**HTTP权威指南**

connection:Keep-Alive首部 是个逐跳首部，只适用于单条传输链路 不应该沿着传输链路向下传输。盲中继（代理不认识connection:Keep-Alive）

TCP的连接是双向的。TCP连接的每一端都有一个输入队列和输出队列，用于数据的读或写

完全关闭：套接字调用close( )会将tcp连接的输入和输出信道都关闭

# **TCP的三次握手**

**1.请求新的TCP连接时，客户端要向服务端发送一个小的TCP分组。这个分组中设置了一个特殊的SYN标记，说明这是一个连接请求。**

**2.如果服务器接受了连接就会对一些连接参数进行计算，并向客户端回送一个TCP分组，这个分组中的SYN和ACK标记都被置位，说明连接请求已被接受**

**3.客户端向服务器端回送一条确认信息（确认分组（ACK分组）），通知它连接已成功建立**

一条tcp连接 由四部分组成

客户端的ip 客户端的端口 服务端的ip 服务端的端口

（若url没有端口号，就是默认使用的80端口啊）

（一般客户端的端口是随机的 服务端的端口是固定的那位）

http://www.baidu.com:80/index.html

ip地址没有使用数字形式 而是文本形式的域名 或者称为主机名(www.baidu.com)。主机名就是ip地址比较人性化的别称。可以通过一种称为域名服务(Domain Name Server,DNS)的机制方便的将主机名装换为IP地址。

**敲下url的那一刻：**

**1 浏览器从url中解析出服务器的主机名(www.baidu.com)**

**2 浏览器将服务器的主机名转换成服务的IP地址**

**3 浏览器将端口号（如果有的话）从url中解析出来**

**4 浏览器建立起一条与web服务器的TCP连接**

**5 浏览器向服务器发送一条http请求报文**

**6 服务器向浏览器回送一条http响应报文**

**7 关闭连接，浏览器显示文档**

**URL都有同样的命名方式：**

**协议(scheme)://服务器位置(host)/资源路径(path)**

**常见的协议及描述**

**http 默认端口80 没有用户名和密码**

**https 默认端口443 唯一区别http是它使用了网景的SSL（安全套接层）**

**提供了端到端的加密机制**

**ftp 文件传输协议**

**file 访问本地文件**

**HTTP报文是简单格式的数据包，每条报文都包含一条来自客户端的请求或者一条来自服务器的响应。报文由三部分组成：起始行，首部，主体（可选）**

**通用首部：客户端和服务器端都可以使用**

**Date：报文被创建的时间和日期**

**Connection：请求/响应连接的有关的选项**

**通用缓存首部：说明了如何或什么时候进行缓存**

**Cache-Control：过期时间，相对时间（优先级高）**

**Pragma：另一种指示，并不专用于缓存**

**请求首部：请求报文特有（告诉服务器一些额外的信息）**

**Accept：用来告知服务器，我客户端希望接收的数据类型**

**Host：接收请求的服务器的主机名和端口号**

**User-Agent：将发起请求的应用程序名称告知服务器（是电脑啊还是手机...）**

**Cookie：客户端向服务器传送的一个‘令牌’**

**Authorization：包含了客户端提供给服务器的自身认证的数据**

**响应首部：响应报文特有（为客户端提供信息）**

**Server：用来告知客户端正在和那个服务器进行交互**

**Age：响应所持续的时间**

**WWW-Authorization：来自服务器对客户端的质询列表**

**Set-Cookie：可以在客户端设置一个‘令牌’，以便服务器对客户端进行标识**

**实体首部：说明(实)主体的信息**

**Content-Type：说明实体主体部分的数据类型**

**Content-Location：资源实际所在的位置**

**Content-Encoding：对主体执行的编码方式**

**实体缓存首部：验证已缓存的资源副本是否依旧有效或者估计缓存何时失效**

**Etag：与此实体相关的标记**

**Expires：过期时间，绝对时间**

**Last-Modified：这个实体最后一次被修改的日期和时间**

#### 解决http无状态的问题

**用户认证：Authorization 缺点：每个站点都要输入密码，变更密码时不好**

**胖URL：url一共享就信息泄露，重写html页面使url变胖，url地址非常的丑陋**

**cookie：实现持久会话的最好方式**

**cookie的介绍**

**分为两类：会话cookie和持久cookie 它们之间唯一的区别就是过期时间**

**如果设置了Discard参数（:丢弃的意思）**

**或者说没有设置Expires**

**或者没有设置Max-Age参数来拓展过期时间的话 这个就是会话cookie**

**cookie是如何工作的**

**服务器：服务器 给 用户贴一张纸条**

**Set-cookie:id=‘15156’;domain=’baidu.com’...**

**(domain是控制条件 告诉那些站点可以看到那个cookie)**

**浏览器：会记住从服务器返回的Set-cookie首部中的cookie的内容，并将cookie集存储在浏览器的cookie数据库中**

**其基本思想就是让浏览器积累一组服务器特有的信息，浏览器每次访问服务器时，都会将这些信息(部分)提供给服务器**

**Cookie：session-id=002-1145265-4544 （客户端发送请求时）**

## HTTP的认证机制（认证协议）： 基本认证 、 摘要认证

**基本认证协议（便捷灵活但是不安全，用户名和密码还是以Base6明文传送的）：**

**请求**

**第一条请求没有认证信息**

**质询**

**(WWW-Authenticate)服务器用401状态拒绝了请求 附带WWW-Authenticate说明如何以及在哪里进行认证**

**认证**

**(Authorization) 客户端重新发请求，并且附带一个Authorization首部，用来说明认证的算法、用户名和密码**

**成功**

**返回200ok**

**Base-64为用户名/密码进行编码变成一串乱码(在别人眼里其实就是明文)形如：**

**客户端发送认证消息：Authorization：Basic AyasHsa2J2jJK3jSoA**

**摘要认证协议**

**（只是在基本认证上改进不会用明文方式发送密码，四个过程基本一样）**

**摘要认证的名言就是‘绝对不通过网络发送密码’。客户端不会发送一个密码出去，而是取而代之发送一个‘指纹’||‘摘要’，这是不可逆的乱码。摘要认证的缺点很明显，它只保护密码，并没有保护其他的内容啊。**

**摘要的概念**

**摘要是‘对信息主体的浓缩’。摘要是一种单向函数，也叫加密的校验和或者单向散列表函数或者是指纹函数**

**此函数的作用就是将无限的输入值转换为有限的输出值。常见的摘要函数MD5（会将任意长度的字节序列转换为一个128位的摘要，即32个16进制的字符哦）。摘要认证要求使用随机数来防止重放攻击**

**传输层安全协议(TLS)和安全HTTP(即HTTPS)协议的组合协议最安全**

**TLS+SSL+HTTP协议无敌哈哈下节将会介绍！**

# **数字加密**

**密钥：改变密码行为的数字化参数**

**对称密钥加密系统：解/编码使用相同密钥算法（共享密钥）**

**非对称密钥加密系统：解/编码使用不同密钥的算法（公开密钥）**

**数字签名（通常是用非对称公开密钥技术产生的）**

**1.签名可以证明是作者编写了这条报文。**

**2.签名可以防止报文被篡改，一旦有人篡改，校验和就不在匹配了**

# SSL握手

**在发送已加密的HTPP报文之前，客户端和服务端要进行一次SSL握手**

1. **客户端发送可供选择的密码和请求证书**
2. **服务器发送选中的密码和证书**
3. **客户端发送保密信息；客户端和服务器生成密钥**
4. **客户端和服务端互相告知，开始加密过程**

**条件请求与验证码**

**请求类型 验证码**

**If-Modified-Since Last-Modified**

**If-None-Match Etag**

**If-Modified-Since:如果是从2018,9,4号之后修改过的，请发给我**

**XSS攻击**

**XSS攻击全称[跨站脚本攻击](https://baike.baidu.com/item/%E8%B7%A8%E7%AB%99%E8%84%9A%E6%9C%AC%E6%94%BB%E5%87%BB/8186208" \t "https://baike.baidu.com/item/XSS%E6%94%BB%E5%87%BB/_blank)，是为不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆，故将跨站脚本攻击缩写为XSS，XSS是一种在web应用中的计算机安全漏洞，它允许恶意web用户将代码植入到提供给其它用户使用的页面中。**

**XSS原理**

**攻击者对含有漏洞的服务器发起XSS攻击（注入JS代码）。**

**诱使受害者打开受到攻击的服务器URL。**

**受害者在Web浏览器中打开URL，恶意脚本执行。**

**XSS攻击方式**

****反射型：**也叫非持久型XSS，发出请求时，XSS代码出现在URL中，作为输入提交到服务器端，服务器端解析后响应，XSS随响应内容一起返回给浏览器，最后浏览器解析执行XSS代码，这个过程就像一次发射，所以叫反射型XSS。**

****存储型:**存储型XSS和反射型的XSS差别就在于，存储型的XSS提交的代码会存储在服务器端（数据库，内存，文件系统等），下次请求目标页面时不用再提交XSS代码。**

**解决方法如下**

**对cookie的保护:**

**对重要的cookie设置httpOnly, 防止客户端通过document.cookie读取cookie。服务端可以设置此字段。**

****对用户输入数据的处理:****

**编码：不能对用户输入的内容都保持原样，对用户输入的数据进行字符实体编码。对于字符实体的概念可以参考文章底部给出的参考链接。**

**解码：原样显示内容的时候必须解码，不然显示不到内容了。**

**过滤：把输入的一些不合法的东西都过滤掉，从而保证安全性。如移除用户上传的DOM属性，如onerror，移除用户上传的Style节点，iframe, script节点等。**

****CSRF攻击****

**CSRF可以叫做(跨站请求伪造)，咱们可以这样子理解CSRF，攻击者可以利用你的身份你的名义去发送(请求)一段恶意的请求,从而导致可以利用你的账号(名义)去--购买商品、发邮件,恶意的去消耗账户资源,导致的一些列恶意行为.CSRF可以简单分为Get型和Post型两种。**

****CSRF的原理****

**发现漏洞可利用处->构造(搭建)搭建代码->发送给用户(管理员)->触发代码(发送请求).........**

**从这个利用的一个流程中，我们可以发现,攻击者仅仅只是做了两处工作.第一处是:发现漏洞利用处，，第二处就是构造利用代码以及发送至用户(管理员)。至于利用，你会发现CSRF与XSS不同，XSS是攻击者自己提交，等待结果，而CSRF呢，是由用户(管理员)自身提交。甚至可以说攻击者只做了构造代码的工作。**

****在开发中如何简单防御CSRF(PHP)****

**其实防御CSRF有很多种 如:验证码、验证Refer、以及验证token，对特殊参数进行加密。**

**但是如果使用验证码去避免CSRF的话，那么这样会验证的影响用户的体验，因为用户不会每个操作都去输入验证码(会很烦)。**

**Refer的话在特殊情况下也是不靠谱的(服务器端出的问题)。**

**那么目前只有token是被大多网站去使用的。因为可以避免用户体验的问题发生。同样服务器边问题也发生也不会很多。**

**那么接下来就开始介绍在PHP开发中如何去简单的生成token，避免CSRF。我们可以通过PHP中函数(rand生成随机数+uniqid生成一个唯一id+time时间戳)最后在讲这几个生成的值用md5加密。**