.baidu.com/>，并进行分析。

5.1 是否能在 Wireshark 中正常捕获 http 数据包？

按照1.2的过滤规则进行数据包捕获，不能捕获到 http 数据包。

ip.addr == 192.168.1.105 and http						
No.	Time	Source	Destination	Protocol	Length	Info
18	2.848245	192.168.1.105	180.163.222.169	HTTP	653	POST /wdinfo.php HTTP/1.1
21	2.856024	180.163.222.169	192.168.1.105	HTTP	452	HTTP/1.1 200 OK
34	3.260015	192.168.1.105	128.119.245.12	HTTP	549	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
40	3.491249	128.119.245.12	192.168.1.105	HTTP	783	HTTP/1.1 401 Unauthorized (text/html)
100	13.566133	192.168.1.105	128.119.245.12	HTTP	584	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
108	13.841766	128.119.245.12	192.168.1.105	HTTP	783	HTTP/1.1 401 Unauthorized (text/html)

>	Ethernet II, Src: IntelCor_5f:31:1a (ac:2b:6e:5f:31:1a), Dst: Tp-LinkT_58:c3:89 (f4:83:cd:58:c3:89)
>	Internet Protocol Version 4, Src: 192.168.1.105, Dst: 128.119.245.12
>	Transmission Control Protocol, Src Port: 62244, Dst Port: 80, Seq: 1, Ack: 1, Len: 518
>	Hypertext Transfer Protocol
>	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1\r\n
>	[Expert Info (Chat/Sequence): GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1\r\n]
>	Request Method: GET
>	Request URI: /wireshark-labs/protected_pages/HTTP-wireshark-file5.html
>	Request Version: HTTP/1.1
>	Host: gaia.cs.umass.edu\r\n
>	Connection: keep-alive\r\n
>	Authorization: Basic YWJjOjEyYkU==\r\n
>	Credentials: abc:123
>	Upgrade-Insecure-Requests: 1\r\n
>	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.87 Safari/537.36\r\n
>	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3\r\n
>	Accept-Encoding: gzip, deflate\r\n
>	Accept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\n
>	\r\n
>	[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html]
>	[HTTP request 1/1]
>	[Response in frame: 108]

Figure 9: 第二次发送的 HTTP GET 消息

5.2 将过滤规则设置为 tcp.port==443 and ip.addr==your_ip_addr and ssl，刷新百度页面，根据捕获到的数据包，查阅相关资料，理解 HTTPS 连接的过程及作用？

按照题目要求设置过滤规则，并进行数据包的捕获，结果如 Figure 10所示。对 HTTPS 连接的过程及作用分析如下：

1. Client Hello

No.22 封包为 Client Hello，客户端发送请求给服务端。其中，发送了一个客户端生成的随机数 Random，在后面会使用到。另外，还发送了客户端支持的加密算法组件 Cipher Suites，服务端后续会选出一个最适合的组件来进行加密通信。

2. Server Hello

No.47 封包为 Server Hello，服务端回应客户端请求。其中，返回了一个服务端生成的随机数 Random，在后面会使用到。另外，还把选中的加密算法组件 Cipher Suite 发送给客户端。

3. Certificate, Server Key Exchange, Server Hello Done

No.49 封包为 Certificate, Server Key Exchange, Server Hello Done。其中，Certificate 表示服务端把数据证书发给客户端，证书中包含了服务端非对称加密的公钥，后续客户端验证通过后会取出来，在后续加密中使用。接着是 Server Key Exchange，根据具体的加密算法组件（在这里是 Diffie-Hellman 算法），把需要的参数发送给客户端。最后是 Server Hello Done，告诉客户端这一阶段结束了。

4. Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

No.69 封包为 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message。其中，Client Key Exchange 是客户端生成随机数，使用从证书中提取的公钥进行加密，然后发送给服务端。接着是 Change Cipher Spec，客户端告知服务端，之后发送消息都是经过加密的。接着是 Encrypted Handshake Message，通过客户端生成的 Random、服务端生成的 Random 和客户端刚刚生成的随机数来生成之后用于通信对称加密的密钥。

5. Change Cipher Spec, Encrypted Handshake Message

No.92 封包为 Change Cipher Spec, Encrypted Handshake Message。其中，Change Cipher Spec 是服务端告知客户端，之后返回的消息都是经过加密的。接着是 Encrypted Handshake Message，服务端也生成之后用于通信对称加密的密钥。

6. Application Data

No.93 封包为 Application Data，这时候 HTTPS 的连接已经建立好，于是接着进行一次通讯测试，其中所发送的数据就是经过密钥加密的。在这之后的全部通讯都是通过密钥加密了的，直到连接断开。

ip.addr == 192.168.1.105 and ssl and tcp.port == 443						
No.	Time	Source	Destination	Protocol	Length	Info
22	2.182173	192.168.1.105	180.163.198.32	TLSv1.2	571	Client Hello
25	2.184899	180.101.49.11	192.168.1.105	TLSv1.2	1203	Application Data
27	2.186875	180.101.49.11	192.168.1.105	TLSv1.2	875	Application Data, Application Data
30	2.186877	180.101.49.11	192.168.1.105	TLSv1.2	1203	Application Data
31	2.186878	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data
33	2.186879	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
36	2.186881	180.101.49.11	192.168.1.105	TLSv1.2	1037	Application Data
37	2.186881	180.101.49.11	192.168.1.105	TLSv1.2	328	Application Data, Application Data
39	2.186882	180.101.49.11	192.168.1.105	TLSv1.2	92	Application Data
43	2.187484	180.101.49.11	192.168.1.105	TLSv1.2	1203	Application Data
44	2.187485	180.101.49.11	192.168.1.105	TLSv1.2	179	Application Data
47	2.192810	180.163.198.32	192.168.1.105	TLSv1.2	1494	Server Hello
49	2.192812	180.163.198.32	192.168.1.105	TLSv1.2	1205	Certificate, Server Key Exchange, Server Hello Done
53	2.195678	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
57	2.196434	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
60	2.196437	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
65	2.198512	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
66	2.198513	180.101.49.11	192.168.1.105	TLSv1.2	1494	Application Data, Application Data
67	2.198514	180.101.49.11	192.168.1.105	TLSv1.2	1101	Application Data, Application Data
69	2.217645	192.168.1.105	180.163.198.32	TLSv1.2	139	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
71	2.224482	180.163.198.32	192.168.1.105	TLSv1.2	320	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
72	2.224482	180.163.198.32	192.168.1.105	TLSv1.2	115	Application Data
75	2.319280	192.168.1.105	180.101.49.11	TLSv1.2	478	Application Data
78	2.330995	192.168.1.105	180.101.49.11	TLSv1.2	706	Application Data
83	2.342438	192.168.1.105	180.101.49.11	TLSv1.2	571	Client Hello
84	2.342541	180.101.49.11	192.168.1.105	TLSv1.2	737	Application Data
85	2.343568	192.168.1.105	180.101.49.11	TLSv1.2	849	Application Data
87	2.352388	180.101.49.11	192.168.1.105	TLSv1.2	150	Server Hello
88	2.352388	180.101.49.11	192.168.1.105	TLSv1.2	60	Change Cipher Spec
89	2.352388	180.101.49.11	192.168.1.105	TLSv1.2	99	Encrypted Handshake Message
92	2.353357	192.168.1.105	180.101.49.11	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
93	2.358323	180.101.49.11	192.168.1.105	TLSv1.2	1366	Application Data

Figure 10: 在 HTTPS 连接过程中捕获的封包