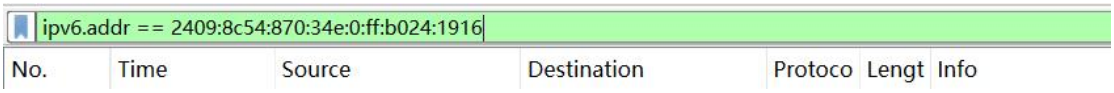
为了分析 http 和 https 交互过程，我决定使用 curl 命令与百度服务器交互

为了在众多网络包中过滤出我们要分析的包，我首先通过 ping 命令获取了 www.baidu.com 对应的 ip 地址

```
PS C:\Users\82082\Desktop> ping www.baidu.com

正在 Ping www.a.shifen.com [2409:8c54:870:34e:0:ff:b024:1916] 具有 32 字节的数据：
来自 2409:8c54:870:34e:0:ff:b024:1916 的回复： 时间=26ms
来自 2409:8c54:870:34e:0:ff:b024:1916 的回复： 时间=26ms
```

观察可知这是一个 ipv6 的地址
之后就可以使用



进行过滤

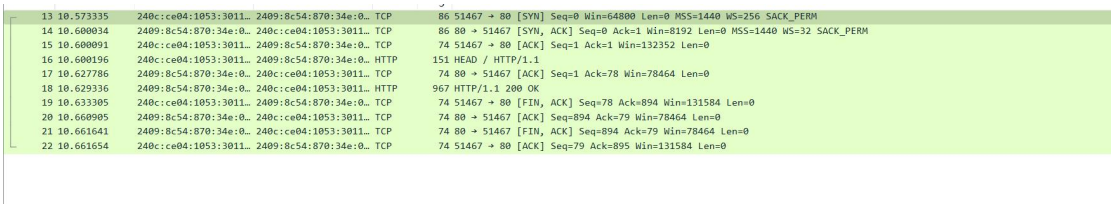
1.HTTP

运行命令 curl -I http://www.baidu.com

```
PS C:\Users\82082\Desktop> curl -I http://www.baidu.com
HTTP/1.1 200 OK
Bdpagetype: 1
Bdqid: 0xf2e8b40b008a54aa
Connection: keep-alive
Content-Length: 469025
Content-Type: text/html; charset=utf-8
Date: Sat, 30 Nov 2024 07:19:58 GMT
Server: BWS/1.1
Set-Cookie: BIDUPSID=1A803500A2D116818C743F5AEE748BCA; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: PSTM=1732951198; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BDSVRTM=2; path=/
Set-Cookie: BD_HOME=1; path=/
Set-Cookie: BAIDUID=1A803500A2D116818C743F5AEE748BCA:FG=1; Path=/; Domain=baidu.com; Max-Age=31536000
Set-Cookie: BAIDUID_BFESS=1A803500A2D116818C743F5AEE748BCA:FG=1; Path=/; Domain=baidu.com; Max-Age=31536000; Secure; SameSite=None
Traceid: 1732951198395412890617503437911122138282
Vary: Accept-Encoding
X-Ua-Compatible: IE=Edge,chrome=1
X-Xss-Protection: 1;mode=block

PS C:\Users\82082\Desktop>
```

发现已经抓取到了想要的网络包



通过所学知识分析这些网络包
前三个是 TCP 协议，可以知道这是在进行三次握手建立 TCP 连接
首先我向服务器发送了一个连接请求
然后服务器返回一个连接接受连接的应答
最后是我确认请求
连接建立成功后，就开始发送 HTTP 请求

```
HEAD / HTTP/1.1\r\n
    Request Method: HEAD
    Request URI: /
    Request Version: HTTP/1.1
Host: www.baidu.com\r\n
User-Agent: curl/8.9.1\r\n
Accept: */*\r\n
\r\n
```

使用 HEAD 方法请求获取 baidu.com 的 HTTP 头部信息，而不是下载整个网页的内容
后面是 http 协议的版本号目的地址和 url

Host：目标主机

User-Agent：浏览器的类型。我用的不是浏览器，所以这里显示的是命令 curl

之后服务器返回了应答

Response Version：响应版本，因为使用的是 HTTP 协议，所以这里显示了 HTTP 的版本

Status Code：响应状态码，这里的 200 表示请求成功。

Response Phrase：响应状态码的提示信息

Date：服务端发送响应报文的时间

Server：服务器和相对应的版本

Accept-Ranges：支持的范围单位

Content-Length：内容长度

Content-Type：资源文件类型

再下面是一些 cookie 等具体的信息

最后进行四次挥手断开连接

我想服务器发送释放连接请求

服务器确认请求

服务器向我发送释放连接请求

我确认请求

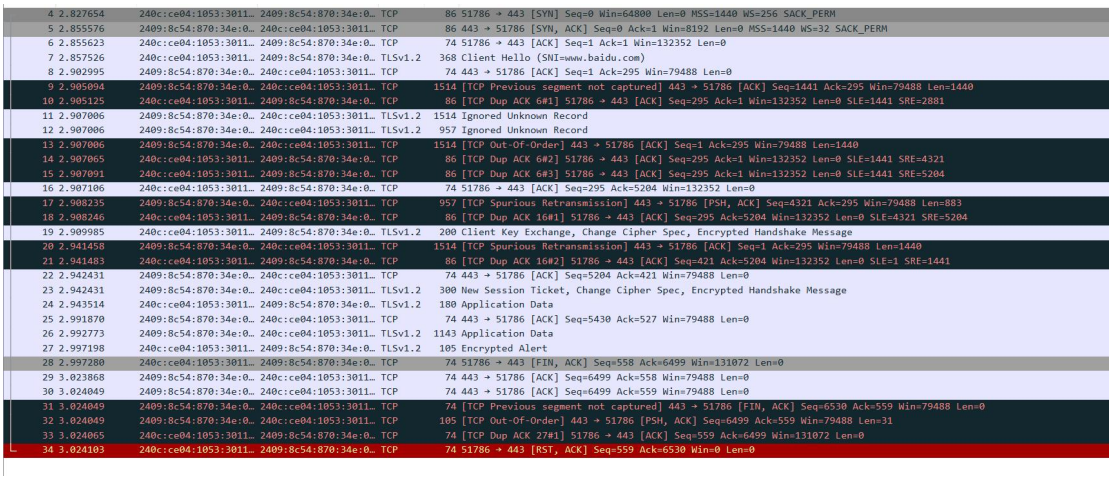
2.HTTPS

运行命令 curl -I https://www.baidu.com

```
PS C:\Users\82082\Desktop> curl -I https://www.baidu.com
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Connection: keep-alive
Content-Length: 227
Content-Security-Policy: frame-ancestors 'self' https://chat.baidu.com http://mirror-chat.baidu.com https://fj-chat.baidu.com https://hba
-chat.baidu.com https://hbe-chat.baidu.com https://njjs-chat.baidu.com https://nj-chat.baidu.com https://hna-chat.baidu.com https://hnb-c
hat.baidu.com http://debug.baidu-int.com;
Content-Type: text/html
Date: Sat, 30 Nov 2024 07:39:37 GMT
Pragma: no-cache
Server: BWS/1.1
Set-Cookie: BD_NOT_HTTPS=1; path=/; Max-Age=300
Set-Cookie: PSTM=1732952377; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BAIDUID=21F1ABB9816F364585160C1A26851641:FG=1; Path=/; Domain=baidu.com; Max-Age=31536000
Set-Cookie: BAIDUID_BFESS=21F1ABB9816F364585160C1A26851641:FG=1; Path=/; Domain=baidu.com; Max-Age=31536000; Secure; SameSite=None
Traceid: 173295237739205744741361009400339466547 8
X-Ua-Compatible: IE=Edge,chrome=1
X-Xss-Protection: 1;mode=block
```

这是抓取到的包

让我们忽略黑色和一些错误的包信息，他们对我们的分析没有贡献



4	2.827654	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	86	51786 → 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS=256 SACK_PERM
5	2.855576	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	86	443 → 51786 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=32 SACK_PERM
6	2.855623	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	74	51786 → 443 [ACK] Seq=1 Ack=1 Win=132352 Len=0
7	2.857526	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TLSv1.2	368	Client Hello (SNI=www.baidu.com)
8	2.902995	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	74	443 → 51786 [ACK] Seq=1 Ack=295 Win=79488 Len=0
9	2.905094	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	1514	[TCP Previous segment not captured] 443 → 51786 [ACK] Seq=1441 Ack=295 Win=79488 Len=1440
10	2.905125	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	86	[TCP Dup ACK 6#1] 51786 → 443 [ACK] Seq=295 Ack=1 Win=132352 Len=0 SLE=1441 SRE=2881
11	2.907006	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TLSv1.2	1514	Ignored Unknown Record
12	2.907006	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TLSv1.2	957	Ignored Unknown Record
13	2.907006	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	1514	[TCP Out-Of-Order] 443 → 51786 [ACK] Seq=1 Ack=295 Win=79488 Len=1440
14	2.907065	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	86	[TCP Dup ACK 6#2] 51786 → 443 [ACK] Seq=295 Ack=1 Win=132352 Len=0 SLE=1441 SRE=4321
15	2.907091	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	86	[TCP Dup ACK 6#3] 51786 → 443 [ACK] Seq=295 Ack=1 Win=132352 Len=0 SLE=1441 SRE=5204
16	2.907106	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	74	51786 → 443 [ACK] Seq=295 Ack=5204 Win=132352 Len=0
17	2.908235	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	957	[TCP Spurious Retransmission] 443 → 51786 [PSH, ACK] Seq=4321 Ack=295 Win=79488 Len=883
18	2.908246	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	86	[TCP Dup ACK 16#1] 51786 → 443 [ACK] Seq=295 Ack=5204 Win=132352 Len=0 SLE=4321 SRE=5204
19	2.909985	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TLSv1.2	200	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
20	2.941458	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	1514	[TCP Spurious Retransmission] 443 → 51786 [ACK] Seq=1 Ack=295 Win=79488 Len=1440
21	2.941483	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	86	[TCP Dup ACK 16#2] 51786 → 443 [ACK] Seq=421 Ack=5204 Win=132352 Len=0 SLE=1 SRE=1441
22	2.942431	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	74	443 → 51786 [ACK] Seq=5204 Ack=421 Win=79488 Len=0
23	2.942431	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TLSv1.2	300	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
24	2.943514	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TLSv1.2	180	Application Data
25	2.991870	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	74	443 → 51786 [ACK] Seq=5430 Ack=527 Win=79488 Len=0
26	2.992773	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TLSv1.2	1143	Application Data
27	2.997198	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TLSv1.2	105	Encrypted Alert
28	2.997280	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	74	51786 → 443 [FIN, ACK] Seq=558 Ack=6499 Win=131072 Len=0
29	3.023868	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	74	443 → 51786 [ACK] Seq=6499 Ack=558 Win=79488 Len=0
30	3.024049	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	74	443 → 51786 [ACK] Seq=6499 Ack=559 Win=79488 Len=0
31	3.024049	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	74	[TCP Previous segment not captured] 443 → 51786 [FIN, ACK] Seq=6530 Ack=559 Win=79488 Len=0
32	3.024049	2409:8c54:870:34e:0…	240c:ce04:1053:3011…	TCP	105	[TCP Out-Of-Order] 443 → 51786 [PSH, ACK] Seq=6499 Ack=559 Win=79488 Len=31
33	3.024065	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	74	[TCP Dup ACK 27#1] 51786 → 443 [ACK] Seq=559 Ack=6499 Win=131072 Len=0
34	3.024103	240c:ce04:1053:3011…	2409:8c54:870:34e:0…	TCP	74	51786 → 443 [RST, ACK] Seq=559 Ack=6530 Win=0 Len=0

我们发现 HTTPS 明显比 HTTP 复杂了很多

前三个和后四个依然是 TCP 三次握手四次挥手，不再赘述

建立连接后首先发送一个 Client Hello 信号，客户端发送随机数字+自己可以支持的加密方

法。

```
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 289
    Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 285
      Version: TLS 1.2 (0x0303)
      Random: 76a72cdc703a48f19faa79eb0f29f732fcee90862cf011e2f689de425a5230a2
      Session ID Length: 32
      Session ID: 0d662c87a2432d134b45dcb5391965ddd03b83d009e65ee8c65d69a0e960debd
      Cipher Suites Length: 40
      Cipher Suites (20 suites)
      Compression Methods Length: 1
      Compression Methods (1 method)
      Extensions Length: 172
```

这是支持的加密方法

```
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
```

然后是一个 ACK 应答

之后服务器发送 Server Hello 信号，服务器发送随机数字+选择双方都支持的加密方式

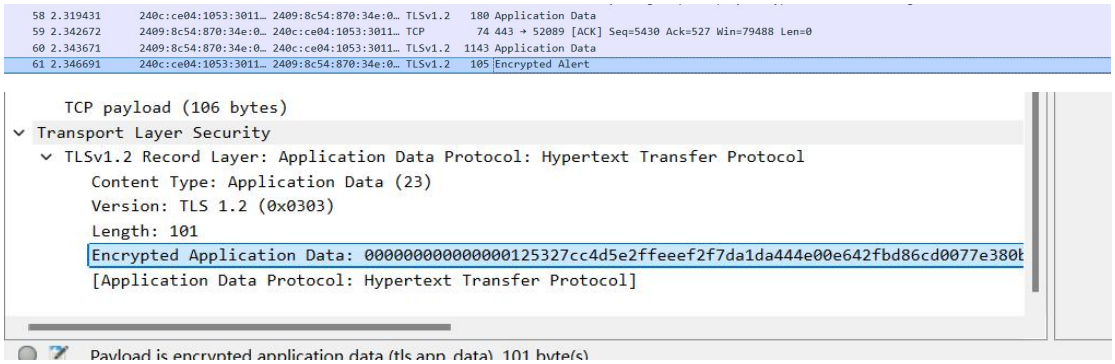
这里 server hello 显示 Ignored Unknown Record，可能是软件的 BUG，不过我们还是可以知道这两条是 server hello 信息

```
    45 2.290864    2409:8c54:870:34e:0… 240c:ce04:1053:3011… TLSv1.2  1514 Ignored Unknown Record
    46 2.290864    2409:8c54:870:34e:0… 240c:ce04:1053:3011… TLSv1.2   957 Ignored Unknown Record
```

然后是一个 ACK 应答

接下来的几条也都是为了加密发送的一些信息

最后终于到了信息的发送



https 进行了加密，无法像 http 协议一样看出发送的明文是什么

最后断开连接