http://www.webryan.net/2011/08/wiki-of-http-cookie/

[**全面解读HTTP Cookie**](http://www.webryan.net/2011/08/wiki-of-http-cookie/)

今天[webryan](http://www.webryan.net)给team做了一个关于HTTP cookie的分享，从各个方面给大家介绍一下大家耳熟能详的Cookie。主要是翻了维基百科的很多内容，因为维基百科的逻辑实在是很清晰：），ppt就不分享了，把原始的草稿贴出来给大家。欢迎批评指正。

HTTP Cookie:  
Cookie通常也叫做网站cookie,浏览器cookie或者http cookie,是保存在用户浏览器端的，并在发出http请求时会默认携带的一段文本片段。它可以用来做用户认证，服务器校验等通过文本数据可以处理的问题。  
Cookie不是软件，所以它不能被携带病毒，不能执行恶意脚本，不能在用户主机上安装恶意软件。但它们可以被间谍软件用来跟踪用户的浏览行为。所以近年来，已经有是欧洲和美国的一些律师以保护用户隐私之名对cookie的种植宣战了。更严重的是，黑客可以通过偷取Cookie获取受害者的帐号控制权。

1.Cookie的历史  
a.概念的产生：  
Cookie原名是”magic cookie”,是用来描述程序接受并原样发出的一组数据。（这里可以拿生活中的情况举例：比如说当我们在商场试穿衣服的时候，在试衣间门口店员会给你一个号码牌表示你试穿衣服的件数。当你出来的时候，店员会检查下拿着这个牌子和对应的衣服。）这个概念是一直就存在于计算机领域的.直到1994年，网景公司前雇员Lou Moutulli将magic cookie的概念引入到web,赋予了web记忆。

卖个关子。请大家设想一下，没有cookie的互联网将是一副什么的场景。  
1.每次访问都像第一次访问一样，无法判断用户是否访问过  
2.任何的购买等交互、验证行为都必须在一次访问中完成  
3.无任何记忆，均需要用户重新点击或填写

大家说的很好，在1994年6月的某天，24岁的，网景公司第九位工程师，moutulli，坐在键盘前，遇到和大家描述的一模一样的困难，他凭借着他高超的编程技能和思想设计出了一个五页纸的方案来解决这些棘手的问题，这个五页纸后来也就演变成了最初版本的Netscape Cookie规范。五页纸的核心观点就是在用户的电脑上存放一个小的文件来记录用户对网站的访问情况。moutulli将其简称为cookie. 同年10月，Netscape浏览器就率先支持了Cookies，并在netscape官网上做了检查统计用户是否访问过的功能。次年10月，微软的IE2也开始支持Cookie。当然，微软是要交“保护费”了，因为moutulli在1995年申请了专利，并在1998年获得专利授权。整体来说，Cookie的引入对于互联网来说，这是一个划时代的转折点，它将零散的访问，无状态的请求变得有序并富有记忆，给互联网增添了更有趣味性的玩法。

b.隐私风险  
1.1995年Q1，当时Cookie没有受到广泛的关注。但cookie不在用户知晓的情况下可以默认种植引来人们的担心。  
2.1996年2月，金融时报（ft中文网就是翻译此杂志）发布关于cookie的文章对公众进行了认知的普及，并引起了媒体的广泛关注和讨论，尤其是在潜在的隐私风险上。  
3.1996年-1997年，Cookie在美国联邦贸易委员会听证会持续讨论。

c.规范的发展：  
1.1994年，Montulli,联合John Giannandrea写了Netscape cookie规范。  
2.1995年4月，在www-talk邮件组第一次开始讨论正式统一的Cookie规范，并在IETF(Internet Engineering Task Force，互联网工程任务组，主要目标是协调制定互联网标准，几乎所有重要的网络底层协议都是有IETF制定,EG:TCP,IP,HTTP and so on,可以毫不夸张的说，没有IETF就没有今天的互联网，今年是IETF成立25周年，老外有很多文章专门回顾总结其光辉成就。IETF是由网民自发组织，自我管理的，任何人都和可以参加的，完全民主平等的，无投票机制的，充分体现了自由、开放、合作、共享的精神)里成立了特别工作小组。两种HTTP Cookie的方案被提议。  
3.1996年2月，IETF认定第三方Cookie存在重大隐私风险。  
4.1997年2月，IETF最终发布了Cookie规范,RFC 2109,其中指出不允许设置第三方cookie或者不能默认支持第三方cookie  
5.2000年10月，RFC 2965发布，主要是由于广告在RFC 2109发布时已经广泛使用第三方Cookie,所以关于第三方cookie部分是没有被Netscape和IE所跟随。  
6.2011年4月，RFC 6265发布。现实版使用的Cookie规范终于诞生了。

2.Cookie的类别  
a.Session Cookie  
这个类型的cookie只在会话期间内有效，即当关闭浏览器的时候，它会被浏览器删除。设置session cookie的办法是：在创建cookie不设置Expires即可。  
b.Persistent Cookie  
持久型cookie顾名思义就是会长期在用户会话中生效。当你设置cookie的属性Max-Age为1个月的话，那么在这个月里每个相关URL的http请求中都会带有这个cookie。所以它可以记录很多用户初始化或自定义化的信息，比如什么时候第一次登录及弱登录态等。  
c.Secure cookie  
安全cookie是在https访问下的cookie形态，以确保cookie在从客户端传递到Server的过程中始终加密的。这样做大大的降低的cookie内容直接暴露在黑客面前及被盗取的概率。  
d.HttpOnly Cookie  
目前主流的浏览器已经都支持了httponly cookie。1.IE5+ 2.Firefox 1.0+ 3.Opera 8.0+ 4.Safari/Chrome。在支持httponly的浏览器上，设置成httponly的cookie只能在http(https)请求上传递。也就是说httponly cookie对客户端脚本语言(javascript)无效，从而避免了跨站攻击时JS偷取cookie的情况。当你使用javascript在设置同样名字的cookie时，只有原来的httponly值会传送到服务器。  
e.3rd-party cookie  
第一方cookie是cookie种植在浏览器地址栏的域名或子域名下的。第三方cookie则是种植在不同于浏览器地址栏的域名下。例如：用户访问a.com时，在ad.google.com设置了个cookie，在访问b.com的时候，也在ad.google.com设置了一个cookie。这种场景经常出现在google adsense，阿里妈妈之类的广告服务商。广告商就可以采集用户的一些习惯和访问历史。  
f.Super Cookie  
超级cookie是设置公共域名前缀上的cookie。通常a.b.com的cookie可以设置在a.b.com和b.com，而不允许设置在.com上，但是很不幸的是历史上一些老版本的浏览器因为对新增后缀过滤不足导致过超级cookie的产生。

e.Zombie Cookie  
僵尸cookie是指那些删不掉的，删掉会自动重建的cookie。僵尸cookie是依赖于其他的本地存储方法，例如flash的share object,html5的local storages等，当用户删除cookie后，自动从其他本地存储里读取出cookie的备份，并重新种植。  
3.Cookie用途  
a.会话管理  
1.记录用户的登录状态是cookie最常用的用途。通常web服务器会在用户登录成功后下发一个签名来标记session的有效性，这样免去了用户多次认证和登录网站。  
2.记录用户的访问状态，例如导航啊，用户的注册流程啊。  
b.个性化信息  
1.Cookie也经常用来记忆用户相关的信息，以方便用户在使用和自己相关的站点服务。例如：ptlogin会记忆上一次登录的用户的QQ号码，这样在下次登录的时候会默认填写好这个QQ号码。  
2.Cookie也被用来记忆用户自定义的一些功能。用户在设置自定义特征的时候，仅仅是保存在用户的浏览器中，在下一次访问的时候服务器会根据用户本地的cookie来表现用户的设置。例如google将搜索设置（使用语言、每页的条数，以及打开搜索结果的方式等等）保存在一个COOKIE里。

c.记录用户的行为  
最典型的是公司的TCSS系统。它使用Cookie来记录用户的点击流和某个产品或商业行为的操作率和流失率。当然功能可以通过IP或http header中的referrer实现，但是Cookie更精准一些。

4. Cookie的实现  
Cookie是web server下发给浏览器的任意的一段文本，在后续的http 请求中，浏览器会将cookie带回给Web Server。同时在浏览器允许脚本执行的情况下，Cookie是可以被JavaScript等脚本设置的。  
a. 如何种植Cookie

[](http://www.webryan.net/wp-content/uploads/1.png)

http方式:以访问http://www.webryan.net/index.php为例  
Step1.客户端发起http请求到Server

GET /index.php HTTP/1.1  
Host: www.webryan.net  
(这里是省去了User-Agent,Accept等字段)

Step2. 服务器返回http response,其中可以包含Cookie设置

HTTP/1.1 200 OK  
Content-type: text/html  
Set-Cookie: name=value  
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT  
(content of page)

Step3. 后续访问webryan.net的相关页面

GET /spec.html HTTP/1.1  
Host: www.webryan.net  
Cookie: name=value; name2=value2  
Accept: \*/\*

需要修改cookie的值的话，只需要Set-Cookie: name=newvalue即可，浏览器会用新的值将旧的替换掉。

脚本方式种植 Cookie：  
JavaScript或类似的寄宿在浏览器中的脚本语言也可以设置Cookie。在JavaScript里，可以通过document.cookie对象实现。例如：  
document.cookie = “key=newvalue”;  
b. Cookie属性  
除了name=value对以外，我们还可以设置Cookie其他属性以支持更丰富的Cookie需求，这些属性通常是浏览器用来判断如何对待cookie，何时删除、屏蔽或者如何发送name-value对给Server。也就是说无论我们设置了某个cookie的多少属性，这些Cookie属性是不会被浏览器发送回给Server的。  
a) Domain and Path  
作用：定义Cookie的生效作用域，只有当域名和路径同时满足的时候，浏览器才会将Cookie发送给Server。如果没有设置Domain和Path的话，他们会被默认为当前请求页面对应值。 举例如下：

提问下：第一个和第三个Cookie有啥不一样??  
结论：浏览器优先匹配domain,而对于Path字段则是以匹配的方式进行判断。  
对于 1.http://docs.foo.com/accounts/index.html  
2.http://docs.foo.com/accountstest.html  
1.会带上Cookie1,2,3; 2会带上1，2  
b) Expires and Max-Age  
作用：设置浏览器何时删除Cookie  
Expires的规定格式是：“Wdy, DD-Mon-YYYY HH:MM:SS GMT”。  
相对于Expires的精准的时间设置，在RFC 2965中规范提供了一个替代方案：Max-Age:seconds,来设置cookie在设置后多长秒后失效。

Set-Cookie: lu=Rg3vHJZnehYLjVg7qi3bZjzg; expires=Tue, 15-Jan-2013 21:47:38 GMT; path=/; domain=.foo.com; httponly  
Set-Cookie: made\_write\_conn=1295214458; path=/; domain=.foo.com  
Set-Cookie: reg\_fb\_gate=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; path=/; domain=.foo.com; httponly

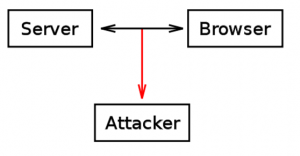
第一个Cookie: 过期时间是2013年1月15日的精确时间；  
第二个：没有设置过期时间，为Session cookie.  
第三个：删除Cookie

C) Secure and HttpOnly  
作用：设置Cookie的安全属性  
特质：Secure和HttpOnly都是没有value字段的。  
Secure字段告诉浏览器在https通道时，对Cookie进行安全加密，这样即时有黑客监听也无法获取cookie内容。  
HttpOnly字段告诉浏览器，只有在HTTP协议下使用，对浏览器的脚本不可见，所以跨站脚本攻击时也不会被窃取。 从前面例子可以看到Google和Facebook都在使用HttpOnly的Cookie。  
5. 浏览器相关  
a) Cookie规范规定浏览器最少支持300个cookie，每个cookie 4kb；每个域名最少20个cookie。  
b) 浏览器都支持删除和禁用cookie  
c) 在浏览器地址栏输入：javascript:alert(document.cookie)可以看到所有cookie  
d) 默认情况下，IE浏览器仅支持设置有P3P “CP” (Compact Policy) 标记的第三方Cookie.

测试方法：  
javascript:for(var i=0;i<100;i++){document.cookie=’cookiename’+i+’=1aaa;’}document.cookie=’test=asdf;’;alert(document.cookie)  
测试结论：ie8:每个域名50个，firefox:每个域名150个；chrome:每个域名170个；ie6、ie7:20个

测试方法：  
javascript:var arr=[],i=5112;while(i){i–;arr.push(‘i’)}document.cookie=’test=’+arr.join(”)+’;';alert($.cookie.get(‘test’).length)  
测试结论：  
ie8:5112+5 = 5117  
但最多10个大字段。字段内容多了的话，导致服务器无法响应。

6.Cookie窃取及会话劫持(hijacking)  
相对很多Session验证方法的缺点和不足，大多数网站都是把Cookie当作用户的唯一标识。在这种情况下，黑客以可以通过窃取用户的cookie来模拟用户的请求行为，但对于服务器来说是无法辨别到底是来自用户还是黑客的。  
下面给出Cookie作为用户标识的风险和安全隐患：  
a. 网络监听

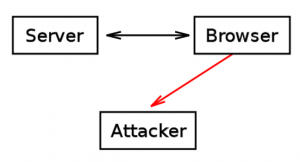
[](http://www.webryan.net/wp-content/uploads/2.png)

在网络上传输的数据都是会被监听获取的，尤其是在公共的、非加密的网络环境（free wifi）。这些数据也包括常规的http（非https加密通道）所有session，当然也就包括了HTTP 会话里的Cookie。当黑客拿到明文的cookie之后就可以模拟用户操作，比如改密码、消费等行为。  
解决这个问题的最根本方法是采取https协议，通过SSL通道对内容及cookie进行加密。此外还有一些二次保护的方法可以作为过渡和折中。

b. DNS cache异常及其他  
通过DNS cache或者DNS服务商的一些漏洞问题（www.baidu.com），黑客可以通过将www.baidu.com的子域名hack.www.baidu.com指向到黑客自己的IP。这样黑客就可以通过方式http://hack.www.baidu.com/a.png图片到公共环境，从而可以获取到baidu.com下的所有cookie，包括设置了HttpOnly属性的Cookie。  
解决办法：1.减少dns无效配置 2.ISP服务商加强自我安全管理。3.通过HTTPS请求对请求进行加密和授权，这样黑客很难从凭证管理中心获得认证，那么用户在操作的时候会收到明显的提示。

c. 跨站脚本XSS—窃取Cookie

由于JavaScript等脚本语言可以读取页面文档内的Cookie值，同时又可以向任意服务器发出任意的请求。综合起来，黑客可以通过脚本将当前文档下的cookie据为己有。如果黑客使用的地址是https://attacker.com/stole.cgi,那么Secure Cookie也将以明文的方式发送个attacker.com.

[](http://www.webryan.net/wp-content/uploads/3.png)

跨站脚本是web安全永久不变的话题。Web开发者有责任去过滤掉恶意代码。同时，HttpOnly Cookies不可以被客户端脚本读取，这就大大降低了Cookie被盗取的风险。

d. 跨站脚本XSS—hijacking  
当黑客可以在www.test.com上插入一段JS脚本的话，那么没有禁用JS的用户很轻易的会收到hijacking攻击。黑客利用用户的浏览器来发出HTTP请求到test.com本身，所以与用户相关的所有cookie都会存在（包括HttpOnly和Secure Cookie）。例如：人人网发生的分享蠕虫。  
对于这种攻击，除了避免跨站脚本漏洞以外，可以采取验证码的方式进行一定程度上的规避。

e. 跨站脚本XSS—代理请求  
老版本的浏览器允许用户使用XMLHttpRequest发出代理请求，黑客可以通过设置代理将本地的cookie全量的发给代理服务器，再从代理服务器转发给原始服务器。当然这是很快被禁止了。  
f. 跨站请求伪造—CSRF  
CSRF主要是黑客将伪造的请求URL放到一个图片或者其他静态资源里，这种成本极低，且传播性和形象力非常大。  
举例：Qzone的签名的修改地址是：http://qzone.qq.com/cgi-bin/modify?nick=123  
那么黑客将其并放到流量很大的论坛或者博客里。那么很多人就在不知情的情况下就执行了某些操作。

7. Cookie的缺点和不足  
a) 被讨论最多的就是隐私问题  
b) Cookie引入的各种安全问题  
c) 与REST软件架构相背离。  
d) 状态不一致，后退导致cookie不会重置。  
c) 过多使用到是HTTP请求流量浪费

8.Cookie的取代方案  
a) window.name  
当前所有浏览器都能通过DOM结构中的window.name存储2-32MB的数据。同时window.name是跨域名，但是只能是在当前tab里使用，每个tab初始化后都是有个空的window.name。Window.name是可以传递对象的，所以我们通过将数据保存在json里进行传递和存储。  
由于window.name不通过网络传送，所以不会存在被窃取的风险。同时所以从某种角度通过window.name进行用户行为分析更为合理，同时又不会像cookie一样引来http流量消耗  
b.)Internet Explorer userData storage (starting IE9, userData is no longer supported)

支持：IE5-IE8  
使用：  
或者，通过脚本来设置：  
object.style.behavior = “url(‘#default#userData’)”  
object.addBehavior (“#default#userData”)  
数据：  
在XP下，一般位于C:\Documents and Settings\用户名\UserData，有些时候会在C:\Documents and Settings\用户名\Application Data\Microsoft\Internet Explorer\UserData。  
在Vista下，位于C:\Users\用户名\AppData\Roaming\Microsoft\Internet Explorer\UserData  
属性：expires 设置或者获取 userData behavior 保存数据的失效日期，不设置则为永久。  
var store = document.documentElement;  
store.addBehavior(‘#default#userdata’);  
var STORE\_NAME = ‘my\_userdata’;  
store.save(STORE\_NAME);  
store.setAttribute(‘a’, 123);  
store.save(STORE\_NAME);  
store.load(STORE\_NAME);  
store.getAttribute(‘a’);  
store.removeAttribute(‘a’);  
store.save(STORE\_NAME);  
c)HTML5特性  
• HTML5 Session Storage  
• HTML5 Local Storage  
• HTML5 Global Storage  
• HTML5 Database Storage via SQLite  
• Storing cookies in RGB values of auto-generated, force-cached PNGs using HTML5 Canvas tag to read pixels (cookies) back out  
• Local Shared Objects (Flash Cookies)  
• Silverlight Isolated Storage  
d)Flash Shared Object  
存在Flash的用户目录  
依赖于flash的安装  
使用数据量大100k，超过的需要用户允许  
简单操作  
var so:SharedObject = SharedObject.getLocal(key);  
so.data.value = value;//return so.data.value

其他相关  
RFC2109:  
1.声明新增了Set-Cookie和Cookie两个HTTP头  
2.延续使用HTTP/1.1的attribute-value对  
3.Server可以在任意HTTP Header中设置cookie  
4.Server可以设置多个Set-Cookie头  
严格之处:  
1.User Agent设置domain必须满足以.开头，且y.x.qq.com不能设置到.qq.com上  
2.User Agent给服务器的是完整的Cookie  
3.Netscape支持 Expires,协议支持max-age:  
域名限制：  
1.rfc要求  
\* at least 300 cookies  
\* at least 4096 bytes per cookie (as measured by the characters  
that comprise the cookie non-terminal in the syntax description  
of the Set-Cookie2 header, and as received in the Set-Cookie2  
header)  
\* at least 20 cookies per unique host or domain name