

实验 1 万维网运行原理分析

1.1 实验介绍

本实验通过对特定网站分析，了解网站运行原理和相关技术；
通过使用抓包工具或浏览器自带工具采集 HTTP 协议包并进行分析。

1.2 实验目标

深入了解万维网结构、原理、技术
深入了解并掌握 WEB 页面组成
深入了解并掌握 HTTP 协议

1.3 实验原理与方法

1.3.1 万维网（www）运行原理

浏览器与 WEB 服务器建立 TCP 连接
浏览器发出 HTTP 请求
WEB 浏览器解析 HTTP 请求，发回 HTTP 响应
浏览器解析 HTTP 响应
TCP 连接关闭

1.3.2 服务器

IIS, Tomcat, Weblogic, jboss

1.3.3 浏览器

IE, Firefox, Google Chrome, Apple Safari

1.3.4 HTTP 协议

简介

Hyper Text Transfer Protocol，超文本传输协议

特点

处理任何类型的数据，MIME-type 数据类型，Multipurpose Internet Mail Extensions，描述消息内容类型的因特网标准

无状态

HTTP 请求报文

请求行：

请求方法，GET 请求数据，POST 提交数据，HEAD 获取报头，PUT 取代指定文档，DELETE 删除页面，CONNECT 管道方式，OPTIONS 查看性能，TRACE 测试诊断

URL 地址

协议名称版本号

请求头：

属性名：属性值

Accept，一个或多个 MIME 类型的值

Cookie，无状态，隶属于一个 Session（会话），jsessionid

Refer，请求来源

Cache-Control，缓存控制

请求体:

Param1=value1¶m2=value2, 多个请求参数数据

请求 URL

HTTP 响应报文

响应行:

报文协议及版本

状态码及状态描述

1xx 通知客户端请求已收到

2xx 处理成功

200 OK 请求成功

3xx 重定向, 让客户端再发起请求

303 See Other 查看其他地址

304 Not Modified 可读取本地缓存

4xx 客户端异常

403 Forbidden 拒绝执行客户端请求

404 Not Found 无法找到客户端请求

5xx 服务器异常

响应头:

属性名: 属性值

Cache-Control, 客户端如何控制缓存, 31536000 秒 (365 天)

Private 客户端缓存

Public 客户端和代理服务器缓存

Max-age=xxx xxx 秒失效

No-cache 对比缓存验证数据

No-store 不缓存

Etag, 服务端资源发生变换

Location, JSP (JavaServer Pages), Redirect (重定向), 让客户端再发送请求

Set-Cookie, 设置客户端会话 Cookie

Max-age, 有效期, 单位秒, 正数有效时间, 负数本窗口及子窗口有效, 0 删除 Cookie

GetMaxage() SetMaxage(int Maxage)

修改 Cookie, 新建同名 Cookie 添加到 response 中覆盖原 Cookie

响应体:

响应报文体

1.4 实验步骤

1.4.1 选择网站

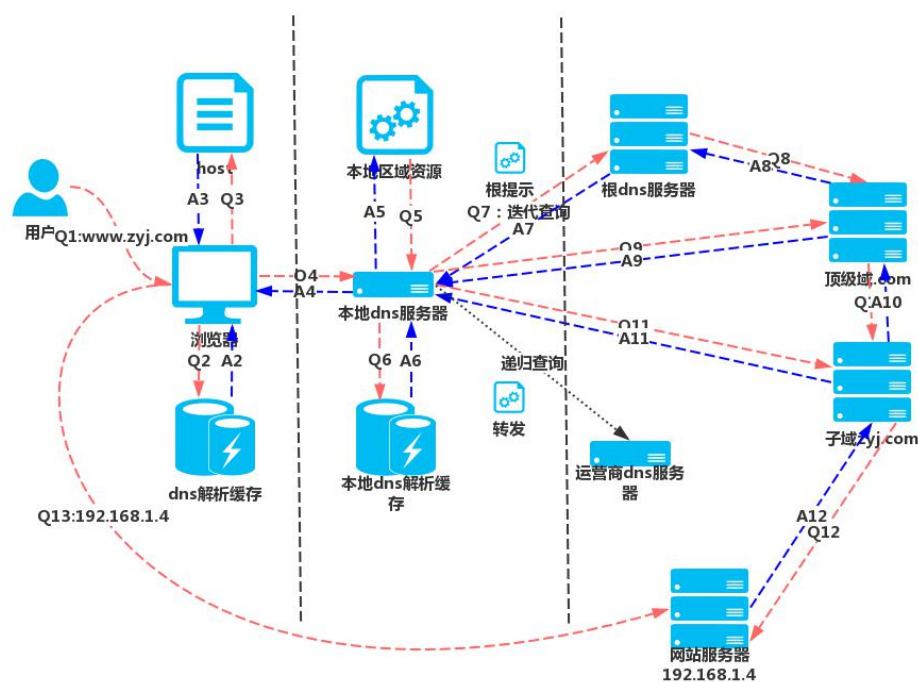
<http://canvas.tongji.edu.cn/>

1.4.2 网络拓扑与数据流向

参考网站: https://blog.csdn.net/qq_22211217/article/details/102175370

网络拓扑:

DNS 网络拓扑图:



https://blog.csdn.net/qq_22211217

数据流向：

Q1: 用户输入网址

Q2: 首先，浏览器查询 dns（Domain Name System，域名系统服务协议）解析缓存，浏览器设置，有时长和个数限制

Q3: Q2 未命中，查询本机 host 文件配置

Q4: Q3 未命中，向本地 dns 服务器查询，获得首选 dns（一般是运营商提供）

Q5: 查询本地区域资源配置，确保权威性

Q6: 查询本地 dns 解析缓存，会定时清理

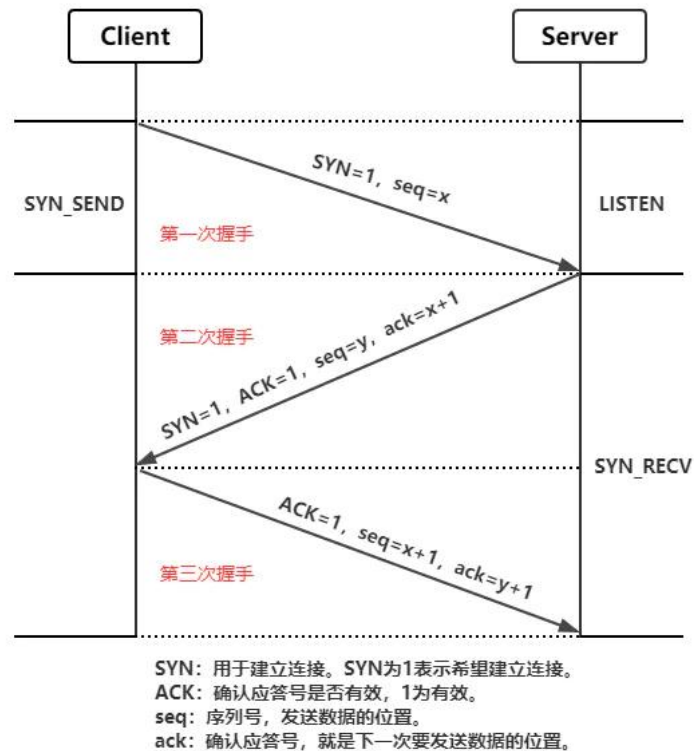
若未命中且本地配置转发 dns，可直接向上一级转发，递归查询

Q7-Q12: 若 Q6 未命中但未配置转发 dns，通过根提示迭代查询

A11, A4: 返回 ip，本地 dns 解析缓存与浏览器 dns 解析缓存保存 ip

Q13: 浏览器与网站服务器连接，上网

TCP 三次握手



这张图是上述 DNS 网络拓扑图中浏览器与网站服务器之间的双向箭头 Q13 的详细情况。

三次握手：

客户端发送连接请求给服务器（ $SYN=1$ ），发送自己的初始序列号（ $seq=x$ ）。客户端变为 SYN_SEND 态。

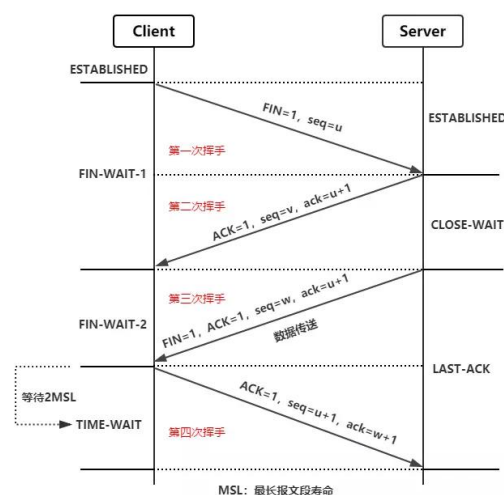
服务器收到报文，同意连接，发回 $SYN=1$ ， $ACK=1$ 表示同意连接，发送自己的初始序列号（ $seq=y$ ），发送希望收到的下一个客户端的序列号 $ack=x+1$ 。服务器变为 SYN_RECV 态。

客户端收到报文，发回一个确认报文段，表示收到服务器同意连接的报文（ $ACK=1$ ），发送自己的下一个序列号（ $seq=x+1$ ），发送希望收到的下一个服务器的序列号 $ack=y+1$ 。

至此，三次握手完毕，双方通信正常，进入 ESTABLISHED 态。

二者互传 HTTP 协议报文。

四次挥手



二者互传 HTTP 协议报文结束，TCP 连接断开，四次挥手。

客户端向服务器发送断开连接请求。参数为：FIN=1，seq=u。

服务器收到请求，通知高层进程连接释放，发回应答。参数为：ACK=1，ack=u+1，seq=v。
自身进入 CLOSE_WAIT 态。

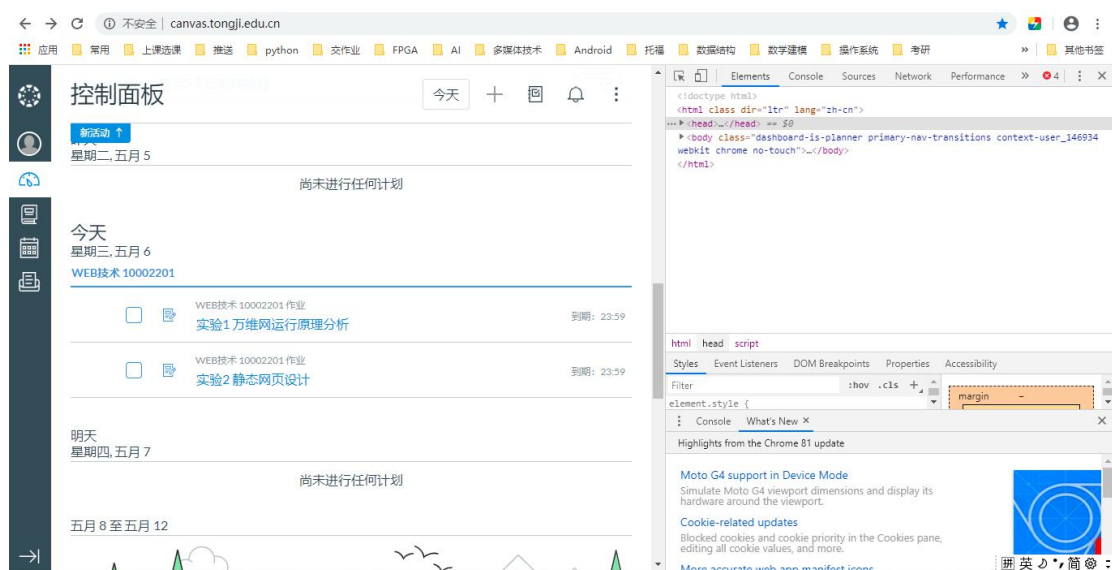
客户端收到应答进入 FIN_WAIT_2 态，等待服务器数据传输结束发送报文。

服务器数据传输结束，发送连接释放报文。参数：FIN=1，ack=u+1（客户端未再发来数据），seq=v+n（服务器可能传输了部分数据），进入 LAST_ACK 态。

客户端收到请求，发回报文。参数：ACK=1，ack=v+n+1，seq=u+1。进入 TIME_WAIT 态，等待 2MSL（最长报文段寿命）时间，如服务器无数据传输，进入 CLOSED 态。服务器收到报文直接进入 CLOSED 态。

1.4.3 分析网页组成

打开 Chrome 的开发者工具。



在 Element 中可以查看网站源代码。

<head></head>标签

其中主要描述了一些外部样式<link>。

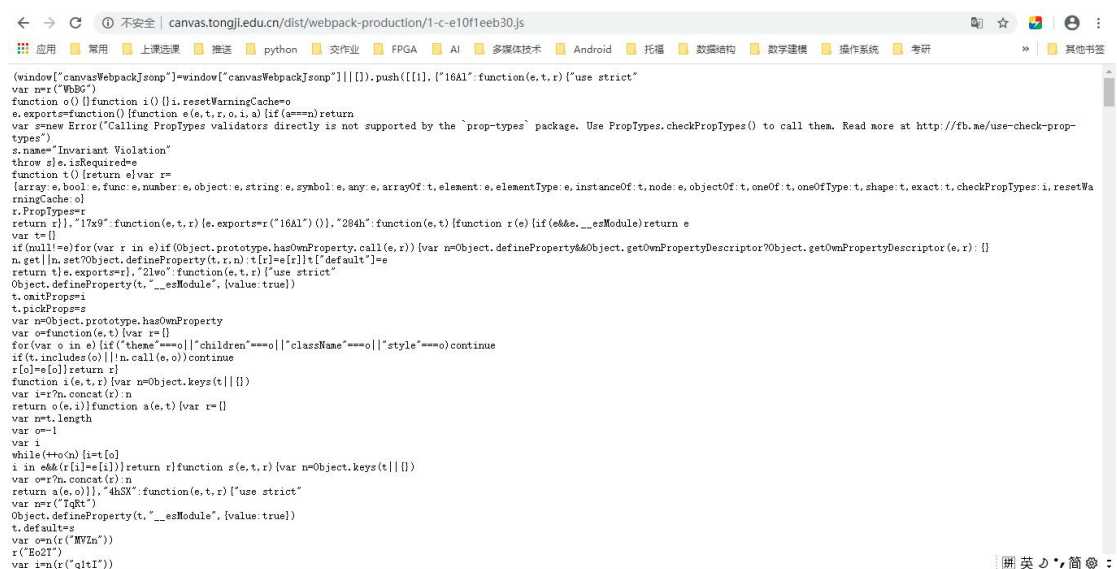
```
*** <link rel="stylesheet" media="all" href="/dist/brandable_css/default/variables-8f05bb4...css"> == $0
```

我们可以通过右键链接，Copy Link Address，进入这个网站查看具体的布局信息。

```
:root {
  --ic-brand-primary-darkened-5: #0087D7;
  --ic-brand-primary-darkened-10: #0080CC;
  --ic-brand-primary-darkened-15: #0079C1;
  --ic-brand-primary-lightened-5: #0C93E3;
  --ic-brand-primary-lightened-10: #1999E4;
  --ic-brand-primary-lightened-15: #269EE6;
  --ic-brand-button--primary-bgd-darkened-5: #0087D7;
  --ic-brand-button--primary-bgd-darkened-15: #0079C1;
  --ic-brand-button--secondary-bgd-darkened-5: #2B3942;
  --ic-brand-button--secondary-bgd-darkened-15: #27333B;
  --ic-brand-font-color-dark-lightened-15: #4C5860;
  --ic-brand-font-color-dark-lightened-30: #6C757C;
  --ic-link-color-darkened-10: #0080CC;
  --ic-link-color-lightened-10: #1999E4;
  --ic-brand-primary: #008EE2;
  --ic-brand-font-color-dark: #2D3B45;
  --ic-link-color: #008EE2;
  --ic-brand-button--primary-bgd: var(--ic-brand-primary);
  --ic-brand-button--primary-text: #ffffff;
  --ic-brand-button--secondary-bgd: #2D3B45;
  --ic-brand-button--secondary-text: #ffffff;
  --ic-brand-global-nav-bgd: #394B58;
  --ic-brand-global-nav-icon-svg-fill: #ffffff;
  --ic-brand-global-nav-icon-svg-fill--active: var(--ic-brand-primary);
  --ic-brand-global-nav-menu-item__text-color: #ffffff;
  --ic-brand-global-nav-menu-item__text-color--active: var(--ic-link-color);
  --ic-brand-global-nav-avatar-border: #ffffff;
  --ic-brand-global-nav-menu-item__badge-bgd: var(--ic-brand-primary);
  --ic-brand-global-nav-menu-item__badge-text: #ffffff;
  --ic-brand-global-nav-logo-bgd: #394B58;
  --ic-brand-header-image: url('/dist/images/canvas_logomark_only@2x-e197434829.png');
  --ic-brand-watermark: ;
  --ic-brand-watermark-opacity: 1.0;
  --ic-brand-favicon: url('/dist/images/favicon-e10d657a73.ico');
  --ic-brand-apple-touch-icon: url('/dist/images/apple-touch-icon-585e5d997d.png');
  --ic-brand-msapplication-tile-color: var(--ic-brand-primary);
  --ic-brand-msapplication-tile-square: url('/dist/images/windows-tile-eda889e7b.png');
  --ic-brand-msapplication-tile-wide: url('/dist/images/windows-tile-wide-44d3cc1060.png');
  --ic-brand-right-sidebar-logo: ;
  --ic-brand-Login-body-bgd-color: #394B58;
}
```

还有一些<script></script>标签，插入了 JavaScript，我们同样可以复制链接查看相应的 js 代码。

```
<script charset="utf-8" src="/dist/webpack-production/1-c-e10f1eeb30.js">
</script>
```



也有<style></style>标签定义样式。

```
<style type="text/css" data-glamor></style>
<style type="text/css">...</style>
<style type="text/css">...</style>
```


在<body></body>中，鼠标移动到对应代码部分网页相应区域会被标记。

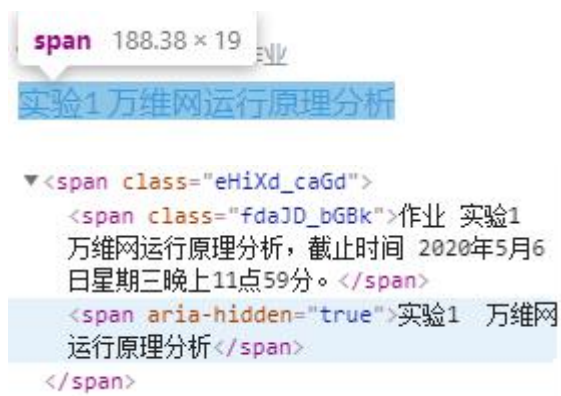


(全局导航)

进一步可以选定更细节的部分。



其他展示。



1.4.4 分析 HTTP 协议

在 NetWork 中可以查看 HTTP 相关协议。

首先进入的是统一身份认证界面。选中该界面。





















在 Headers 中查看 HTTP 报文。(view source)

需要使用对比缓存来验证缓存数据。

Set-Cookie: JSESSIONID=86E82C5A83FFF06674B9FAC7414A1268; Path=/nidp/; Secure; HttpOnly
设置了客户端的 cookie，对比发现和刚才 HTTP 请求报文中的第一条 JSESSIONID 是相同的，
为 86E82C5A83FFF06674B9FAC7414A1268。

输入学号密码验证码进入 canvas。选择主网页。其他网页都是之前 1.4.3 分析网页组成中调用的网页。

	canvas.tongji.edu.cn
	Shanghai-f1a172c811.js
	zh_CN-a04f4e8062.js
	zh-cn-c-55c8b84795.js
	main-e-04ea0cb52f.js
	1-c-e10f1eeb30.js
	2-c-06a5d09f20.js
	3-c-9e59f9e1e8.js
	4-c-0de4e47220.js
	5-c-124499549e.js
	7-c-d5f6ceef46.js
	8-c-672f0e5e1e.js
	9-c-52f7674a96.js
	10-c-d0f420ee62.js
	11-c-711d0da023.js
	12-c-9bf3db9897.js
	13-c-b761dfd4ea.js
	14-c-a52229b5b8.js

查看 HTTP 请求。

```
GET / HTTP/1.1
Host: canvas.tongji.edu.cn
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/81.0.4044.129 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: SRV=760ab19e-b8db-4cd9-987f-d98f2370e11c; log_session_id=fa3c40782debeeff5e404e52ba5051f; _normandy_session=P8IGd0_XBTNswSz7PpzVQ+4dIvUV-eabwR1J-1AG662rU7LYZ4oCc-MEaMd1RwLAWzCM2ugFJ5SkMtDJR1Ky26RJKJkNvHvO1Ax6HNWJ0bGbDBzWda7107_AUvVrFpVM-TS7uDtN1jV4FudEivDuY7AUwQuTCzrjpkaSKv46ZsYfI68scgHAaU1dcR5TehGGxK_gynMmZrNE6GZrYevv5DiyHqooYZ99AQYmPYoqNhjfaGYAYNIIt-Yezj4iBnQw4CjeYaySL-8LkxGCxXaxyhS7dN6iNzFBrNS1AQjz_z9UFW1GMM_i0kj8wuTIosda9iFyD51Przsh3DDWw0uTZG66m1QhoswZDpVw8VpgK7bD8NAOp7djQkUU6EPiZLOcVoY1Efoc69b81Y6Cyx7JFjpV.Gd6WdIp1NbuGqZUnP0ZEckFnz48.XrLDig; _csrf_token=gDh8Glm%2BCTbHcSsZWqg8mbPeQa1LbXre9F5Z0I338e%2FvcXh7Ks06YLIoWiAb%2B0zdxesJyBkYCrKhPG6dpoZy1g%3D%3D
```

对报文进行分析，基本同登陆界面，唯一的区别在于在 Cookie 中查找不到 JSSESSIONID。

```
Cookie: SRV=760ab19e-b8db-4cd9-987f-d98f2370e11c; log_session_id=fa3c40782debeeff5e404e52ba5051f; _normandy_session=P8IGdO_XBTNswSzx7PpzVQ+4dIvUV-eabwR1J-1AG662rU7LYZ4oCc-MEaMd1RwLAWzCM2ugFJ5kMtDJR1Ky26RJJKkNvHvO1Ax6HNWJ0bGbDBzWda7107_AUvVrfpVM-TS7uDtN1jV4FudEivDuY7AUwQuTCzrjpkasKv46ZsYfI68scgHaaU1dcR5TehGGxK_gynMmZrNE6GZrYevv5DiyHqooYZ99AQYmPYoqNhjfaGYAYNIIt-Yezj4iBnQw4CjeYaySL-8LkxGCxXaxyhS7dN6iNzFBrNS1AQjz_z9UFW1GMm_i0kj8wuTIosda9iFyD51Przsh3DDWw0uTZG66m1QhoswZDpvw8VpgK7bD8NAOp7djQkUU6EPiZLOcVoY1Efoc69b81Y6Cyx7JfjPv.Gd6WdIp1NbuGqZUnP0ZECKFnz48.XrLDig; _csrf_token=gDhBGWm%2BCTbHcSsZWqg8mbPeQa1LbXre9F5Z0I33Be%2FvcXh7Ks06YLIoWiAb%2B0zdxesJyBkYCrKhPG6dpoZy1g%3D%3D
```

查看 HTTP 响应。

▼ Response Headers	view parsed
HTTP/1.1 200 OK	
Server: nginx/1.17.6	
Date: Wed, 06 May 2020 14:03:35 GMT	
Content-Type: text/html; charset=utf-8	
Transfer-Encoding: chunked	
X-XSS-Protection: 1; mode=block	
X-Content-Type-Options: nosniff	
X-UA-Compatible: IE=Edge,chrome=1	
X-Canvas-User-Id: 10000000146934	
X-Frame-Options: SAMEORIGIN	
Pragma: no-cache	
Cache-Control: no-cache, no-store	
X-Request-Cost: 0.2586586639979897	
X-Rate-Limit-Remaining: 600.0	
Set-Cookie: _csrf_token=Wg%2FGgxTenPQ4IAS8I5LJLeHjDKISBvk7a8Z1PF5Pgio1Rv%2FhV62vok15dYVivblp19ZEw0BziVc%2BpEJxdT71Ew%3D%3D; path=/ X-Session-Id: fa3c40782debeeff5e404e52ba5051f X-Request-Context-Id: a8aa2435-24c4-4a37-92f3-d77e1a999bf6	

分析也同之前，区别在于 Set-Cookie 中设置了 _csrf_token=Wg%2FGgxTenPQ4IAS8I5LJLeHjDKISBvk7a8Z1PF5Pgio1Rv%2FhV62vok15dYVivblp19ZEw0BziVc%2BpEJxdT71Ew%3D%3D。

匹配刚才查看的 HTTP 请求中的 Cookie 属性，发现确实存在 _csrf_token=gDhBGWm%2BCTbHcSsZWqg8mbPeQa1LbXre9F5Z0I33Be%2FvcXh7Ks06YLIoWiAb%2B0zdxesJyBkYCrKhPG6dpoZy1g%3D%3D。

但二者并不相同。并不清楚这个属性同 JSSESSIONID 存在什么关系。

1.5 心得体会

通过这次的实验，我对客户端浏览器与服务器之间的建立联系的过程，以及其中的数据交互有了更加具体的认识。这个数据交互的系统非常严谨，传输数据可能非常简单，但是通过巧妙地设计缓存系统，互相确认的流程，公认的协议，保证了数据传输的效率与准确性。特别分析了 HTTP 协议中的请求与响应后，更能体会到这一点。但因为自己这方面的知识还是有限，并不能讲整个报文彻底分析透彻，还是会有一些遗憾。