基于跨站脚本的网络漏洞攻击与防范

Attack and Prevention on Defects of Web Page Based on Cross Site Scripting

吴耀斌 王 科 （中南大学信息科学与工程学院 长沙 410083）
龙岳红 （中南大学信息物理工程学院 长沙 410083）

摘 要： 目前很多的Web应用程序为了提高用户体验都包含大量动态内容，从而给Web应用程序带来了安全隐患。而跨站脚本攻击（XSS）是目前安全漏洞中排在前列并引起广泛注意的安全隐患之一。本文主要分析跨站脚本攻击漏洞存在形式和攻击产成流程，并总结出XSS攻击的防范方法。

关键词： XSS 过滤 会话标志符 存取权限

1 引言

2007年7月9号安全组织发布2007年十大Web安全漏洞，利用网页及Cookies写作漏洞的跨站脚本攻击（XSS）登上首位。随后开放Web软件安全计划（Open Web Application Security Project，OWASP）台湾分会于14日也发表2007十大Web安全漏洞，年初曾发生在知名文件阅读器Adobe Acrobat Reader上的跨站脚本攻击（Cross Site Scripting，XSS）同样位居榜首。跨站脚本攻击XSS又称CSS——Cross Site Scrip——指的是恶意攻击者往Web页面里插入恶意Html代码，当用户浏览该页之时，嵌入其中Web里面的Html代码会被执行，从而达到恶意用户的特殊目的。

现在的网站为了提高用户体验包含大量的动态内容，程序比过去要复杂得多，而这些动态内容给XSS攻击提供了可能。所谓动态内容，就是根据用户输入的数据，Web应用程序能够输出相对应的内容。当攻击者输入的数据包含了恶意的Html代码时，就会产生XSS攻击。而静态站点（不允许用户输入），则完全不会存在跨站脚本攻击。

2 XSS的常见形式

Web页面经常在应用程序的某个地方对用户的输入进行回显。一般而言，在预先设计好的某个特定域中输入的纯文本才能回显。但是Html并不仅仅支持纯文本，还可以包含多种客户机端的脚本代码来完成许多操作，诸如验证表单数据，或者提供动态的用户界面元素。在Web应用程序过程中，当用户提交数据与服务器进行交互时，就有可能存在跨站脚本攻击。对Web应用程序进行跨站脚本攻击主要有以下两种形式。

（1）存储好的XSS。攻击者最常检查的XSS漏洞就是那些让用户输入信息，以供其他浏览此页的用户进行检查的地方，括留言，评论，或者博客上的评论。如果将来其他用户浏览此Web页面，应用程序就从存储单元搜集数据，并且把它显示出来。采用这种回显静态数据的方法通常是有害的，攻击者可以用脚本代替静态的数据。攻击者通常都将脚本直接输入到被攻击站点的表格域中。此时脚本代码被留在被攻击的服务器。当其他用户访问这个页面，这些脚本就开始执行。

（2）反射的XSS。反射的XSS就是将脚本嵌入到Url地址的CGI参数，此时，攻击者可以通过电子邮件将一个链接发送到潜在的受害者；当被攻击者点击链接时，页面被下载，但是其中的内容被嵌入在Url中的脚本修改了。这种攻击不需要将脚本存放在受攻击的站点中，因为脚本是在用户不幸的点击了那个被篡改过的链接时，页面载入过程中才完成执行。

3 XSS形成过程

3.1 寻找跨站漏洞

首先我们得找到有跨站漏洞的Web程序，这类程序很多，例如论坛、博客、留言版程序，其中很大一部分

都存在跨站漏洞。Html 支持了很多种脚本嵌入页面中的机制，最常见的就是用 < script > … < /script > 标签。查看代码是否成功执行，最简单的办法就是使用 alert 函数。在表单中输入如下代码：

< script > alert('test') < /script >

运行程序看是否弹出消息框。假如弹出消息框，则存在明显的 XSS 漏洞。

有些 Html 标签也容易被脚本输入所攻击。例如说，一个 image 标签

< img src = " path/to/image. gif" >

可以修改为：

< img src = javascript: alert(" xss") / >

其他标签，如 < frame > 、< html > 、< body > 、< applet > 、< img > 、< iframe > 、< frameset > 、< layer > 、< meta > 、< embed > 、< object > 以及 < style > 可能引发 XSS 问题。下表给出了常见出现的跨站脚本示例。

示例	描述
< img src = " javascrip&#116&#58alert(/xss/) " >	利用 Html 支持 &#ASCII，逃避过滤
< img src = " javas cript: alert(/xss/) " >	中间一个 Tab，把关键字拆分。很多程序员忘记了 Tab 这类的特殊的字符
< img src = " #" onerror = alert('xss') >	图片没有正常输出便会触发这个事件
< div ondragstart = " alert ('xss') " > 1234 < /div > 没有对代码中的 ondragstart	件进行过滤
< img src = " java/ * xss * /script: alert('xss') " >	浏览器执行 Html 文档时，不执行注释，这样可以漏掉对 javascript 的过滤
< body onload = " javascript: alert('xss') " >	使用一个事件触发脚本执行
< link rel = " stylesheet" href = " http: //... " >	使用动态的 href

3.2 利用漏洞进行攻击

攻击者的目标不是弹出消息框，而是要得到会话的标志符和存取权限。应用程序区分不同访问者，唯一的途径就是给每个用户分配一个独一无二的标志符，因为 Http 本身是无状态的。而在 XSS 中，攻击者无须知道这些标志符是怎样的产生的，只是简单的将已有的“偷取过来”。通过使用其他用户的标志符，攻击者就可以伪装自己的身份，完成自己想要的操作。

客户机端的脚本被某些特定的操作和它们可以存取的资源所限制。其中就包括浏览器正在载入的页面。这时，没有弹出静态文字提示。通过以下代码，脚本显示任何的 Cookie < script > alert (document. cookie) < /script >

对攻击者来说。把 Cookie 显示出来，没有利用价值。因为多数的 Cookie 都是经过加密的，如果想 Cookie 欺骗的话，同样也要受到其它的条件的限约。攻击者目的是把这些数据远程收集起来。最常用的方法是在页面上放置一幅链接到其他站点的图像。

```
< script >
    Document. write ( " < img src = http: //evilhacker.
com/ >
    px. gif? cookie = " + document. cookie)
< /script >
```

这样就可以得到的图像是一幅 1 * 1 像素的透明. gif 文件。用户可能无法发现，当用户请求获取图像时，会产生一个“Cookie”CGI 参数。尽管服务器不再需要这个参数来重新获取图像，它还是保存在服务器的日志文件中。攻击者查看日志纪录，在其中寻找对 px. gif 文件的请求。这样就获得了合法的会话标识符。

当不存在会话标识符时，页面仍然可以被攻击。以新闻站点为例，攻击者不会对 Cookie 感兴趣，因为在这种站点中，并没有对攻击者来说感兴趣的东西，也没有独一无二的用户验证才能获的敏感的数据或进行操作。但是，攻击者也可以用页面中的信息诱导用户完成正常情况下不会去做的事情。这其实是利用了用户对这些站点的信任。

此时，并没有使用客户机端的脚本从站点上获取信息，相反的，却向站点输出信息，修改其上已经存在的内容。比如说，当一个普通用户打开一个 Web 页面，攻击者利用到了受害者的客户机端的代码对当前载入页面的存取权限，完成对其修改。下面代码修改了第 30 幅图像的位置：

```
< script >
    Document. images [ 30 ]. src = http: //evilhacker.
```

com/msft. gif
</script>

使页面上的一副图像被替换成 msft. gif，用来欺骗用户使之做一些诸如卖掉 Microsoft 的股票等等。攻击者对页面的篡改不局限于图像，甚至可以重写整个页面。2004 年 jibbering 的 jim ley 就利用了 Google 中的 XSS 漏洞，修改了 Google 的整个页面。XSS 代码是通过修改一个 CGI 参数实现，而这个 CGI 参数就包含在通过电子邮件发送给用户的链接中。

4　XSS 防范对策

跨站脚本攻击相对于其他网络漏洞攻击而言显得更隐蔽，也更难防范。XSS 攻击的产生过程是一个用户和 web 程序交互的过程，既有程序本身的问题也有客户端用户的因素。而重点的防范应放在 web 程序编码的防范上，因为很多时候用户无法辨认程序是否安全，但是用户的警觉会减少 XSS 的产生。因此防范跨站脚本攻击我们得从两方面入手，既要求开发者编码过程进行检测防范也要求用户浏览时采取相应防范手段。

4.1　编码过程中的应对措施

(1) 过滤用户提交数据中的代码。这种方法的实施过程比较复杂，不仅仅需要考虑 <script> … </script> 标签，各种可能的 XSS 攻击载体都要考虑进来。任何脚本语言或代码输入，包括 HTML，JavaScript、VbScript、Java，ActiveX、Flash 等等，这些都被称 XSS 攻击的载体。与其过滤掉数据中不应该被接受的部分，倒不如只是将接受的数据过滤(允许通过)。也就是说白名单和黑名单的区别。事实上，保存一份黑名单，纪录所有非法输入的各种编码形式并对它进行不断更新，几乎是不可能的。更好的方式就是只允许合法的数据通过。

(2) 不再将数据看成代码。在 Html 中所有的代码都包含在 <… > 中。因此，当把编码后的尖括号(&lt 和 &gt)放到数据段中时，代码就不会看成是数据了。但是当应用程序需要对文本进行格式上的区分时，就会造成一点小麻烦。比如，在公告牌系统中，用户需要使用黑体，下划线和斜体，还需要插入图像和超链接。采用处理方式是：使用其他格式标识(正如 BBCode 那样—[i]…[/i] 用来标识斜体，而不是用 …)，这样很容易转换为 HTML 代码。

(3) 限制输入字符的长度。对于一些可以进行跨站攻击的表单对象中，我们可以限制其输入字符的长度。跨站脚本代码往往较长，如果限制了输入字符长度，也可以起到很好的防范作用。这个功能可以在数据库中设置，也可以编写一段程序代码来实现。

(4) 限制用户上传 Flash 文件。使用 flash 文件进行跨站攻击可谓防不胜防，如果不能检测用户上传的 Flash 文件的安全性，索性限制用户上传 Flash 文件，以彻底阻断 Flash 跨站攻击的途径。

4.2　客户端用户的应对措施

(1) 慎重点击网站上的链接。保护用户的最好方法就是仅点击你想访问的那个网站上的链接。例如，如果你访问了一个网站，该网站有一个链接指向了 baidu，那么不要单击该链接，而是访问 baidu 的主站点并使用搜索引擎查找相关内容。这样可以杜绝 90% 以上的 XSS 攻击。有时候 XSS 会在你打开电子邮件、打开附件、阅读留言板、阅读论坛时自动进行。当你打开电子邮件或是在公共论坛上阅读你陌生人的帖子时一定要注意。最好的解决办法就是关闭浏览器的 JavaScript 功能。

(2) 提高 IE 浏览器的安全等级。个人用户防范跨站攻击有一定的难度，但仍可以通过设置来降低被攻击的概率。运行 IE 浏览器，选择“工具”菜单→“Internet? 选项”，切换到“安全”标签，将安全级别设置为高，同时可以在“自定义”中进行详细的设置，将一些不需要运行的脚本禁用。

(3) 增强安全意识和防范措施。安装杀毒软件是必须的，碰到那些插入网页木马的跨站攻击，即使跨站成功，我们运行了网页木马，杀毒软件仍能在最后一步将木马拦截下来。同时要在不同的 Web 应用程序中使用不同的密码，即使黑客通过跨站攻击获取了你的 Cookie，并破解出了你的密码原文，这样你损失的也只是一个账户而已，不至于全军覆没。

5　结束语

目前已有一些厂商对 XSS 攻击采取了应对措施。如 Microsoft 提出了一种针对利用 XSS 在 Internet Explorer 中盗取 Cookie 的技术，被称为 HTTP – ONLY。

(下转第 44 页)

(上接第 40 页)

HTTP – ONLY 并不能阻止 XSS 攻击,但是它能在特定的浏览器下封闭 XSS 多种攻击途径中一种。目前还没有其他的浏览器厂商提出自己的解决方案。此外很多专业人员也已开始进入对 XSS 各个方面的研究,如对 SQL 注入技术和跨站脚本攻击的检测的研究,UBB 的跨站脚本攻击的漏洞研究,对跨站脚本攻击隐藏的研究等等,因此将来会有更好更全面的解决方案来应对 XSS 带来的问题。

参考文献

1 XSS (Cross Site Scripting) Cheat Sheet [EB/OL]. http://ha. ckers. org/xss. html

2 汪青青,Web 入侵安全测试与对策[M],北京. 清华大学出版社,2006 年 10 月.

3 王辉、陈晓平、林邓伟,关于跨站脚本问题的研究[J],计算机工程与设计,2004,25(8):1317 – 1319.

4 古开元、周安民,跨站脚本攻击原理与防范[J],网络安全技术与应用,2006,12:19 – 21.