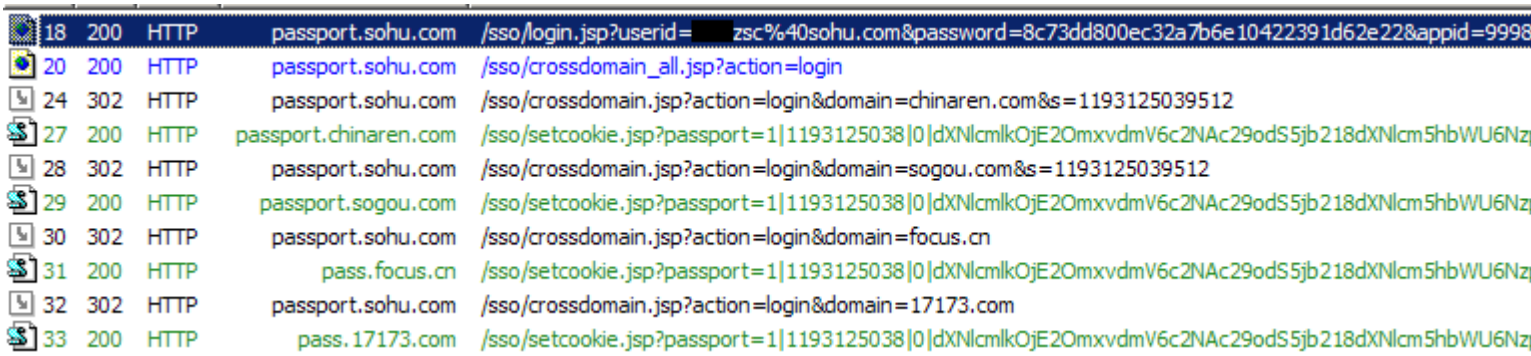
转自:http://www.zhangsichu.com/blogview.asp?Content_Id=89

问题描述：在一个比较复杂的网站环境下。有多个产品向外提供服务。每个产品下都有自己的用户登录界面。现在需要设计一个统一的登录界面。当用户在这个界面登录后就可以自由的使用各个产品和服务。同时意味着用户用一个帐号可以在不同服务里登录，另一方面就是在一个服务里面登录后可以无障碍的漫游到其他服务里面去。

实际应用：Sohu 的 Passport 将 focus.cn，17173.com，sogou.com，chinaren.com 这四个域名下的产品全部整合在一起了。用户在这四个站点中任何一个地方都可以登录。当用户登录后可以自由的使用其他域名下的服务。现在很多网站上都有 bbs blog album 服务。这些服务一般也是自己维护自己的用户信息。当发展到一定时候，也需要一个 Passport 机制整合所有服务，使用户可以单点登录。

Sohu 的实现方案

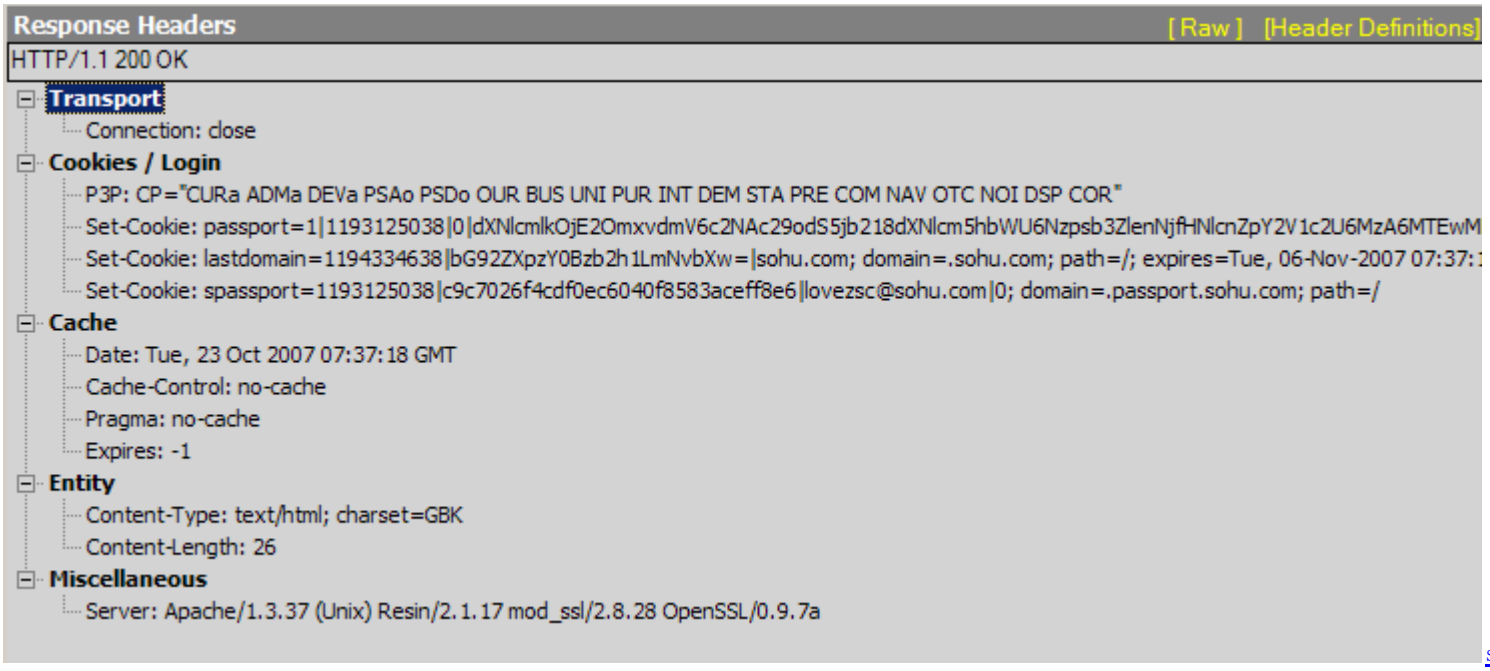
在 http://passport.sohu.com/ 登录后 fiddler 可以拦截到如下的返回信息：



18	200	HTTP	passport.sohu.com	/sso/login.jsp?userid=[redacted]zsc%40sohu.com&password=8c73dd800ec32a7b6e10422391d62e22&appid=9998
20	200	HTTP	passport.sohu.com	/sso/crossdomain_all.jsp?action=login
24	302	HTTP	passport.sohu.com	/sso/crossdomain.jsp?action=login&domain=chinaren.com&s=1193125039512
27	200	HTTP	passport.chinaren.com	/sso/setcookie.jsp?passport=1|1193125038|0|dXNlcmlkOjE2OmxvdmV6c2NAc29odS5jb218dXNlcm5hbWU6Nz
28	302	HTTP	passport.sohu.com	/sso/crossdomain.jsp?action=login&domain=sogou.com&s=1193125039512
29	200	HTTP	passport.sogou.com	/sso/setcookie.jsp?passport=1|1193125038|0|dXNlcmlkOjE2OmxvdmV6c2NAc29odS5jb218dXNlcm5hbWU6Nz
30	302	HTTP	passport.sohu.com	/sso/crossdomain.jsp?action=login&domain=focus.cn
31	200	HTTP	pass.focus.cn	/sso/setcookie.jsp?passport=1|1193125038|0|dXNlcmlkOjE2OmxvdmV6c2NAc29odS5jb218dXNlcm5hbWU6Nz
32	302	HTTP	passport.sohu.com	/sso/crossdomain.jsp?action=login&domain=17173.com
33	200	HTTP	pass.17173.com	/sso/setcookie.jsp?passport=1|1193125038|0|dXNlcmlkOjE2OmxvdmV6c2NAc29odS5jb218dXNlcm5hbWU6Nz

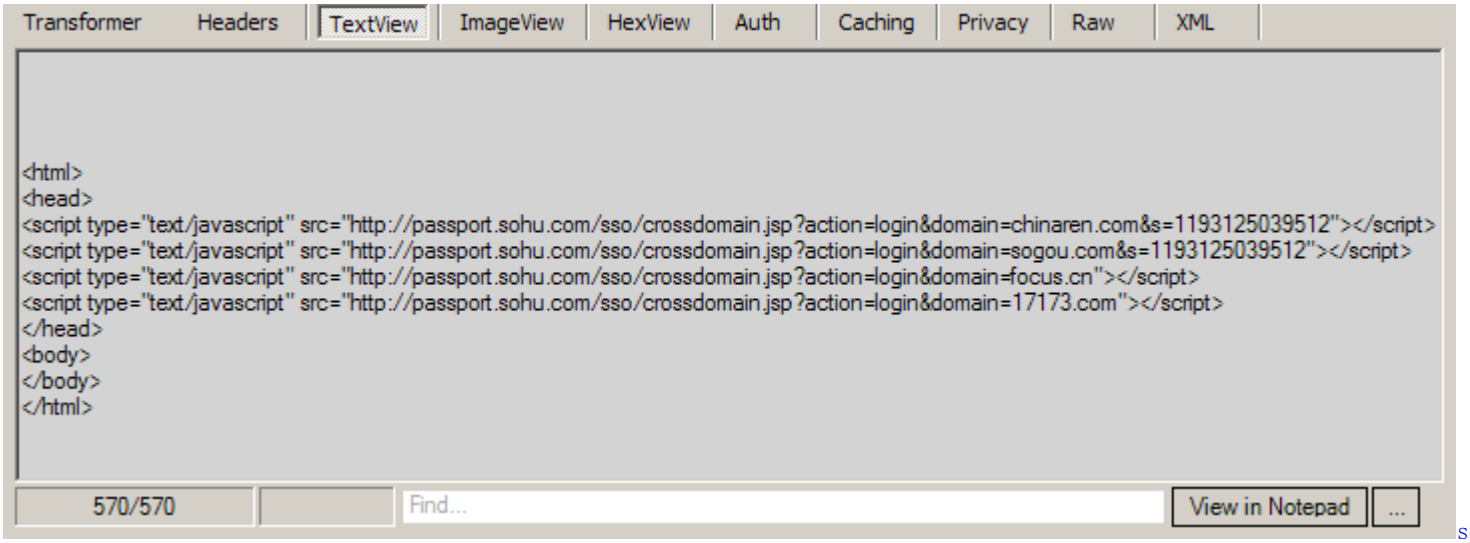
screen.width-300)this.width=screen.width-300″ border=0>

由于 passport.sohu.com 的登录界面使用了 iframe 隐藏提交。所以页面没有看到刷新。隐藏的 iframe 把用户名和加密的 password 和其他信息发送给了 passport.sohu.com。passport.sohu.com 在 Response 中设置了成功登录的 cookie。这个 cookie 可以证实这个用户成功登录了 passport.sohu.com。



sc
reen.width-300)this.width=screen.width-300" border=0>

当用户在 Passport 成功登录后。客户端的 Javascript 根据成功登录的标志，操作 iframe 请求 http://passport.sohu.com/sso/crossdomain_all.jsp?action=login 因为在同一个域名下，没有跨域，在这次请求中，上次成功登陆的 cookie 会被一并带着回去。服务器端检查到成功登录的 cookie 后会 Render 回一段同时登录多个站点的 html。

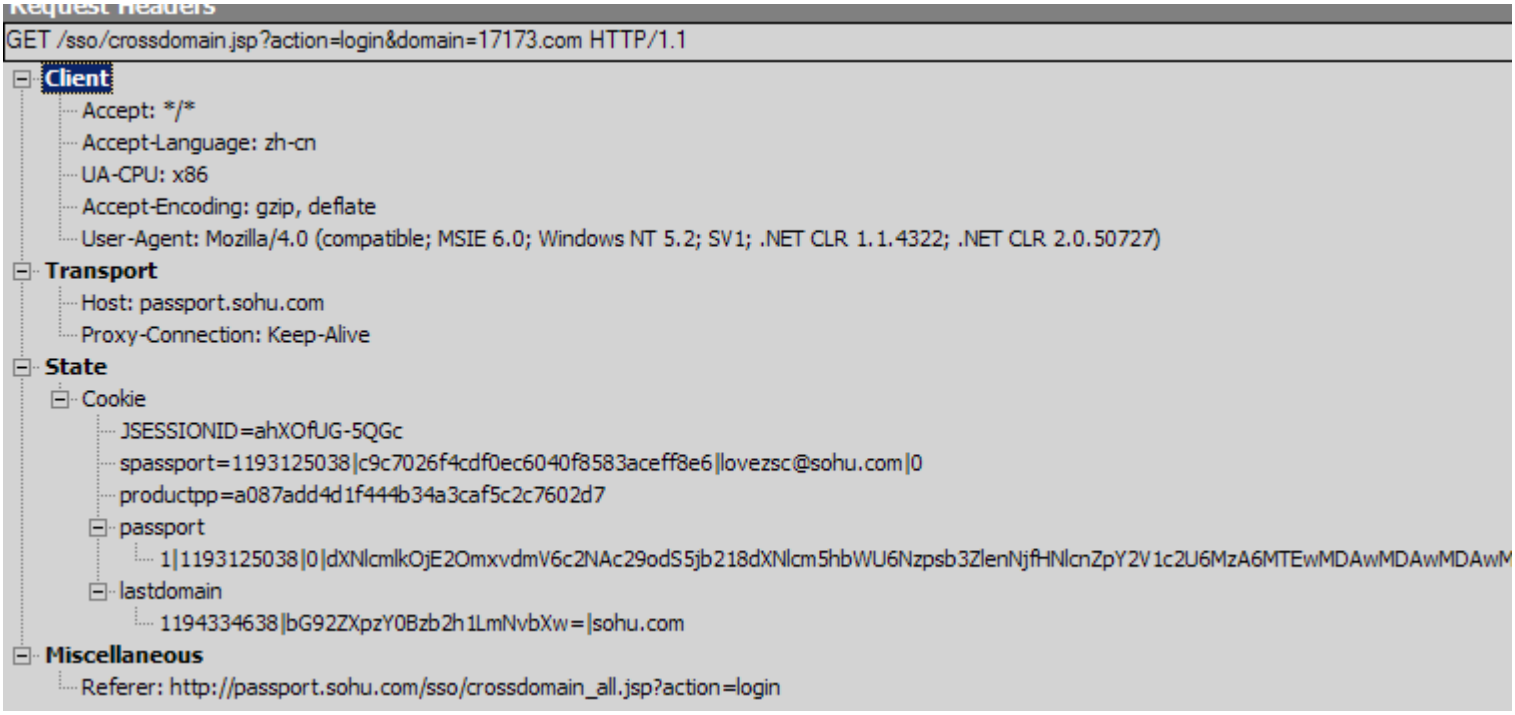


scre
en.width-300)this.width=screen.width-300" border=0>

这段 html 要向 4 个地址发送请求。截至到现在都是在相同的 Domain(passport.sohu.com)请求和返回，为真正的跨站点登录做准备，真正的跨站点登录还没有开始。下面 passport.sohu.com 通过 sso/crossdomain.jsp 在服务器端进行 Redirect 设置 http head 为 302 进行跳转。跳转后在这个跳转后的域名下设置登录成功的 cookie。这就是 sohu 实现跨站点登录的核心过程。下面是 passport.sohu.com 登录 17173.com 的过程。

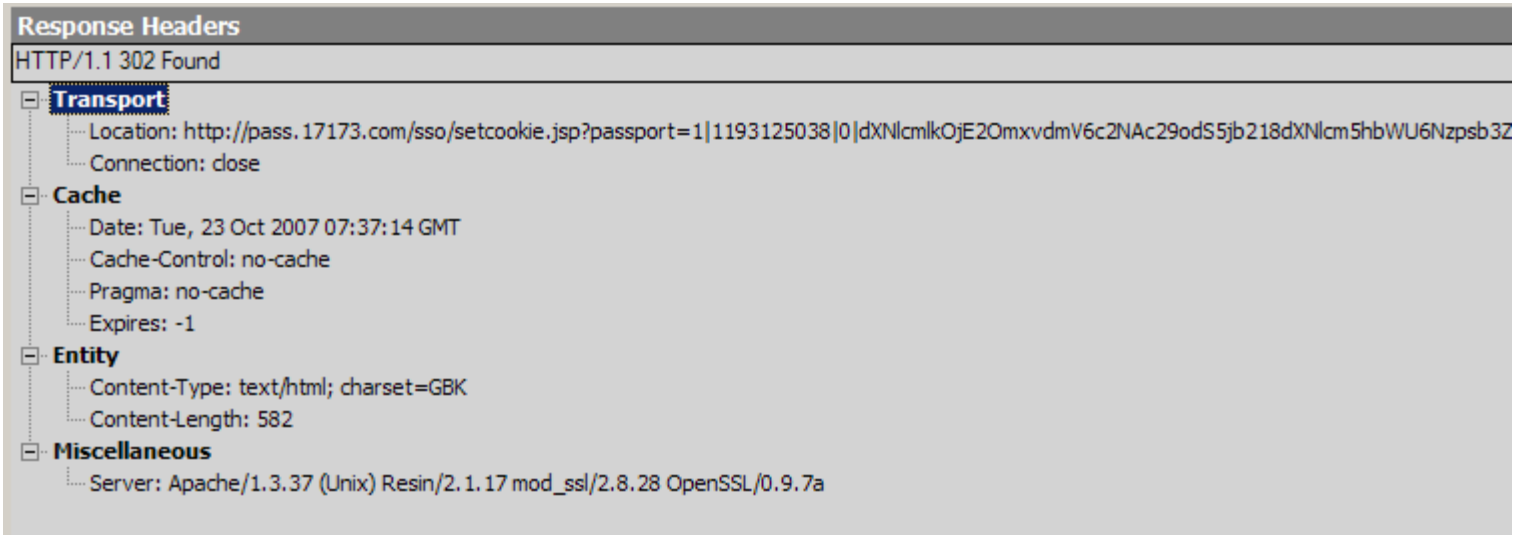
1. 通过 http://passport.sohu.com/sso/crossdomain_all.jsp?action=login Render 回来的 script <script type="text/

javascript" src="http://passport.sohu.com/sso/crossdomain.jsp?action=login&domain=17173.com"></script> 请求同域下的 http://passport.sohu.com/sso/crossdomain.jsp?action=login&domain=17173.com 这时 passport.sohu.com 下成功登录的 cookie 会被带回去。



screen.width-300)this.width=screen.width-300" border=0>

2. 服务器看到成功登录的 Cookie 后。在服务器端计算出一个加密后的 17173.com 的登录 Url，并 Redirect 到这个 Url。



screen.width-300)this.width=screen.width-300" border=0>

3. 17173.com 从 url 的 QueryString 中取得信息。并在 Response 中设置 Cookie。这个 Cookie 终于写到了 17173.com 下。而不是 passport.sohu.com 下。从而使得用户在 17173.com 下登录。其实用户在 17173.com 下手动登录也是写上同样的 Cookie。以后用户再访问 17173.com 的页面时这个 Cookie 会被带回去。这就表示用

户在 17173.com 下成功登录过了。

s
creen.width-300)this.width=screen.width-300" border=0>

经过上面的步骤。用户在 passport.sohu.com 下登录的同时也在其他站点登录了。

在上面的过程中，最核心的技巧就是在指定的域下写入想要的 Cookie：

1. Sohu 使用了在同一个域名登录后通过再次请求这个域名下某个链接后，得到要登录站点的请求 Url，通过 javascript 使隐藏的 iframe 请求要登录站点的 Url，服务器端接到请求 Redirect 到要登录站点，然后通过 Response 写入 Cookie，完成跨域名写 Cookie 的操作。这种写 Cookie 的方式，需要在跳转时对请求的 QueryString 进行加密。接受方需要对 QueryString 进行解密。

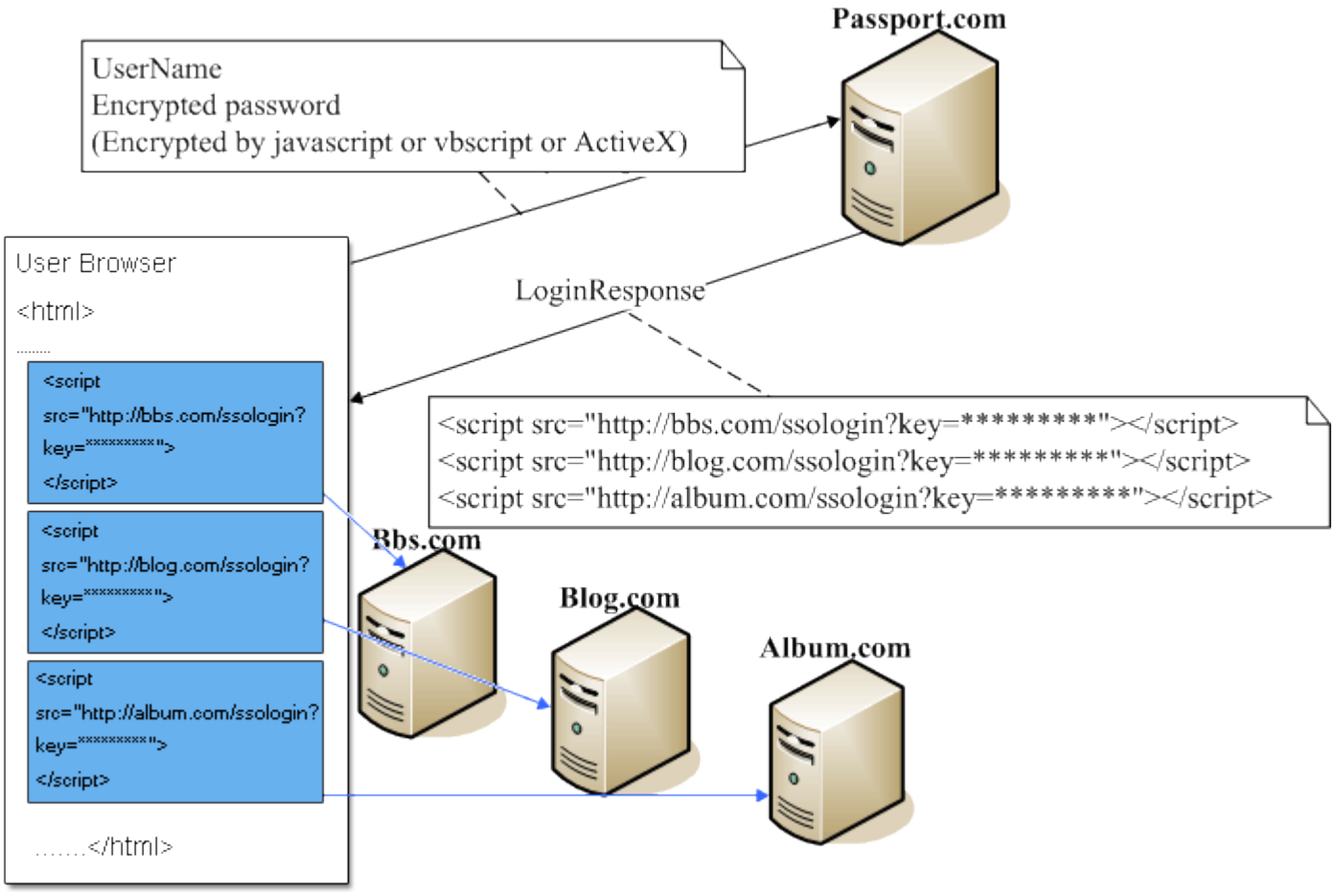
2. 这种做法在服务器端不需要特别的处理。只要写好相应 Post 操作 WriteCookie 操作 Redirect 操作 就可以了。在 FireFox 下就可以正常工作了。但是在 IE 下写 Cookie 的操作还不行，总是写不进去 Cookie。需要在 Response 中加入一段特别的 Header. P3P: CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR"

这个 Http Header 是 P3P 安全的要求。P3P 的详解 http://www.oreilly.com.cn/book.php?bn=7-302-07170-5
微软对这个的解释： http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q323752

一个更加轻量级的方案

Sohu 的通行证方案已经可以轻松的将各个域名下的用户都同步登录了。但是在实现上 Sohu 会让客户端的浏览器请求两次 passport.sohu.com。在第二次得到一个登录多个站点的地址列表。在第三次请求时通过本域下的 cookie 进行身份验证，最后在服务器端跳转到一个含有加密 Key 的其它域名的 Url 地址，最终写入登录成功的 Cookie。除去最开始的登录，要求用户在登录后再进行两次请求。并且服务器端要再做一次跳转。Sohu 的做法可能由于 Sohu 服务器环境和数据存储的结构所决定。

其实总共只需一次登录请求，和每个域名下一次请求就可以完成多站点登录了，同时也不需要服务器端的跳转。



screen.wi

dth-300)this.width=screen.width-300" border=0>

跨站点的请求由 script 的 src 发出。各个域名下的 ssologin 处理 QueryString 中的 key，解密 key，验证 Key 的合法性。在 Response 中写入登录成功的 Cookie。完成跨站点 Cookie 的写入。

两种方案的比较

1. Sohu 使用的登录方式，请求次数多，但是每次请求都有对应的验证过程，在服务端跳转时，重要的跳转 Url 地址在 HttpHeader 中，使得跳转地址更加安全，使得用户在跨域登录时非常安全可靠。

2. 轻量级方案的登录方式，请求次数少，没有服务器端的跳转，对服务器压力小。但是需要对 Key 进行加密解密。跳转的 Url 会在 Response 的 Http Body 中 Render 给用户。在使用轻量级方案的时候，最好在 Key 中加上时间戳，过期时间设置为 3 分钟。Key 过期认为这个 Key 是非法 Key，不在 Response 中写入登录成功的 Cookie。