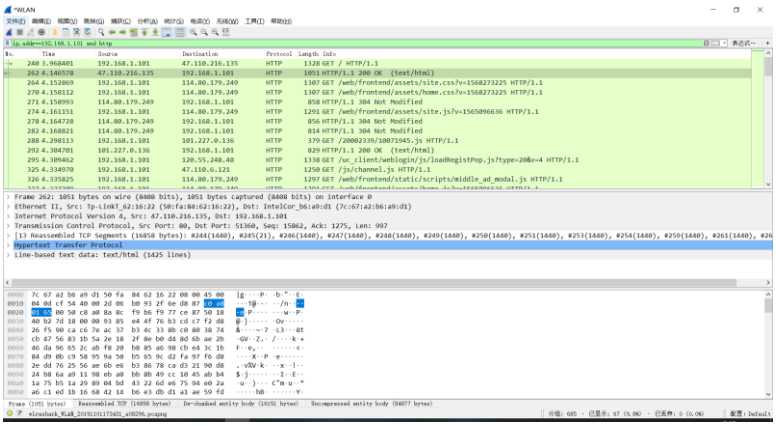
Lab2 使用 Wireshark 分析 HTTP 协议

17307130178 宁晨然

一、实验准备

设置显示过滤器为 ip.addr==192.168.1.101 and http，wireshark 开始捕获分组后打开小站托福 http://www.zhan.com/。页面加载完毕后，停止捕获，可以获得下面的信息。

实验期间为了捕获到想要的结果，换了几次网站。



二、问题解答

1.分析其中一条请求报文

1.1 浏览器可以接受哪些类型的文件？可以接受哪些编码的文件？

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: www.zhan.com\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 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\r\n
```

①分析第一条 HTTP GET 的请求报文，在 HTTP 的详细分析中可以找到 accept 的内容，其中包含了**浏览器（客户端）希望接受的文件类型**，例如：text/html,application/xhtml+xml。其中 accept 的内容由多个 MIME 格式组成，MIME 即形如 type/subtype 的结构体，type 是数据类型属于总体的类别，subtype 是实际具体数据类型。所以浏览器可以接受的类型有：html,xhtml,xml,webp,apng 等。

②accept-encoding 是浏览器发给服务器的，**声明了浏览器支持的编码类型**。例如本次 GET 请求报文中，可以接受的编码文件就有：gzip,deflate

1.2 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？

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: www.zhan.com\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 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\r\n
  [truncated]Cookie: acw_tc=2f624a7615718282989736060e7c611270980e1cd6604b48db053cf1429edd; _ga=GA1.2.1961423480.1571828302; NPS_2c156
  \r\n
  [Full request URI: http://www.zhan.com/]
  [HTTP request 1/2]
  [Response in frame: 454]
  [Next request in frame: 1224]
```

HTTP 请求报文中一般有三类头部字段，通用头部、请求头部、实体头部。图中从 Host 开始均为头部，例如 Host/Connection/Upgrade-Insecure-Requests/User-agent/Accept/Accept-Encoding/Accept-Language/Cookie。
Host：客户机通过 Host 告诉服务器想访问的主机名。
Connection：处理完本次请求后是否保持连接。此处为 keep-alive 所以将持久连接。
Upgrade-Insecure-Requests：告诉服务器，浏览器可以处理 https 协议。
User-Agent：包含发出请求的用户信息。
Accept：浏览器可以接受的 MIME 类型。
Accept-Encoding：浏览器能够进行解码的数据编码方式。
Accept-Language：浏览器所希望的语言种类。
Cookie：客户机可以通过这个头向服务器带数据。

2.分析其中一条响应报文

2.1 服务器返回了什么类型的文件？

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Server:  \r\n
  Date: Sun, 03 Nov 2019 02:09:23 GMT\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  Transfer-Encoding: chunked\r\n
  Connection: keep-alive\r\n
  Vary: Accept-Encoding\r\n
  X-Powered-By: PHP/7.1.20\r\n
  Set-Cookie: PHPSESSID=tvcnp4oevkl7f5300vlqe33mjs; expires=Sun, 03-Nov-2019 14:09:23 GMT; Max-Age=43200; path=/; domain=.zhan.com\r\n
  Pragma: no-cache\r\n
  Set-Cookie: _csrf=128e22b79a13484fa1fea10d29edcf078d26ebe73b3805acdfd2a6766274e0e1a%3A2%3A%7Bi%3A0%3Bs%3A5%3A%22_csrf%22%3Bi%3A1%3Bs%
  \r\n
```

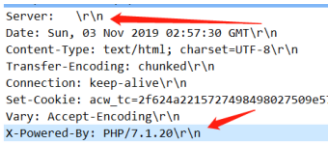
根据响应报文中的头部内容 Content-Type 可知，本次报文返回的类型是 text/html 的 MIME 类型文件。Content-Type 表示的内容就是后面的文档属于什么类型的 MIME。

2.2 服务器返回了多少字节的内容？

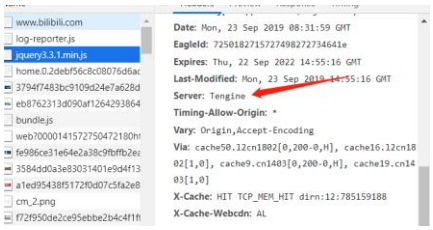
```
  HTTP chunked response
  File Data: 84958 bytes
Line-based text data: text/html (1426 lines)
```

这次响应返回的数据是 HTML 代码段，一共有 84958 字节。内容即 html 代码。

2.3 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以及语言？



根据 server，服务器通过这个头告诉浏览器服务器的类型。Server 响应头包含处理请求的原始服务器的软件信息。
但是抓包出来的 server 是三个空格，所以换成 Chorme 中的 network 中查看请求报文（随意）

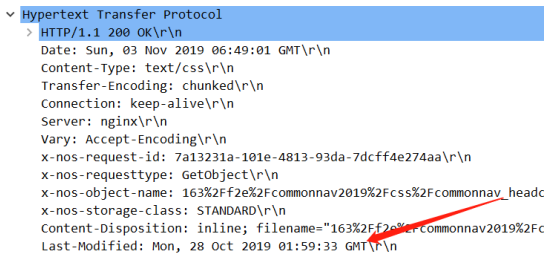


这是请求哔哩哔哩网站中某个 js 信息的 response，根据 server 可知服务器使用的 Tengine 的 web 服务器。

3.开启 Wireshark，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

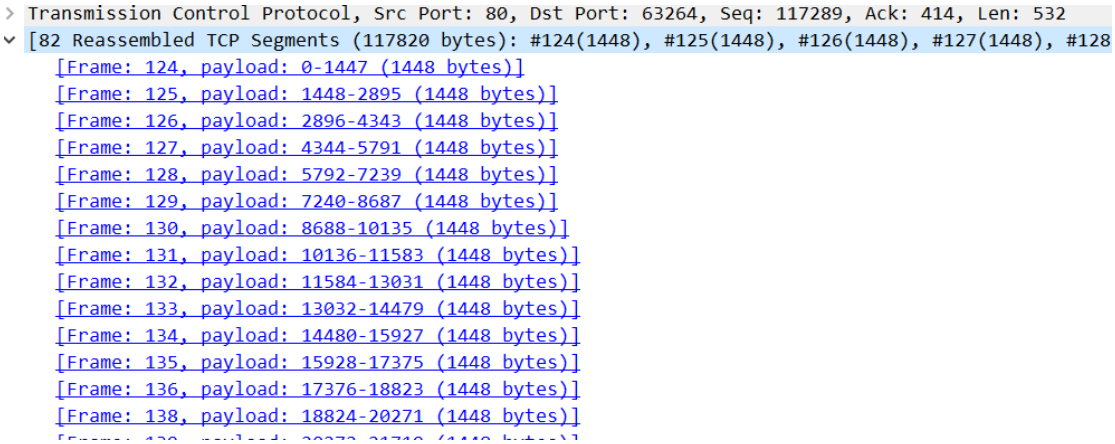
3.1 获取的页面的最后修改时间（last-modified）是什么时候？

抓包 www.163.com 网易新闻的网站。



Last-Modified 是服务端传输给客户端，表明最后一个修改的时间，主要用于判断本地是否有缓存。比如图中是发送的 css，最后修改时间是 2019/10/28 1:59:33。

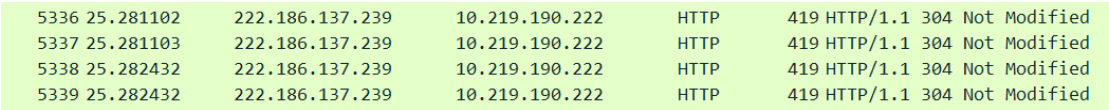
3.2 分析显示信息为 Continuation or non-HTTP traffic 的数据包，此种类型的数据包是什么含义？其最大有效载荷为多少字节？为什么？



当 HTTP 发送的数据大小超过了一个 tcp 数据包的 MSS，就需要分几个包多次传输，所以对应的包就标记为"HTTP Continuation or non-HTTP traffic"。MSS 即最大分段大小。

最大有效载荷为 1448 字节。MSS 为 TCP 数据包每次能传输的最大量，为了达到最佳一般会先双方协商，而由于在以太网最大数据帧是 1518 字节，去掉 14 字节帧头和 4 字节帧尾和 TCP 包头的 12 字节时间戳、12 字节 TCP 头、20 字节 IP 头，就剩下 1448 字节。

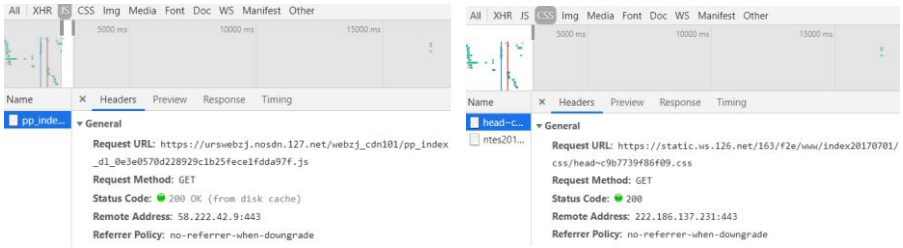
3.3 分析 HTTP 状态码为 304 的数据包，解释其含义。

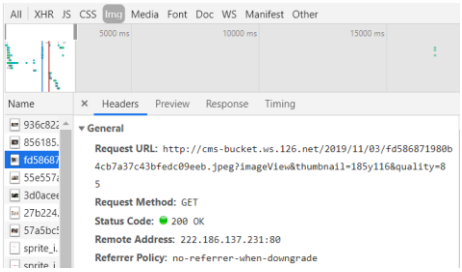


304 的状态码意思是重定向中的没有修改。如果客户端在请求一个文件的时候，发现自己的缓存文件中有 last modified，则在请求中就会包含 if modified since，并且传输的时间就是 last modified，服务端可以根据修改时间判断返回 304 还是 200，如果没有任何修改就返回 304（未修改）。

3.4 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？

不是来自同一个服务器。





如图，分别是 js/css/img 文件，分别来自不同的服务器，因为请求的 URL 不同，remote address 明显不同。这样做的好处是，可以同时与多个服务器建立连接同时传输数据，使得数据传输、页面加载效率更高。利用浏览器的控制台调试工具（开发者工具，快捷键 F12,）的 Network 页面帮助理解上一小问。

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和密码

4.1 服务器对最初的 HTTP 消息的响应（状态码和短语）是什么？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？

```
Hypertext Transfer Protocol
  HTTP/1.1 401 Unauthorized\r\n
  Date: Sun, 03 Nov 2019 06:04:45 GMT\r\n
  Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3\r\n
  WWW-Authenticate: Basic realm="wireshark-students only"\r\n
  Content-Length: 381\r\n
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=iso-8859-1\r\n
  \r\n
```

最初的 HTTP 消息响应，状态码是 401，短语是 Unauthorized，这是响应报文中的 client error，表示未授权。

相比前面，新增了 WWW-Authenticate:的字段，该字段的作用是简单有效地验证用户身份，用来完成对客户端请求的数据的合法性进行验证。缺点是，这种认证方式传输过程中是明码传输，采用的用户名密码加密方式是 BASE-64，破解方式极其简单。所以导致不安全。

当客户端请求服务器某个数据时，GET 请求报文中如果没有 Authorization 的字段，则服务端会使用 www-authenticate 要求客户端发送用户密码，客户端收到该字段的时候，会弹出一个会话框，要求用户输入验证信息。

4.2 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？

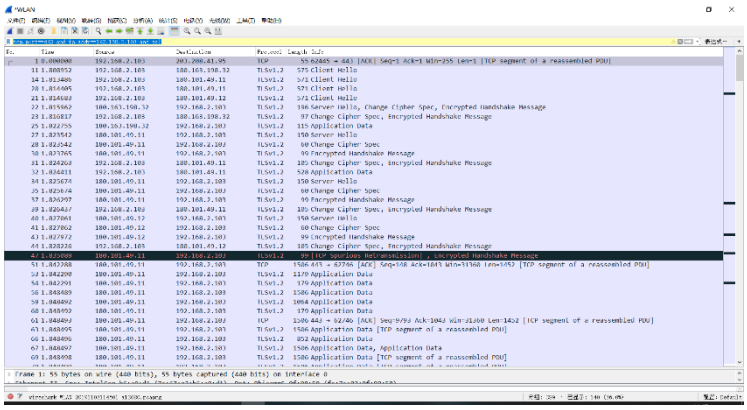
第二次发送 GET 的时候，新增了 Authorization 的字段信息，并且里面明文保存了发送的用户名和密码（123/abc）用于给服务器验证是否为可操作的权限用户。

```
Transmission Control Protocol, Src Port: 56894, Dst Port: 80, Seq: 1, Ack: 1, Len: 510
Hypertext Transfer Protocol
  GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1\r\n
  Host: gaia.cs.umass.edu\r\n
  Connection: keep-alive\r\n
  Authorization: Basic MTIzOmFiYw==\r\n
    Credentials: 123:abc
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.377
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/si
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: zh-CN,zh;q=0.9\r\n
```

5.访问 https://www.baidu.com/ ，是否能在 Wireshark 中正常捕获 http 数据包？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。

使用 ip.addr==192.168.2.103 and http 抓包百度网站，发现并不能捕获 http 数据包。

使用 tcp.port==443 and ip.addr==192.168.2.103 and ssl 再次抓包百度网站，出现以下结果。



查资料介绍:

HTTPS 是在 HTTP 的基础上和 ssl/tls 证书结合起来的一种协议,保证了传输过程中的安全性,减少了被恶意劫持的可能.很好的解决了解决了 http 的三个缺点（被监听、被篡改、被伪装）

http 和 https 都是建立在连接的基础上传输数据。http 建立连接后，客户端就请求响应，服务器回复响应。但是由于 http 不加密，容易被监听、篡改和伪装。如果服务端配置了对应的安全证书，可以使用 https，保证更加的安全。客户端发送请求到服务端后，服务端返回公钥和证书到客户端，客户端接收后先验证证书的安全性，通过后随机产生随机数，用公钥加密并发送。服务端接收后用私钥对其解密得到的随机数，并且使用这个随机数当作私钥对要发送的数据进行对称加密，此时客户端就可以使用随机数作为私钥进行解密并且解析数据呈现给客户。这样 SSL 加密就建立成功了。

tcp.port==443 and ip.addr==10.219.190.222 and ssl

o.	Time	Source	Destination	Protocol	Length	Info
15	0.502099	10.219.190.222	180.101.49.11	TLSv1.2	571	Client Hello
16	0.502455	10.219.190.222	180.101.49.11	TLSv1.2	571	Client Hello
18	0.518448	180.101.49.11	10.219.190.222	TLSv1.2	150	Server Hello
19	0.518449	180.101.49.11	10.219.190.222	TLSv1.2	60	Change Cipher Spec
20	0.518449	180.101.49.11	10.219.190.222	TLSv1.2	99	Encrypted Handshake Message
22	0.518450	180.101.49.11	10.219.190.222	TLSv1.2	150	Server Hello
23	0.518450	180.101.49.11	10.219.190.222	TLSv1.2	60	Change Cipher Spec
24	0.518450	180.101.49.11	10.219.190.222	TLSv1.2	99	Encrypted Handshake Message
27	0.519558	10.219.190.222	180.101.49.11	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
28	0.520459	10.219.190.222	180.101.49.11	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
29	0.520648	10.219.190.222	180.101.49.11	TLSv1.2	937	Application Data

具体时序过程:

1. client->server
   client hello 随机数，支持的加密算法
2. server->client
   server hello 随机数，选择一个 client 支持的加密算法
   server 证书
3. client
   验证证书
4. client->server
   client key exchange 用 server 公钥加密 pre-master key
   change cipher spec 通知 server 参数协商完成
   Encrypted Handshake message 结束握手
5. server->client
   cipher spec + finish handshake

三、实验感受

之前上课的时候，只是讲解了 http 的很多理论知识，听的懵懵懂懂，现在抓包之后发现原来现实生活中这些理论知识是这样被应用的，所以觉得还是蛮神奇的。通过实验显著增加了对网络 http 的理解，明白了 http 的请求和响应等知识。