Web 安全测试之跨站请求伪造(CSRF)篇

本文将简单介绍 CSRF 漏洞，并详细说明造成这种漏洞的原因所在，以及针对该漏洞的黑盒测试与灰盒子测试具体方法和示例，最后提提了一些防范该攻击的建。

跨站请求伪造（即 CSRF）被 Web 安全界称为诸多漏洞中“沉睡的巨人”，其威胁程度由此“美誉”便可见一斑。本文将简单介绍该漏洞，并详细说明造成这种漏洞的原因所在，以及针对该漏洞的黑盒测试与灰盒子测试具体方法和示例，最后提提了一些防范该攻击的建议，希望本文对读者的安全测试能够有所启发。

一、CSRF 概述

我们首先来了解一下什么是跨站请求伪造（CSRF）？跨站请求伪造是一种挟制终端用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。攻击者只要借助少许的社会工程诡计，例如通过电子邮件或者是聊天软件发送的链接，攻击者就能迫使一个 Web 应用程序的用户去执行攻击者选择的操作。例如，如果用户登录网络银行去查看其存款余额，他没有退出网络银行系统就去了自己喜欢的论坛去灌水，如果攻击者在论坛中精心构造了一个恶意的链接并诱使该用户点击了该链接，那么该用户在网络银行帐户中的资金就有可能被转移到攻击者指定的帐户中。

当 CSRF 针对普通用户发动攻击时，将对终端用户的数据和操作指令构成严重的威胁；当受攻击的终端用户具有管理员帐户的时候，CSRF 攻击将危及整个 Web 应用程序。

二、造成 CSRF 的原因

跨站请求伪造能否得逞，与以下几个方面密不可分，分别是浏览器对会话的处理，攻击者对 Web 应用有关 URL 的了解，应用程序赖以管理会话的信息对浏览器的透明性以及各种能够引发资源请求 HTML 标签等。下面分别加以解释。

首先，我们来了解一些 Web 浏览器对于 Cookie 和 HTTP 身份验证信息之类的会话信息的处理方式。目前，浏览器会自动地发送标识用户对话的信息，而无需用户干预，换句话说，当浏览器发送这些身份信息的时候，用户根本感觉不到。假设站点 A 上有一个 Web 应用程序，并且受害者正好已经在该站点上通过了身份认证，这时，站点会向受害者发送一个 cookie 作为响应，这个 cookie 的作用是什么呢？主要是被站点作为用户会话的标志，即如果站点收到了带有受害者的 cookie 的请求，那么它就会把这个请求看作是已登录的受害者发来的。一般情况下，浏览器收到站点设置的 cookie 之后，每当向该站点发送请求的时候，浏览器都会“自动地”连同该 cookie 一起发出。

然后，我们再来讨论一下攻击者对 Web 应用程序 URL 的了解。如果应用程序没有在 URL 中使用跟会话有关的信息的话，那么通过代码分析或者通过访问该应用程序并查看嵌入 HTML/JavaScript 中的 URL 以及表单来了解应用程序有关的 URL、参数和容许值。

接下来，我们讨论一下应用程序赖以管理会话的信息对浏览器的透明性问题。我们知道，为了提高 Web 应用的便利性，用来管理会话的信息，例如 Cookie 或者基于 HTTP 的身份验证（例如 HTTP 基本认证、非基于表单的认证）等敏感信息，都是由浏览器来存放的，并在每当向需要身份验证的应用程序发送请求时自动捎带上这些信息。也就是说，浏览器可以访问会话管理信息，如果 Web 应用程序完全依赖于这类信息来识别一个用户会话，这就为跨站请求伪造创造了条件。

上面所说的三个因素，是跨站请求伪造攻击的必要的条件，而下面所说的，是一个“锦上添花”的因素，即没有它也能发动跨站请求伪造攻击，但是有了它能使该攻击更加容易。这就是存在多种 HTML 标签，如果页面内出现这些标签，会立刻引起浏览器对 http[s]资源的访问，例如图像标签 img 便是其中之一。

为简单起见，我们这里讨论 GET 方式的 URL（不过这里讨论的内容同样适用于 POST 请求）。如果受害者已经通过身份验证，那么当他提交其它请求时，该 cookie 也会自动地随之发送（如图，这里用户正在访问 www.example.com 上的一个应用程序 ）。

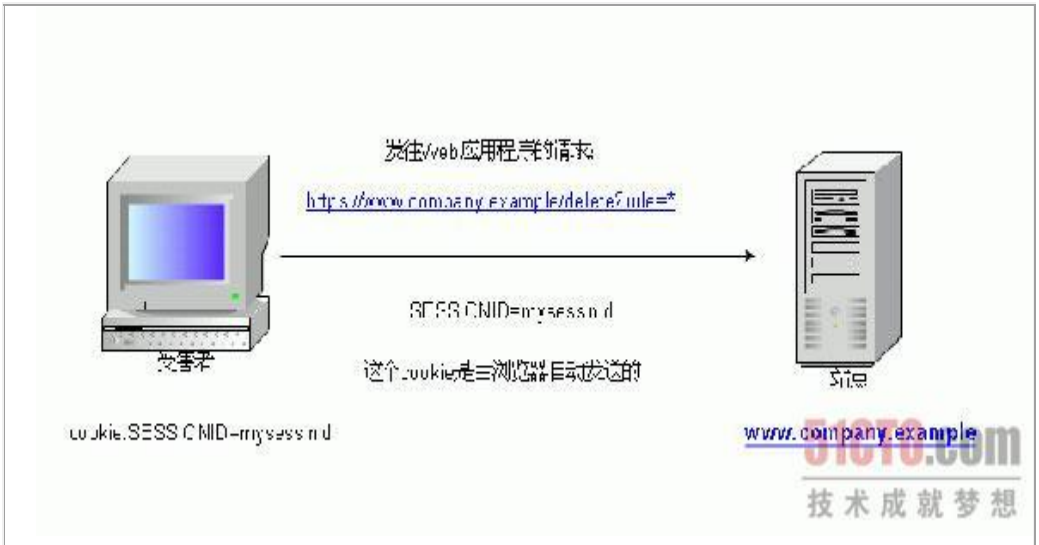

图 1　浏览器发送请求的同时还自动发送 cookie

那么，什么情况下会引起发送这个 GET 请求呢？这可就有多种可能了，首先当用户正常使用该 Web 应用程序的过程中有可能引发这个 GET 请求；其次，当用户直接在浏览器地址栏中键入该 URL 时也会引发该 GET 请求；再者，用户单击了指向该 URL 的链接，即使该链接位于该应用程序外部，也会引发该 GET 请求。

对于应用程序来说，它是无法区分上面的这些差别的。特别是第三种可能是非常危险的。有许多技术（和漏洞）可以隐藏一个链接的真实属性。链接可以嵌入电子邮件消息中，也可以出现在存心不良的 Web 站点，然后引诱用户浏览该站点，例如链接出现在位于其他主机上（其它 Web 站点、HTML 格式的电子邮件消息，等等）的内容中，并且指向应用程序的资源。如果用户单击了该链接，由于他已经通过了站点上 Web 应用程序的认证，所以浏览器就会发出一个 GET 请求给该 Web 应用程序，同时将验证信息（包含会话 id 的 cookie）一并发过去。这样会导致在 Web 应用程序上执行一个有效操作——该操作可能不是该用户所想要的，例如一个引起在网络银行上进行转帐恶意的链接，等等。

如前所述，通过使用诸如 img 之类的标签，甚至不需要用户点击具体的链接就能发动攻势。假设攻击者向用户发送了一封电子邮件，诱骗用户访问一个 URL，而该 URL 则指向一个包含下列 HTML 内容（注意，内容已作精简）的页面：

```
 [html][body]

...

(img src="https://www.company.example/action" width="0"
height="0")

...

[/body][/html]

用时将[]换成<>
```

当浏览器显示该页面时，它也将设法显示这个指定宽度为 0 的图像，即该图像是不可见的——这会导致自动向站点中的 Web 应用程序发送一个请求。要紧的是，浏览器不管该图像 URL 实际是否指向一个图片，只要 src 字段中规定了 URL，就会按照该地址触发一个请求。当然，这里有一个前提，那就是浏览器没有禁止下载图像——实际上浏览器都配置成允许下载图像，因为禁用图像后大多数 Web 应用程序的可用性就会大打折扣。与跨站请求伪造有关的 HTML 标签问题是总结如下：

页面中有许多标签会导致自动发出 HTTP 请求（img 标签便是其中之一）；

浏览器无法断定 img 标签所引用的资源到底是不是图像以及是否是有害的；

当加载图像时，根本就不考虑所涉及的图像所在的位置，即表单和图像不必位于同一个主机上，甚至可以不再同一个域内。虽然这是一个非常便利的特性，但是却给应用程序的隔离制造的障碍。正是由于与 Web 应用程序无关的 HTML 内容可以引用应用程序中的各种组件，以及浏览器可以自动为该应用程序构造一个有效的请求这两个事实才导致了这种攻击的出现。这意味着，正确的 URL 必须包含用户会话有关的信息，而攻击者却无法得知这些信息，因此不可能识别这样的 URL。

对于集成了邮件/浏览器功能的工作平台，跨站请求伪造问题可能更为严重，因为，仅仅显示一封包含该图像的电子邮件就会导致请求及有关浏览器 cookie 一起向 Web 应用程序发去。

另外，攻击者还可以对这些东西进行伪装，例如引用貌似合法的图像 URLs，例如

```
(img src="https://[attacker]/picture.gif" width="0" height="0")

用时将()换成<>
```

这里，[attacker]是攻击者控制下的站点，并通过重定向机制将 http://[attacker]/picture.gif 转向 http://[thirdparty]/action。

如果Web应用程序的会话信息完全靠浏览器来提供的话，那么这样的Web应用程序也都易于受到攻击，其中包括那些仅依赖于HTTP身份验证机制的那些应用程序，因为浏览器知道这些验证信息，并会在发送每个请求的时候自动附带这些验证信息。当然，这不包括基于表单的认证，它只发出一次，并且生成了有关会话信息的表单——当然，如果这样的信息就像cookie那样进行简单的传递的话，这就有回到了之前的情形。

三、跨站请求伪造情景分析

假如受害者已经登录到一个防火墙的Web管理应用程序上。我们知道，当用户登录时，Web应用会进行身份验证，通常是要求用户提供其用户名和密码，如果用户名和密码正确则会通过身份认证；随后，Web应用会在客户端存放一个小文本文件，即cookie来存放会话信息。

假设防火墙有一个基于Web的管理接口，我们称之为防火墙Web管理程序，其中具有一个这样功能或者说一个页面，即允许认证的用户通过规则编号删除指定的规则，或者通过输入“*”删除全部已配置的规则——这的确是一个非常危险的功能。该删除页面显示如下。假如该表单（我们已经对其进行了简化）引起一个GET请求，那么该请求将如下所示

https://[target]/fwmgt/delete?rule=1
(删除一号规则)
https://[target]/fwmgt/delete?rule=*
(删除全部规则)。

该范例是故意十分天真的，这是为了便于说明CSRF的危险性。删除防火墙规则的页面如下所示：

图2 防火墙管理页面

因此，如果我们键入值“*”，并按下删除按钮，就会提交下列GET请求：

https://www.company.example/fwmgt/delete?rule=*

其结果当然是删除所有防火墙规则了，呵呵，后果很严重呀！如下图所示：

图 3　删除所有防火墙规则

实际上，这并非唯一可能的情形。用户也可以手动提交 URLhttps://[target]/fwmgt /delete ?Rule =*来达到相同的效果。或者通过单击一个直接指向以上 URL 的链接或通过重定向到达以上 URL 的链接。或者同样也可访问一个嵌入了指向相同的 URL 的 img 标签的 HTML 页面。在这些情况下，如果用户当前已经登录到防火墙管理程序，那么该请求将得逞并会修改防火墙的配置。

此外，我们还可以设想攻击是针对敏感的应用程序、进行自动的拍卖投标、转帐、订货、改变关键软件组件的配置等等。

更有趣的是，这些漏洞都可以在防火墙之后进行利用，即只要被攻击的链接是受害者所能访问的即可，而不必非得攻击者能够直接访问才行。

尤其是，它可以是任何内部网 Web 服务器，例如，之前提到的防火墙管理工作站，它大不可能暴露于国际互联网。对于那些既可以用作攻击矢量，有是攻击目标的应用程序(例如 web 邮件应用程序)，情况就更糟了：如果这样的应用程序是易受攻击的，那么用户阅读包含 CSRF 攻击的信件时很明显已经登录到程序了，它会以 web 邮件应用程序为目标，并让邮件应用程序执行删除信件、发送消息等动作，而这些动作看起来将像是用户本身所做的。

四、黑盒子测试

为了进行黑盒测试，需要知道受限制的(认证的)区域的 URL。如果您具有有效证书，那么就可以扮演攻击者和受害者这两种角色了。在这种情况下，仅仅通过浏览应用程序您能获悉受测试的有关 URLs。否则，如果没有有效的证书可用的话，必须组织一个实际的攻击，以引诱一个合法的已登录的用户来点击某个适当的链接，这可以通过社会工程来进行。

不管怎样，测试案例可以如下构造：

把 u 改成受测的 URL，例如 u=http://www.example.com/action

构造一个 HTML 页面，让它包含引用 url u（规定全部相关参数；使用 GET 方式的时候很简单，如果使用的是 POST 请求，则需要诉诸于一些 JavaScript 代码）的 HTTP 请求；

确保合法的用户已经登录该应用程序；

引诱他点击一个链接，而该链接指向受测试的 URL（如果你无法冒充用户的话，则需要借助于社会工程）；

观察结果，如检测 Web 服务器是否执行了该请求。

五、灰盒子测试

对应用程序进行安全评估的时候，要调查其会话管理是否是易受攻击的。如果会话管理完全依赖于客户端的值（即对于浏览器来说也是可用的），那么该应用程序是易受攻击的。通过这些客户端的值，我们就弄明白了 Cookie 和 HTTP 认证证书（基本认证及其他形式的 HTTP 认证；不是基于表单的认证,即应用程序级别的认证)。对于没有此弱点的应用程序来说,它在 URL 中必须包括跟会话 session 有关信息，并且要采取一种令用户无法辨认或者不可预见的形式([3]使用术语 secret 来表示这种信息单元)。

可以经由 HTTP 的 GET 请求访问的资源很容易受弱点的影响,但是 POST 请求可以通过 JavaScript 实行自动化，并且也易于受到攻击；因此，单独使用 POST 不足以防止 CSRF 漏洞的发生。

六、跨站请求伪造对策

跨站请求伪造的危害非常之大，所以无论是用户还是开发人员都应该引起足够的重视，下面是给 Web 应用程序终端用户和 Web 应用程序开发人员的一些有用的建议。

用户

因为 CSRF 漏洞有流行之趋势，所以建议用户遵循最佳实践来降低风险，可以降低风险的习惯包括：

使用 Web 应用程序之后立即登出

不要让浏览器保存用户名/口令，也不要让站点“记住”您不要使用同一个浏览器同时访问敏感的应用程序和随意冲浪；如果必须同时做多件事情的话，最好单独使用不同的浏览器。

对于支持 HTM 格式邮件/浏览器集成是式软件以及集成了新闻阅读程序/浏览器的软件都会带来额外的风险,因为只要查看邮件或者新闻就有可能被迫执行一次攻击,所以使用这类软件时格外小心。

开发人员

开发人员应当向 URL 添加跟会话有关信息。该攻击类型之所以得逞，是因为会话是由 cookie 唯一标识的，并且该 cookie 是由浏览器自动发送的。

如果我们在 URL 级别为会话生成其它相关信息，那么就会给攻击者为发动攻击而了解 URL 的结构造成更多的障碍。

至于其它的对策，虽然也无法解决该问题，但是能够使得利用该漏洞更加困难，例如使用 POST 而不是 GET。虽然 POST 请求可以通过 JavaScript 进行模仿，但是它提高了发动这种攻击的难度。使用中间确认页也能带来相同的效果，比如“您确信要这样做吗？”之类的页面。虽然攻击者可以绕过这些措施，但是这些措施提供了实施攻击的难度。因此，不能完全依赖这些手段来保护您的应用程序。自动登出机制也能减轻这种攻击带来的危害，但这最终依赖于具体情况（一个整天跟有这种漏洞的网络银行程序打交道的用户所面临的风险要远远大于临时使用同一网络银行的用户所面临的风险）。

七、小结

跨站请求伪造，即 CSRF，是一种非常危险的 Web 安全威胁，它被 Web 安全界称为“沉睡的巨人”，其威胁程度由此“美誉”便可见一斑。本文不仅对跨站请求伪造本身进行了简单介绍，还详细说明造成这种漏洞的原因所在，以及针对该漏洞的黑盒测试与灰盒子测试具体方法和示例，最后提提了一些防范该攻击的建议，希望本文对读者的安全测试能够有所启发。