http和https协议

https://zhuanlan.zhihu.com/p/72616216

2020年2月12日 星期三　下午7:21

http简介

HTTP协议是超文本传输协议的缩写，英文是Hyper Text Transfer Protocol。它是从WEB服务器传输超文本标记语言(HTML)到本地浏览器的传送协议。

设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。

HTTP（超文本传输协议）是应用层上的一种客户端/服务端模型的通信协议,它由请求和响应构成，且是无状态的。（暂不介绍HTTP2）

URL构成

http协议构成

请求

- 样例：

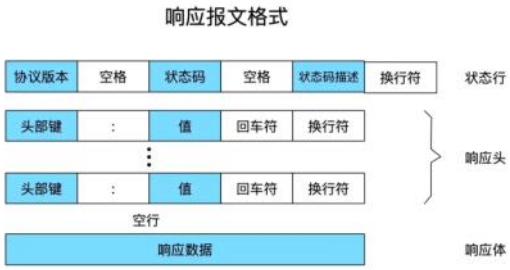
```
GET /department/87423/users HTTP/1.1
host: www.xxx.com
accept: application/json
accept-encoding: gzip, deflate, br
accept-language: zh-CN,zh;q=0.9
user-agent: AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109

name=flyhero
```

码上实战

知乎 @flyhero

响应



- 样例：

```
HTTP/1.1 200 OK
Date: Tue, 19 Mar 2019 02:33:04 GMT
Content-Length: 4393
Content-Type: application/json; charset=utf-8
Expires: Tue, 19 Mar 2019 01:35:29 GMT
Pragma: no-cache

{
  "Name":"flyhero",
  "Id":"push-code"
```

更多关注

码上实战

知乎 @flyhero

HTTP和HTTPS有什么区别

1. 端口不同：HTTP使用的是80端口，HTTPS使用443端口；
2. HTTP（超文本传输协议）信息是明文传输，HTTPS运行在SSL(Secure Socket Layer)之上，添加了加密和认证机制，更加安全；
3. HTTPS由于加密解密会带来更大的CPU和内存开销；
4. HTTPS通信需要证书，一般需要向证书颁发机构（CA）购买

HTTPs连接过程

1. 客户端向服务器发送请求，同时发送客户端支持的一套**加密规则**（包括对称加密、非对称加密、摘要算法）；
2. 服务器从中选出一组加密算法与HASH算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含了网站地址，加密公钥（用于非对称加密），以及证书的颁发机构等信息（证书中的私钥只能用于服务器端进行解密）；
3. 客户端验证服务器的合法性，包括：证书是否过期，CA 是否可靠，发行者证书的公钥能否正确解开服务器证书的“发行者的数字签名”，服务器证书上的域名是否和服务器的实际域名相匹配；
4. 如果证书受信任，或者用户接收了不受信任的证书，浏览器会生成一个随机密钥（用于对称算法），并用服务器提供的公钥加密（采用非对称算法对密钥加密）；使用Hash算法对握手消息进行摘要计算，并对摘要使用之前产生的密钥加密（对称算法）；将加密后的随机密钥和摘要一起发送给服务器；
5. 服务器使用自己的私钥解密，得到对称加密的密钥，用这个密钥解密出Hash摘要值，并验证握手消息是否一致；如果一致，服务器使用对称加密的密钥加密握手消息发给浏览器；
6. 浏览器解密并验证摘要，若一致，则握手结束。之后的数据传送都使用对称加密的密钥进行加密

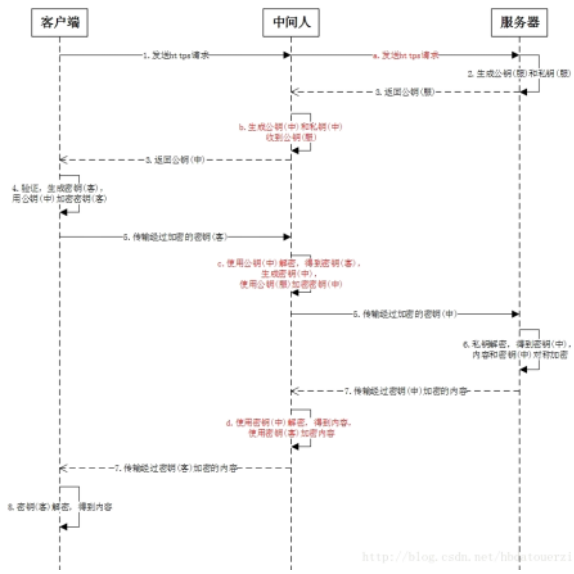
输入网址为什么会自动跳转到https

其实大家平时在使用浏览器的时候，一般是直接在地址栏里面输入域名，然后就访问了。但大家应该知道，大多数浏览器在默认情况下会先用 HTTP 发起请求的。也就是说即便你的站点已经支持了 HTTPS，但如果不做任何处理的话，用户还是很难触及到

而为什么我们平时访问很多网站的时候自动就跳转到 HTTPS 站点了呢，也是因为这些站点对这一点做了处理。 最原始的方法就是 302 跳转，服务端把所有的 HTTP 流量跳转到 HTTPS 上。 但这样做有一个明显的安全漏洞， 就是第一次访问站点的时候如果是 HTTP 就有可能被中间人劫持，很可能都没到 302 跳转的时候就被劫持了。

这也就是为什么要引入 HSTS 机制的原因了。 用户的浏览器一旦得到了 HSTS 的信息，下次再访问站点的时候客户端浏览器就会强制使用 HTTPS。 无论你在地址栏里输入什么，都会以 HTTPS 访问。 这样就避免了每次服务端跳转可能导致的潜在安全问题。

中间人劫持攻击



中间人截取客户端发送给服务器的请求，然后伪装成客户端与服务器进行通信；将服务器返回给客户端的内容发送给客户端，伪装成服务器与客户端进行通信。
通过这样的手段，便可以获取客户端和服务器之间通信的所有内容。
使用中间人攻击手段，必须要让客户端信任中间人的证书，如果客户端不信任，则这种攻击手段也无法发挥作用。

```
{
    "Name":"flyhero",
    "Id":"push-code"
}
```


状态码

HTTP状态码由三个十进制数字组成，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

- 200 OK - 客户端请求成功
- 301 - 资源（网页等）被永久转移到其它URL
- 302 - 请求的资源现在临时从不同的 URI 响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求
- 400 Bad Request - 客户端请求有语法错误，不能被服务器所理解。或者请求参数有错误
- 401 Unauthorized - 当前请求需要用户验证。该响应必须包含一个适用于被请求资源的 WWW-Authenticate 信息头用以询问用户信息。客户端可以重复提交一个包含恰当的 Authorization 头信息的请求。如果当前请求已经包含了 Authorization 证书，那么401响应代表着服务器验证已经拒绝了那些证书。
- 403 - Forbidden 服务器已经理解请求，但是拒绝执行它
- 404 - 请求资源不存在，可能是输入了错误的URL
- 500 - 服务器内部发生了不可预期的错误
- 501 - 服务不可用
- 503 Server Unavailable - 服务器没有准备好处理请求。 常见原因是服务器因维护或重载而停机。 请注意，与此响应一起，应发送解释问题的用户友好页面。

请求方法

- GET:请求指定的页面信息，并返回实体主体。

- POST:向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。

- HEAD:类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
- PUT:从客户端向服务器传送的数据取代指定的文档的内容。
- DELETE:请求服务器删除指定的页面。

POST和GET的区别

- 都包含请求头请求行，post多了请求body。
- get多用来查询，请求参数放在url中，不会对服务器上的内容产生作用。post用来提交，如把账号密码放入body中。
- GET是直接添加到URL后面的，直接就可以在URL中看到内容，而POST是放在报文内部的，用户无法直接看到。
- GET提交的数据长度是有限制的，因为URL长度有限制，具体的长度限制视浏览器而定。而POST没有。

请求和响应常见通用头

也就是请求和响应一般都有的

名称	作用
Content-Type	请求体/响应体的类型，如：text/plain、application/json
Accept	说明接收的类型，可以多个值，用 , (半角逗号)分开
Content-Length	请求体/响应体的长度，单位字节
Content-Encoding	请求体/响应体的编码格式，如gzip,deflate
Accept-Encoding	告知对方我方接受的Content-Encoding
ETag	给当前资源的标识，和 Last-Modified 、 If-None-Match 、 If-Modified-Since 配合，用于缓存控制
Cache-Control	取值为一般为 no-cache 或 max-age=XX，XX为个整数，表示该资源缓存有效时间

Content-Type，内容类型，一般是指网页中存在的Content-Type，用于定义网络文件的类型和网页的编码，决定浏览器将以什么形式、什么编码读取这个文件

Content-Type(Mime-Type)	描述
text/html	HTML格式
text/plain	纯文本格式
text/xml	XML格式

通过这样的手段，便可以获取客户端和服务器之间通信的所有内容。

使用中间人攻击手段，必须要让客户端信任中间人的证书，如果客户端不信任，则这种攻击手段也无法发挥作用。

预防中间人劫持方法

针对安全性要求比较高的 app，可采取客户端预埋证书的方式锁死证书，只有当客户端证书和服务端的证书完全一致的情况下才允许通信，如一些银行类的app，但这种方式面临一个问题，证书过期的问题，因证书有一定的有效期，当预埋证书过期了，只有通过强制更新或者要求用户下载证书来解决

什么是对称加密、非对称加密？区别是什么？

- 对称加密：加密和解密采用相同的密钥。如：DES、RC2、RC4
- 非对称加密：需要两个密钥：公钥和私钥。如果用公钥加密，需要用私钥才能解密。如：RSA
- 区别：对称加密速度更快，通常用于大量数据的加密；非对称加密安全性更高（不需要传送私钥）

数字签名、报文摘要的原理

- 发送者A用私钥进行签名，接收者B用公钥验证签名。因为除A外没有人有私钥，所以B相信签名是来自A。A不可抵赖，B也不能伪造报文。
  主要使用公开密钥加密系统实现，如RSA；发送者通过私钥加密，接收方通过公秘解密。
- 摘要算法:MD5、SHA
  报文摘要：用于对发送的报文生成一个非常小的摘要信息。这个摘要信息保证原报文的完整性，即原报文只要有一位被改变，则摘要信息就会不匹配。

GET与POST的区别

1. GET是幂等的，即读取同一个资源，总是得到相同的数据，POST不是幂等的；
2. GET一般用于从服务器获取资源，而POST有可能改变服务器上的资源；
3. 请求形式上：GET请求的数据附在URL之后，在HTTP请求头中；POST请求的数据在请求体中；
4. 安全性：GET请求可被缓存、收藏、保留到历史记录，且其请求数据明文出现在URL中。POST的参数不会被保存，安全性相对较高；
5. GET只允许ASCII字符，POST对数据类型没有要求，也允许二进制数据；
6. GET的长度有限制（操作系统或者浏览器），而POST数据大小无限制

Session和Cookie的区别

Session是服务器端保持状态的方案，Cookie是客户端保持状态的方案

Cookie保存在客户端本地，客户端请求服务器时会将Cookie一起提交；Session保存在服务端，通过检索Sessionid查看状态。保存Sessionid的方式可以采用Cookie，如果禁用了Cookie，可以使用URL重写机制（把会话ID保存在URL中）。

从输入网址到获得页面的过程 (越详细越好)

1. 浏览器查询 DNS，获取域名对应的IP地址:具体过程包括浏览器搜索自身的DNS缓存、搜索操作系统的DNS缓存、读取本地的Host文件和向本地DNS服务器进行查询等。对于向本地DNS服务器进行查询，如果要查询的域名包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析(此解析具有权威性)；如果要查询的域名不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析（此解析不具有权威性）。如果本地域名服务器并未缓存该网址映射关系，那么将根据其设置发起递归查询或者迭代查询；
2. 浏览器获得域名对应的IP地址以后，浏览器向服务器请求建立链接，发起三次握手；
3. TCP/IP链接建立起来后，浏览器向服务器发送HTTP请求；
4. 服务器接收到这个请求，并根据路径参数映射到特定的请求处理器进行处理，并将处理结果及相应的视图返回给浏览器；
5. 浏览器解析并渲染视图，若遇到对js文件、css文件及图片等静态资源的引用，则重复上述步骤并向服务器请求这些资源；
6. 浏览器根据其请求到的资源、数据渲染页面，最终向用户呈现一个完整的页面。

从用户在浏览器输入域名,到浏览器显示出页面,这中间发生了什么

1.在浏览器中输入地址,如:www.baidu.com

2.向DNS服务器查询网站IP地址

3.DNS服务器返回网站IP地址(如:119.75.217.56)

Content-Type(Mime-Type)	描述
text/html	HTML格式
text/plain	纯文本格式
text/xml	XML格式
image/gif	gif图片格式
image/jpeg	jpg图片格式
image/png	png图片格式

Content-Type(Mime-Type)	描述
application/xml	XML数据格式
application/json	JSON数据格式
application/pdf	pdf格式
application/msword	Word文档格式
application/octet-stream	二进制流数据（如常见的文件下载）
application/x-www-form-urlencoded	form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据的格式）
multipart/form-data	需要在表单中进行文件上传时，就需要使用该格式

常见请求头

请求一般会包含的头部

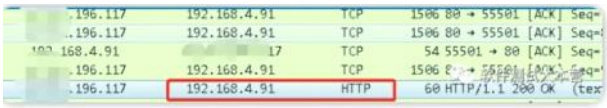
名称	作用
Authorization	用于设置身份认证信息
User-Agent	用户标识，如：OS和浏览器的类型和版本
If-Modified-Since	值为上一次服务器返回的 `Last-Modified` 值，用于确认某个资源是否被更改过，没有更改过(304)就从缓存中读取
If-None-Match	值为上一次服务器返回的 ETag 值，一般会和 `If-Modified-Since` 一起出现
Cookie	已有的Cookie
Referer	表示请求引用自哪个地址，比如你从页面A跳转到页面B时，值为页面A的地址
Host	请求的主机和端口号

常见响应头

名称	作用
Date	服务器的日期
Last-Modified	该资源最后被修改时间
Transfer-Encoding	取值为一般为chunked，出现在Content-Length不能确定的情况下，表示服务器不知道响应版体的数据大小，一般同时还会出现 `Content-Encoding` 响应头
Set-Cookie	设置Cookie
Location	重定向到另一个URL，如输入浏览器就输入baidu.com回车，会自动跳到 https://www.baidu.com ，就是通过这个响应头控制的
Server	后台服务器

HTTPs协议

实际使用中，绝大说的网站现在都采用的是https协议，这也是未来互联网发展的趋势。下面是通过wireshark抓取的一个博客网站的登录请求过程



可以看到访问的账号密码都是明文传输， 这样客户端发出的请求很容易被不法分子截取利用，因此，HTTP协议不适合传输一些敏感信息，比如：各种账号、密码等信息，使用http协议传输隐私信息非常不安全。

一般http存在以下问题

- 请求信息明文传输，容易被窃听截取。
- 数据的完整性未校验，容易被篡改
- 没有验证对方身份，存在冒充危险

为了解决上述HTTP存在的问题，就用到了HTTPS。

HTTPS 协议（HyperText Transfer Protocol over Secure Socket Layer）：一般理解为HTTP+SSL/TLS，通过 SSL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进行加密。

SSL是什么

SSL（Secure Socket Layer，安全套接字层）：1994年为 Netscape 所研发，SSL 协议位于 TCP/IP 协议与各种应用层协议之间，为数据通讯提供安全支持。

浏览器在使用HTTPS传输数据的流程是：

1.在浏览器中输入地址,如:www.baidu.com

2.向DNS服务器查询网站IP地址

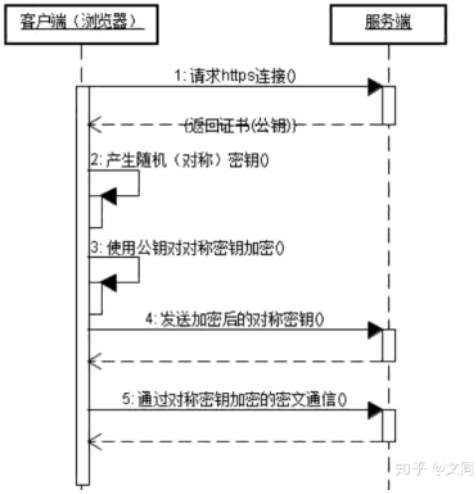
3.DNS服务器返回网站IP地址(如:119.75.217.56)

4.浏览器得到IP地址后.浏览器会把用户输入的域名转化为HTTP服务请求

5.服务器接收到请求后,返回网页信息

6.客户端浏览器将这些信息组织成用户可以查看的网页形式

注:由于数字式的IP地址难于记忆,所以就使用易于记忆的符号地址,这就需要一个数字地址和符号地址相互转换的机制,也就是DNS.而DNS服务器完成域名与IP地址转换的过程,就是**域名解析**.



1. 首先客户端通过URL访问服务器建立SSL连接。
2. 服务端收到客户端请求后，会将网站支持的证书信息（证书中包含公钥）传送一份给客户端。
3. 客户端的服务器开始协商SSL连接的安全等级，也就是信息加密的等级。
4. 客户端的浏览器根据双方同意的安全等级，建立会话密钥，然后利用网站的公钥将会话密钥加密，并传送给网站。
5. 服务器利用自己的私钥解密出会话密钥。
6. 服务器利用会话密钥加密与客户端之间的通信。
   对称密钥就是 客户端和服务端用同一个密钥进行加密解密，客户端先用RSA的方法将随机生成的对称密钥告诉服务端，服务端和客户端就用这个密钥进行交流。

HTTPS的缺点

- HTTPS协议多次握手，导致页面的加载时间延长近50%；
- HTTPS连接缓存不如HTTP高效，会增加数据开销和功耗；
- 申请SSL证书需要钱，功能越强大的证书费用越高。
- SSL涉及到的安全算法会消耗 CPU 资源，对服务器资源消耗较大。

HTTPS和HTTP的区别

- HTTPS是HTTP协议的安全版本，HTTP协议的数据传输是明文的，是不安全的，HTTPS使用了SSL/TLS协议进行了加密处理。
- http和https使用连接方式不同，默认端口也不一样，http是80，https是443

HTTP2.0和1.1区别

1. HTTP/2采用二进制格式而非文本格式
2. HTTP/2是完全多路复用的，而非有序并阻塞的——只需一个连接即可实现并行
3. 使用报头压缩，HTTP/2降低了开销
4. 服务器可以自动推送他认为客户端需要的数据，比如静态数据，避免等待客户端重复请求