ID 22031212122
NAME xiaoning Shu
TEAC zhiwei Zhang
DATE 20230523

西安电子科技大学
XIDIAN UNIVERSITY

Experimentation and practice of new generation information technology

HOMEWORK 2 Network Analysis Technology Experiment

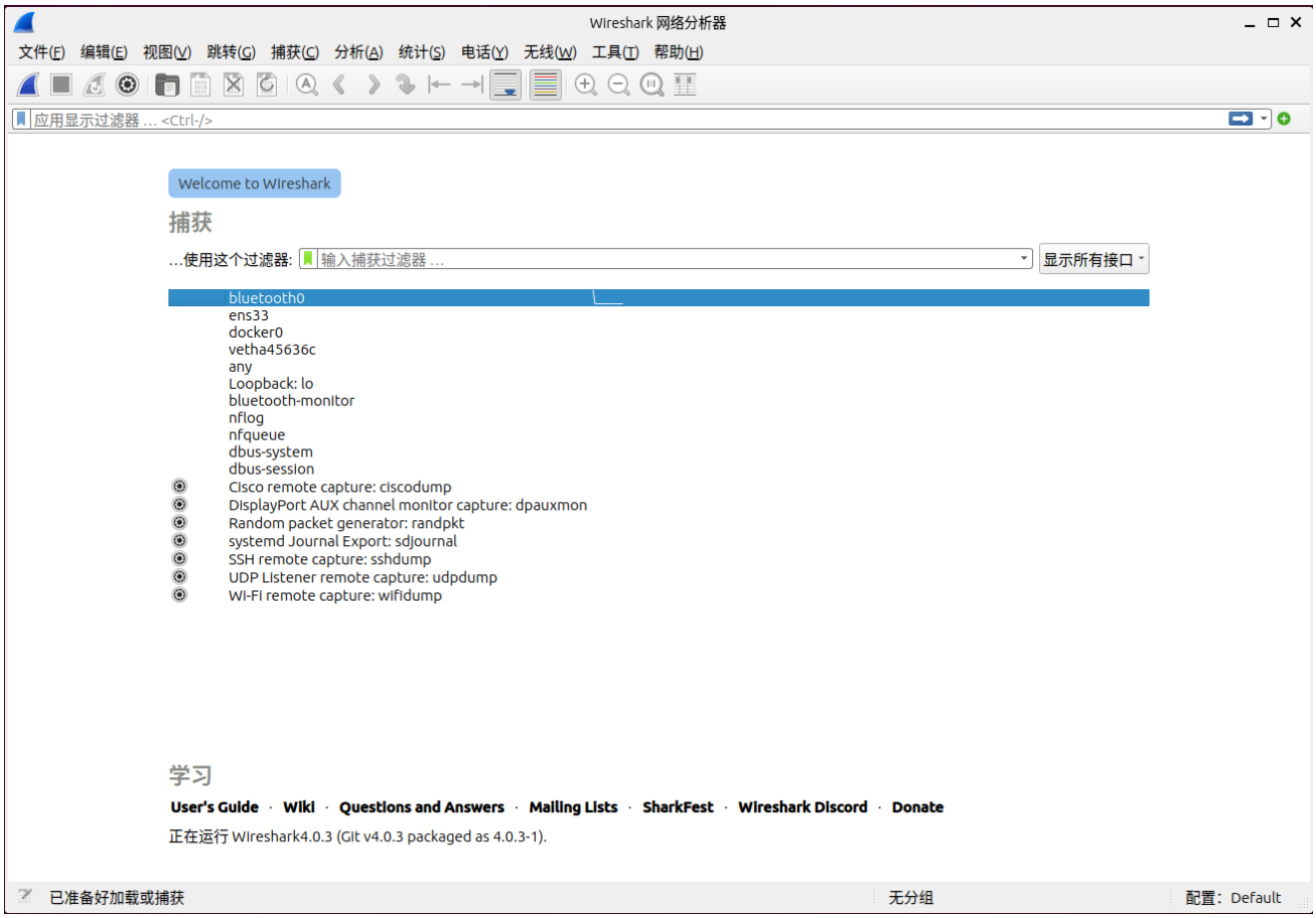
网络分析技术实验

HTTP协议分析实验

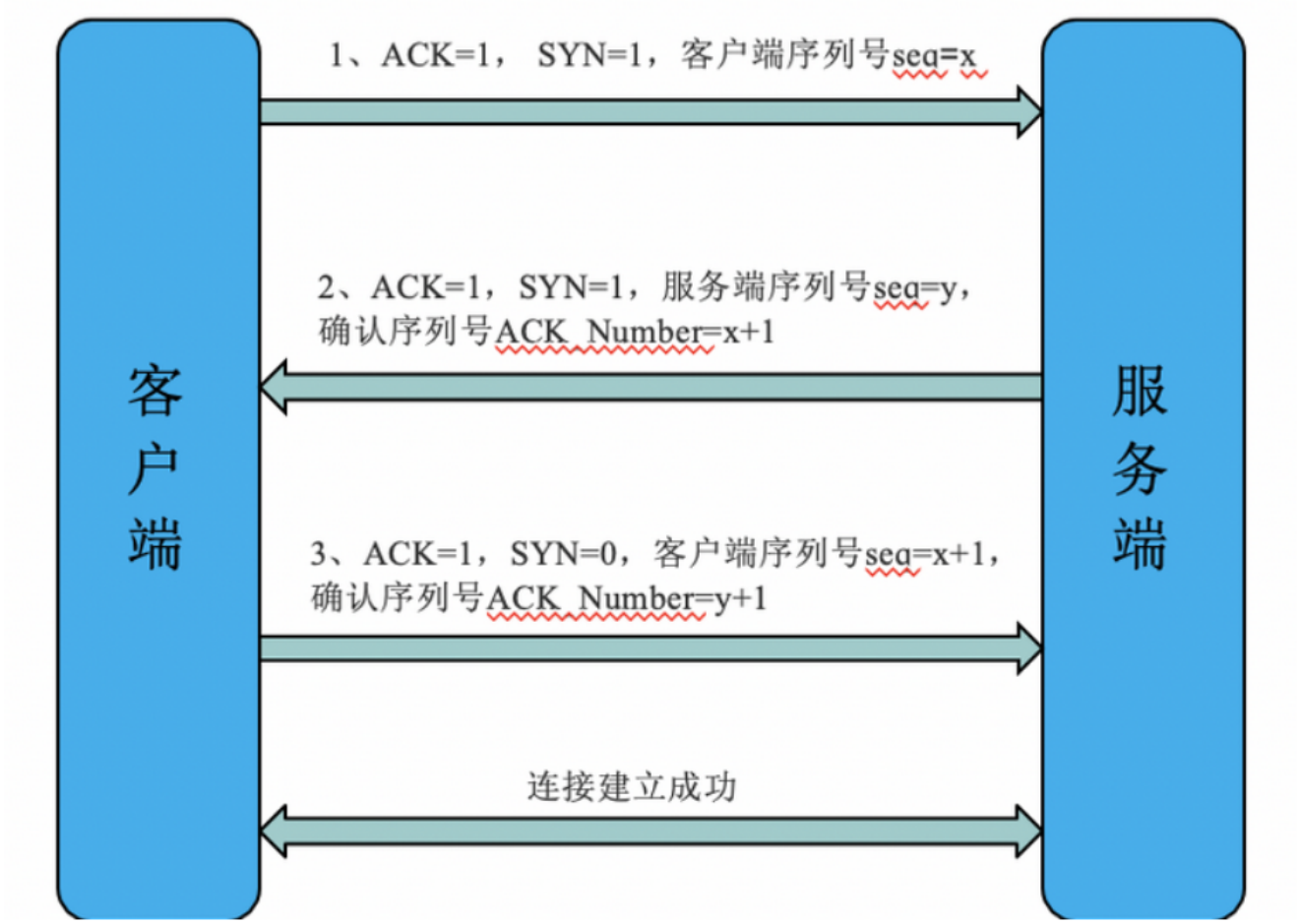
利用实验（一）建立的Web应用（虚拟机和容器两个版本），在用户侧使用网络抓包分析工具进行协议分析，观察并分析BS之间交互的过程和内容，对比虚拟机和容器版本交互内容有无异差。

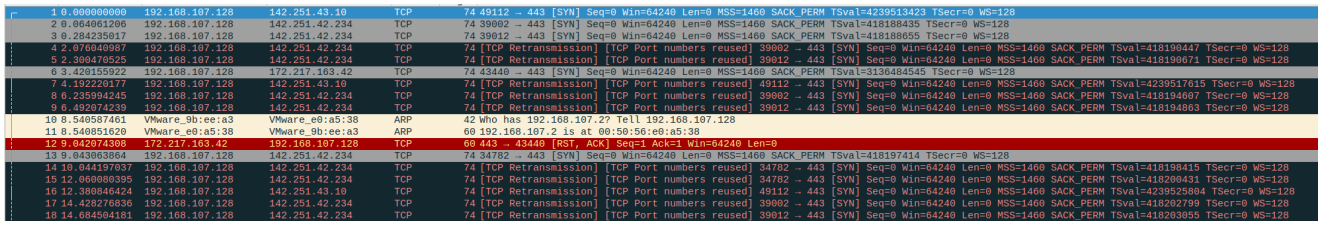
安装抓包工具Wireshark

```
sudo apt-get install wireshark
```



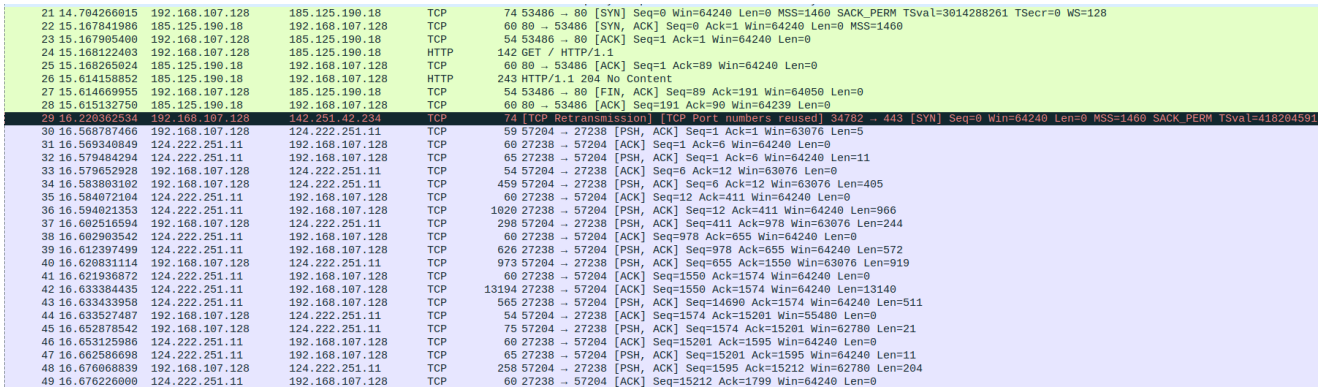
这是抓包的记录，因为我使用了云MySql服务器，所以在进行运行登录界面时，会对云MySql服务器进行握手规则。





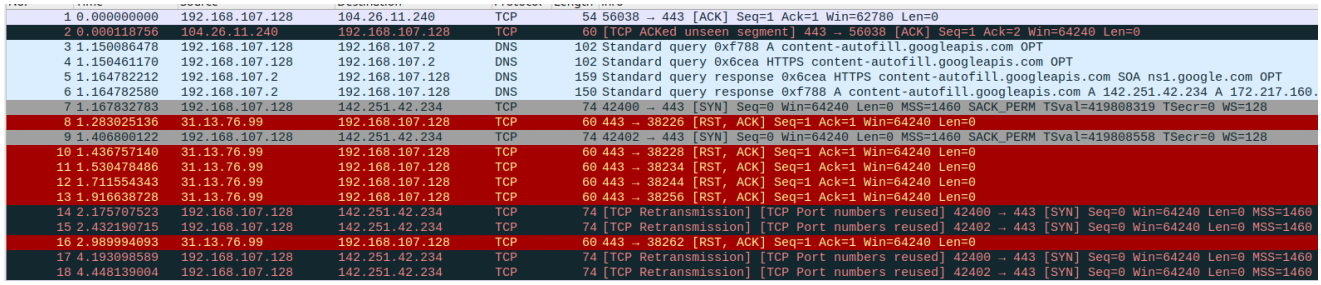
1	0.000000000	192.168.107.128	142.251.43.10	TCP	74 49112 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4239513423 TSecr=0 WS=128
2	0.064061206	192.168.107.128	142.251.42.234	TCP	74 39002 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418188435 TSecr=0 WS=128
3	0.284235017	192.168.107.128	142.251.42.234	TCP	74 39012 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418188655 TSecr=0 WS=128
4	2.076040987	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 39002 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418190447 TSecr=0 WS=128
5	2.300470525	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 39012 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418190671 TSecr=0 WS=128
6	3.420155922	192.168.107.128	172.217.163.42	TCP	74 43440 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3136484545 TSecr=0 WS=128
7	4.192220317	192.168.107.128	142.251.43.10	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 49112 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4239517615 TSecr=0 WS=128
8	0.235994245	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 39002 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418194607 TSecr=0 WS=128
9	6.492074239	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 39012 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418194863 TSecr=0 WS=128
10	8.540587461	VMware_9b:ee:a3	VMware_e0:a5:38	ARP	42 Who has 192.168.107.2? Tell 192.168.107.128
11	8.540851620	VMware_e0:a5:38	VMware_9b:ee:a3	ARP	60 192.168.107.2 is at 00:50:56:e0:a5:38
12	9.042074308	172.217.163.42	192.168.107.128	TCP	60 443 → 43440 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
13	9.043063864	192.168.107.128	142.251.42.234	TCP	74 34782 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418197414 TSecr=0 WS=128
14	10.044197037	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 34782 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418198415 TSecr=0 WS=128
15	12.060080395	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 34782 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418200431 TSecr=0 WS=128
16	12.380846424	192.168.107.128	142.251.43.10	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 49112 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4239525804 TSecr=0 WS=128
17	14.428276836	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 39002 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418202799 TSecr=0 WS=128
18	14.684504181	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 39012 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418203055 TSecr=0 WS=128

然后是我的主机像我的Tomcat服务器进行三次握手规则。后面进行了登陆之后的信息传输。



21	14.704266015	192.168.107.128	185.125.190.18	TCP	74 53486 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3014288261 TSecr=0 WS=128
22	15.167841986	185.125.190.18	192.168.107.128	TCP	60 80 → 53486 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
23	15.167905400	192.168.107.128	185.125.190.18	TCP	54 53486 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
24	15.168122403	192.168.107.128	185.125.190.18	HTTP	142 GET / HTTP/1.1
25	15.168265024	185.125.190.18	192.168.107.128	TCP	60 80 → 53486 [ACK] Seq=1 Ack=89 Win=64240 Len=0
26	15.614158852	185.125.190.18	192.168.107.128	HTTP	243 HTTP/1.1 204 No Content
27	15.614669955	192.168.107.128	185.125.190.18	TCP	54 53486 → 80 [FIN, ACK] Seq=89 Ack=191 Win=64050 Len=0
28	15.615132750	185.125.190.18	192.168.107.128	TCP	60 80 → 53486 [ACK] Seq=191 Ack=90 Win=64239 Len=0
29	16.220362534	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 34782 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=418204591
30	16.568787466	192.168.107.128	124.222.251.11	TCP	59 57204 → 27238 [PSH, ACK] Seq=1 Ack=1 Win=63076 Len=5
31	16.569340849	124.222.251.11	192.168.107.128	TCP	60 27238 → 57204 [ACK] Seq=1 Ack=6 Win=64240 Len=0
32	16.579484294	124.222.251.11	192.168.107.128	TCP	65 27238 → 57204 [PSH, ACK] Seq=1 Ack=6 Win=64240 Len=11
33	16.579652928	192.168.107.128	124.222.251.11	TCP	54 57204 → 27238 [ACK] Seq=6 Ack=12 Win=63076 Len=0
34	16.583803102	192.168.107.128	124.222.251.11	TCP	459 57204 → 27238 [PSH, ACK] Seq=6 Ack=12 Win=63076 Len=405
35	16.584072104	124.222.251.11	192.168.107.128	TCP	60 27238 → 57204 [ACK] Seq=12 Ack=411 Win=64240 Len=0
36	16.594021353	124.222.251.11	192.168.107.128	TCP	1020 27238 → 57204 [PSH, ACK] Seq=12 Ack=411 Win=64240 Len=966
37	16.602516594	192.168.107.128	124.222.251.11	TCP	298 57204 → 27238 [PSH, ACK] Seq=411 Ack=978 Win=63076 Len=244
38	16.602903542	124.222.251.11	192.168.107.128	TCP	60 27238 → 57204 [ACK] Seq=978 Ack=655 Win=64240 Len=0
39	16.612397499	124.222.251.11	192.168.107.128	TCP	626 27238 → 57204 [PSH, ACK] Seq=978 Ack=655 Win=64240 Len=572
40	16.620831114	192.168.107.128	124.222.251.11	TCP	973 57204 → 27238 [PSH, ACK] Seq=655 Ack=1550 Win=63076 Len=919
41	16.621936872	124.222.251.11	192.168.107.128	TCP	60 27238 → 57204 [ACK] Seq=1550 Ack=1574 Win=64240 Len=0
42	16.633384435	124.222.251.11	192.168.107.128	TCP	13194 27238 → 57204 [ACK] Seq=1550 Ack=1574 Win=64240 Len=13140
43	16.633433958	124.222.251.11	192.168.107.128	TCP	565 27238 → 57204 [PSH, ACK] Seq=14690 Ack=1574 Win=64240 Len=511
44	16.633527487	192.168.107.128	124.222.251.11	TCP	54 57204 → 27238 [ACK] Seq=1574 Ack=15201 Win=55480 Len=0
45	16.652878542	192.168.107.128	124.222.251.11	TCP	75 57204 → 27238 [PSH, ACK] Seq=1574 Ack=15201 Win=62780 Len=21
46	16.653125986	124.222.251.11	192.168.107.128	TCP	60 27238 → 57204 [ACK] Seq=15201 Ack=1595 Win=64240 Len=0
47	16.662586698	124.222.251.11	192.168.107.128	TCP	65 27238 → 57204 [PSH, ACK] Seq=15201 Ack=1595 Win=64240 Len=11
48	16.676068839	192.168.107.128	124.222.251.11	TCP	258 57204 → 27238 [PSH, ACK] Seq=1595 Ack=15212 Win=62780 Len=204
49	16.676226000	124.222.251.11	192.168.107.128	TCP	60 27238 → 57204 [ACK] Seq=15212 Ack=1799 Win=64240 Len=0

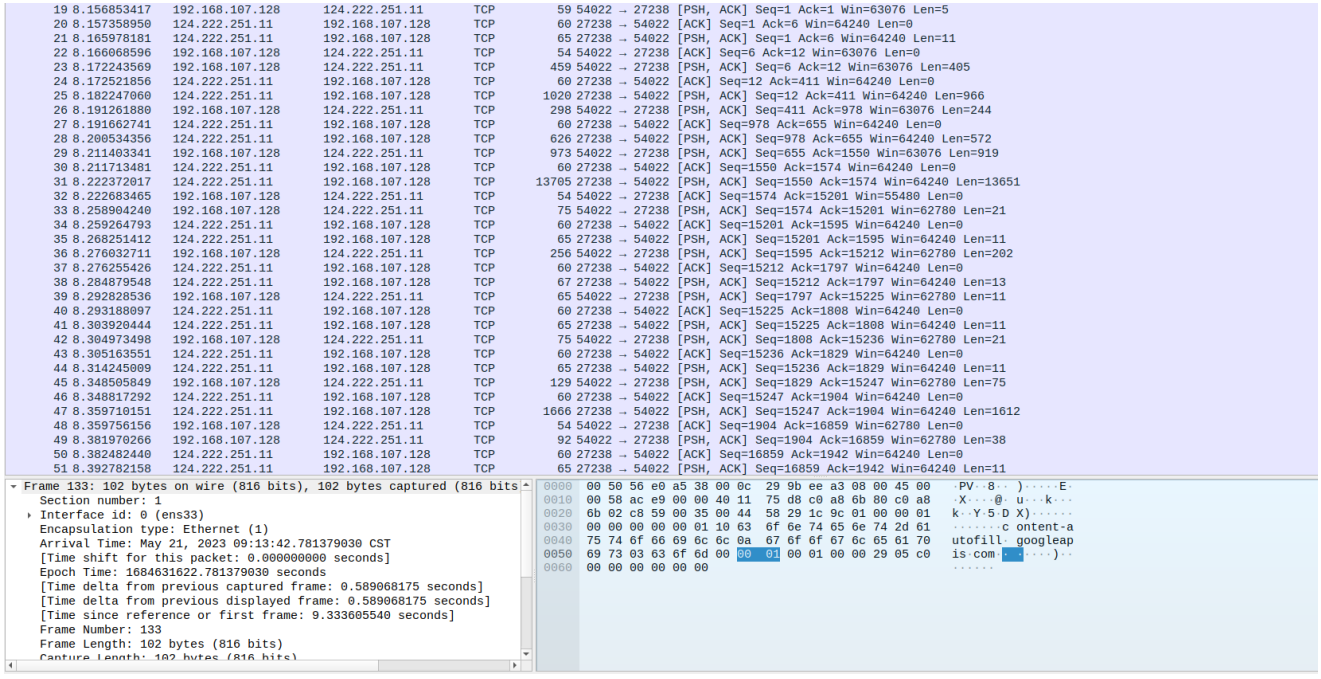
以上是8080端口的idea登录方式。以下证明docker形式的登录。



1	0.000000000	192.168.107.128	104.26.11.240	TCP	54 56038 → 443 [ACK] Seq=1 Ack=1 Win=62780 Len=0
2	0.000118756	104.26.11.240	192.168.107.128	TCP	60 [TCP ACKed unseen segment] 443 → 56038 [ACK] Seq=1 Ack=2 Win=64240 Len=0
3	1.150086478	192.168.107.128	192.168.107.2	DNS	102 Standard query 0xf788 A content-autofill.googleapis.com OPT
4	1.150461170	192.168.107.128	192.168.107.2	DNS	102 Standard query 0x6cea HTTPS content-autofill.googleapis.com OPT
5	1.164782212	192.168.107.2	192.168.107.128	DNS	159 Standard query response 0x6cea HTTPS content-autofill.googleapis.com SOA ns1.google.com OPT
6	1.164782580	192.168.107.2	192.168.107.128	DNS	150 Standard query response 0xf788 A content-autofill.googleapis.com A 142.251.42.234 A 172.217.160.
7	1.167832783	192.168.107.128	142.251.42.234	TCP	74 42400 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=419808319 TSecr=0 WS=128
8	1.283025136	31.13.76.99	192.168.107.128	TCP	60 443 → 38226 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
9	1.406800122	192.168.107.128	142.251.42.234	TCP	74 42402 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=419808558 TSecr=0 WS=128
10	1.436757140	31.13.76.99	192.168.107.128	TCP	60 443 → 38228 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
11	1.530478486	31.13.76.99	192.168.107.128	TCP	60 443 → 38234 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
12	1.711554343	31.13.76.99	192.168.107.128	TCP	60 443 → 38244 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
13	1.916638728	31.13.76.99	192.168.107.128	TCP	60 443 → 38256 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
14	2.175707523	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 42400 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
15	2.432190715	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 42402 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
16	2.989994093	31.13.76.99	192.168.107.128	TCP	60 443 → 38262 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
17	4.193098589	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 42400 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
18	4.448139004	192.168.107.128	142.251.42.234	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 42402 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460

这是docker 8060端口进行登录时的对云MySql的链接。可以看到进行了握手规则。

下面是对Tomcat服务器以及登录后渲染界面的信息传输。



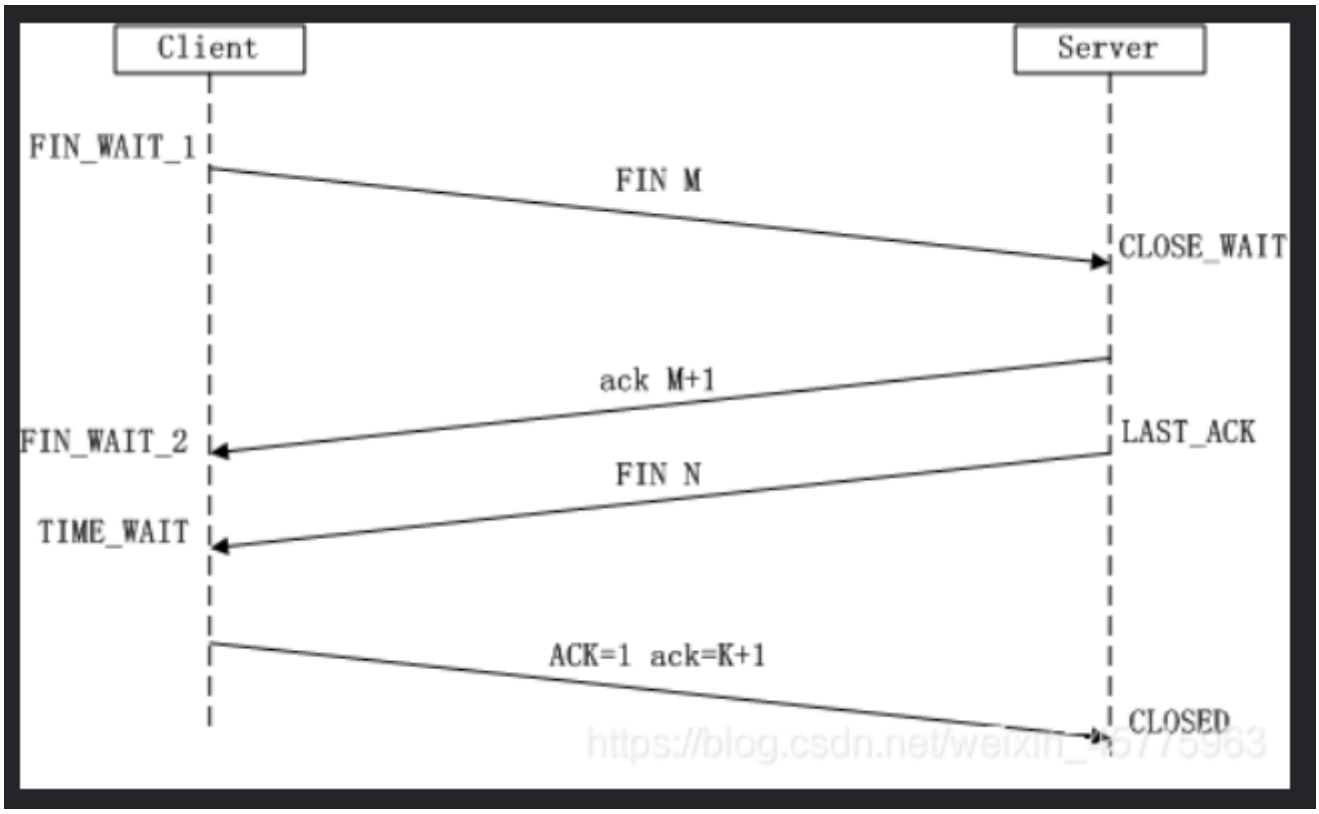
19	8.156853417	192.168.107.128	124.222.251.11	TCP	59 54022 → 27238 [PSH, ACK] Seq=1 Ack=1 Win=63076 Len=5
20	8.157358950	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=1 Ack=6 Win=64240 Len=0
21	8.165978181	124.222.251.11	192.168.107.128	TCP	65 27238 → 54022 [PSH, ACK] Seq=1 Ack=6 Win=64240 Len=11
22	8.166068596	192.168.107.128	124.222.251.11	TCP	54 54022 → 27238 [ACK] Seq=6 Ack=12 Win=63076 Len=0
23	8.172243569	192.168.107.128	124.222.251.11	TCP	459 54022 → 27238 [PSH, ACK] Seq=6 Ack=12 Win=63076 Len=405
24	8.172521856	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=12 Ack=411 Win=64240 Len=0
25	8.182247060	124.222.251.11	192.168.107.128	TCP	1020 27238 → 54022 [PSH, ACK] Seq=12 Ack=411 Win=64240 Len=966
26	8.191261880	192.168.107.128	124.222.251.11	TCP	298 54022 → 27238 [PSH, ACK] Seq=411 Ack=978 Win=63076 Len=244
27	8.191662741	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=978 Ack=655 Win=64240 Len=0
28	8.200534356	124.222.251.11	192.168.107.128	TCP	626 27238 → 54022 [PSH, ACK] Seq=978 Ack=655 Win=64240 Len=572
29	8.211403341	192.168.107.128	124.222.251.11	TCP	973 54022 → 27238 [PSH, ACK] Seq=655 Ack=1550 Win=63076 Len=919
30	8.211713481	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=1550 Ack=1574 Win=64240 Len=0
31	8.222372017	124.222.251.11	192.168.107.128	TCP	13705 27238 → 54022 [PSH, ACK] Seq=1550 Ack=1574 Win=64240 Len=13651
32	8.222683465	192.168.107.128	124.222.251.11	TCP	54 54022 → 27238 [ACK] Seq=1574 Ack=15201 Win=55480 Len=0
33	8.258904240	192.168.107.128	124.222.251.11	TCP	75 54022 → 27238 [PSH, ACK] Seq=1574 Ack=15201 Win=62780 Len=21
34	8.259264793	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=15201 Ack=1595 Win=64240 Len=0
35	8.268251412	124.222.251.11	192.168.107.128	TCP	65 27238 → 54022 [PSH, ACK] Seq=15201 Ack=1595 Win=64240 Len=11
36	8.276032711	192.168.107.128	124.222.251.11	TCP	256 54022 → 27238 [PSH, ACK] Seq=1595 Ack=15212 Win=62780 Len=202
37	8.276255426	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=15212 Ack=1797 Win=64240 Len=0
38	8.284879548	124.222.251.11	192.168.107.128	TCP	67 27238 → 54022 [PSH, ACK] Seq=15212 Ack=1797 Win=64240 Len=13
39	8.292828536	192.168.107.128	124.222.251.11	TCP	65 54022 → 27238 [PSH, ACK] Seq=1797 Ack=15225 Win=62780 Len=11
40	8.293188097	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=15225 Ack=1808 Win=64240 Len=0
41	8.303920444	124.222.251.11	192.168.107.128	TCP	65 27238 → 54022 [PSH, ACK] Seq=15225 Ack=1808 Win=64240 Len=11
42	8.304973498	192.168.107.128	124.222.251.11	TCP	75 54022 → 27238 [PSH, ACK] Seq=1808 Ack=15236 Win=62780 Len=21
43	8.305163551	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=15236 Ack=1829 Win=64240 Len=0
44	8.314245009	124.222.251.11	192.168.107.128	TCP	65 27238 → 54022 [PSH, ACK] Seq=15236 Ack=1829 Win=64240 Len=11
45	8.348505849	192.168.107.128	124.222.251.11	TCP	129 54022 → 27238 [PSH, ACK] Seq=1829 Ack=15247 Win=62780 Len=75
46	8.348817292	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=15247 Ack=1904 Win=64240 Len=0
47	8.359710151	124.222.251.11	192.168.107.128	TCP	1666 27238 → 54022 [PSH, ACK] Seq=15247 Ack=1904 Win=64240 Len=1612
48	8.359756156	192.168.107.128	124.222.251.11	TCP	54 54022 → 27238 [ACK] Seq=1904 Ack=16859 Win=62780 Len=0
49	8.381970266	192.168.107.128	124.222.251.11	TCP	92 54022 → 27238 [PSH, ACK] Seq=1904 Ack=16859 Win=62780 Len=38
50	8.382482440	124.222.251.11	192.168.107.128	TCP	60 27238 → 54022 [ACK] Seq=16859 Ack=1942 Win=64240 Len=0
51	8.392782158	124.222.251.11	192.168.107.128	TCP	65 27238 → 54022 [PSH, ACK] Seq=16859 Ack=1942 Win=64240 Len=11


```
▾ Frame 133: 102 bytes on wire (816 bits), 102 bytes captured (816 bits
    Section number: 1
  ▸ Interface id: 0 (ens33)
    Encapsulation type: Ethernet (1)
    Arrival Time: May 21, 2023 09:13:42.781379030 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1684631622.781379030 seconds
    [Time delta from previous captured frame: 0.589068175 seconds]
    [Time delta from previous displayed frame: 0.589068175 seconds]
    [Time since reference or first frame: 9.333605540 seconds]
    Frame Number: 133
    Frame Length: 102 bytes (816 bits)
    Capture Length: 102 bytes (816 bits)
```

```
0000  00 50 56 e0 a5 38 00 0c  29 9b ee a3 08 00 45 00   ·PV··8·· )·····E·
0010  00 58 ac e9 00 00 40 11  75 d8 c0 a8 6b 80 c0 a8   ·X····@· u···k···
0020  6b 02 c8 59 00 35 00 44  58 29 1c 9c 01 00 00 01   k··Y·5·D X)······
0030  00 00 00 00 00 01 10 63  6f 6e 74 65 6e 74 2d 61   ·······c ontent-a
0040  75 74 6f 66 69 6c 6c 0a  67 6f 6f 67 6c 65 61 70   utofill· googleap
0050  69 73 03 63 6f 6d 00 00  01 00 01 00 00 29 05 c0   is·com·· ·····)··
0060  00 00 00 00 00 00                                 ·····
```

以上完美的证明了连接服务器时的三次握手规则。

至于关闭软件时的四次断手规则，与其差不多，用一张图就可以表明。



内容之间的交互差异：

除了网络之外，文件也是重要的进行数据交互的资源。在以往的虚拟机中，我们通常直接采用虚拟机的文件系统作为应用数据等文件的存储位置。然而这种方式其实并非完全安全的，当虚拟机或者容器出现问题导致文件系统无法使用时，虽然我们可以很快的通过镜像重置文件系统使得应用快速恢复运行，但是之前存放的数据也就消失了。

为了保证数据的独立性，我们通常会单独挂载一个文件系统来存放数据。这种操作在虚拟机中是繁琐的，因为我们不但要搞定挂载在不同宿主机中实现的方法，还要考虑挂载文件系统兼容性，虚拟操作系统配置等问题。值得庆幸的是，这些在 Docker 里都已经为我们轻松的实现了，我们只需要简单的一两个命令或参数，就能完成文件系统目录的挂载。

能够这么简单的实现挂载，主要还是得益于 Docker 底层的 Union File System 技术。在 UnionFS 的加持下，除了能够从宿主操作系统中挂载目录外，还能够建立独立的目录持久存放数据，或者在容器间共享。

在 Docker 中，通过这几种方式进行数据共享或持久化的文件或目录，我们都称为数据卷（Volume）。

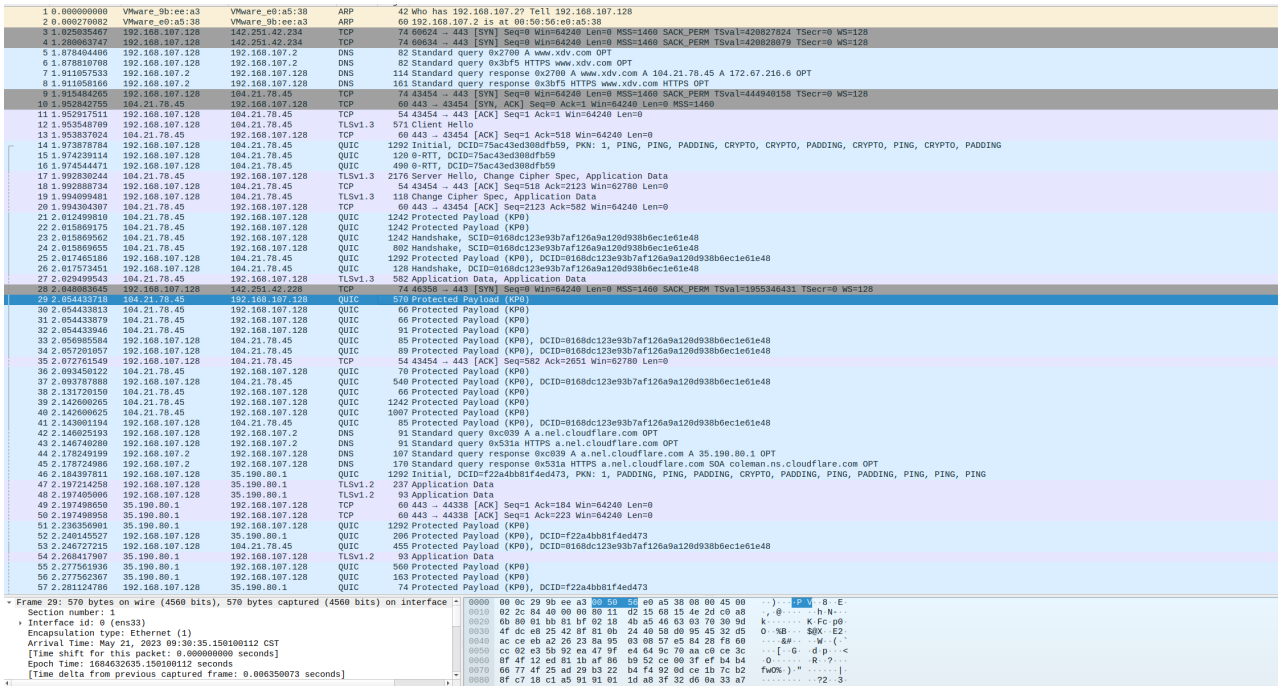
通过挂载数据卷的方式，我们可以让宿主机的文件数据和容器中的文件数据进行交互，可以得知容器内部发生的情况，还可以通过容器之间继承数据卷的方式，实现容器集群共享数据。

HTTPS协议分析实验

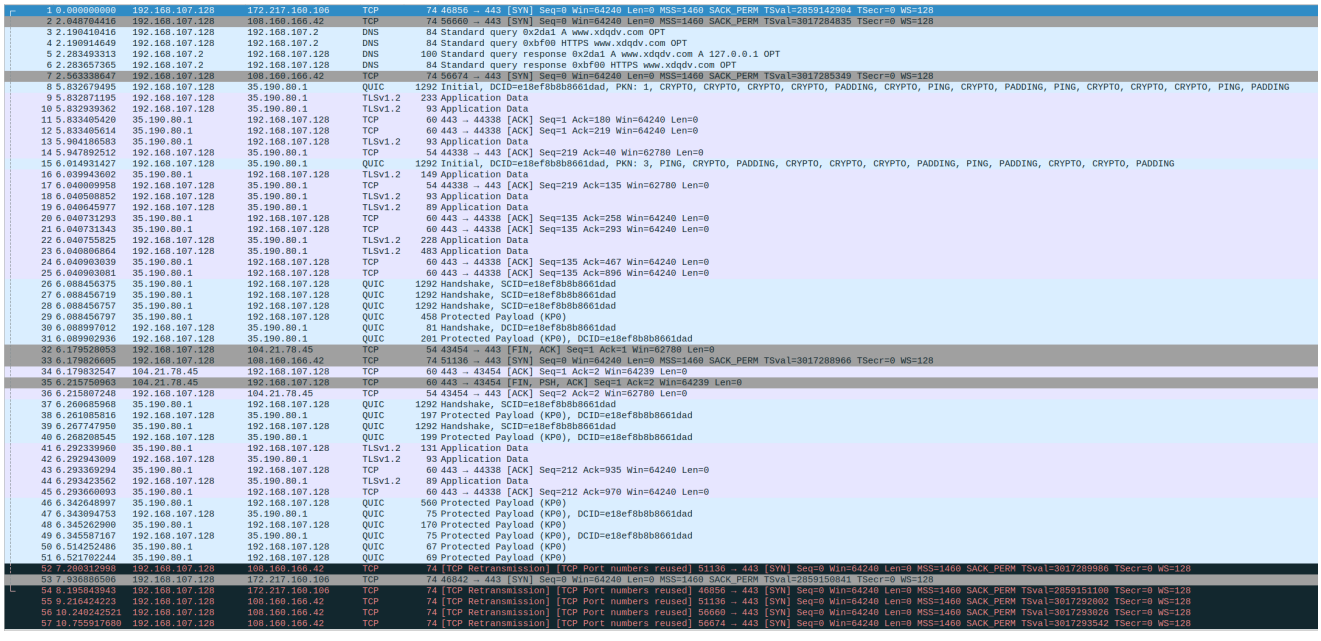
- **在HTTP实验选择的Web应用版本基础上（虚拟机和容器两个版本），开启HTTPS服务；在用户侧使用网络抓包分析工具进行协议分析，观察并分析BS之间交互的过程和内容，对比虚拟机和容器版本交互内容有无异差。**

我在上海，仔细看了看网址，好像需要登录VPN，要不打不开网页。

这是HTTP协议抓的包：



这是在HTTPS上抓的包：



我似乎登录不上HTTPS的网络。但并不代表我不能看清这些东西。HTTPS似乎是比HTTP传输的更加隐秘，不再是明文传输，而是进行了加密规则。更有利于保护用户的隐私

- **比较分析HTTP和HTTPS协议的差异**

  TTP协议以明文方式发送内容，不提供任何方式的数据加密。HTTP协议不适合传输一些敏感信息，比如：信用卡号、密码等支付信息。https则是具有安全性的ssl加密传输协议。http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。并且https协议需要到ca申请证书。HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。

  HTTPS协议的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全;另一种就是确认网站的真实性。HTTPS在HTTP的基础上加入了SSL协议，SSL依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。

实验说明

- 域名为http://www.xdv.com、http://www.xdc.com和https://www.xdqdv.com、https://xdqdc.com
- 网络抓包分析工具不限，公钥证书等工具不限
- 分析范围为用户发去页面请求到用户登录成功或失败期间，浏览器与服务器之间所有的网络交互，至少应包括应用层、传输层和网络层的协议内容。