搜狗实验室技术交流文档

Vol 2:3 XSS 跨站脚本攻击及防范

摘要

XSS(Cross Site Script)跨站脚本攻击。它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。本文介绍了该攻击方式，并给出了一些防范措施。

原理

XSS 属于被动式的攻击。攻击者先构造一个***跨站页面***，利用 script、<IMG>、<IFRAME>等各种方式使得用户浏览这个页面时，触发对被攻击站点的 http 请求。此时，如果被攻击者如果已经在被攻击站点登录，就会持有该站点 cookie。这样该站点会认为被攻击者发起了一个 http 请求。而实际上这个请求是在被攻击者不知情的情况下发起的，由此攻击者在一定程度上达到了冒充被攻击者的目的。精心的构造这个攻击请求，可以达到冒充发文，夺取权限等等多个攻击目的。在常见的攻击实例中，这个请求是通过 script 来发起的，因此被称为 Cross Site Script。

攻击 Yahoo Mail 的 Yamanner 蠕虫是一个著名的 XSS 攻击实例。Yahoo Mail 系统有一个漏洞，当用户在 web 上察看信件时，有可能执行到信件内的 javascript 代码。病毒可以利用这个漏洞使被攻击用户运行病毒的 script。同时 Yahoo Mail 系统使用了 Ajax 技术，这样病毒的 script 可以很容易的向 Yahoo Mail 系统发起 ajax 请求，从而得到用户的地址簿，并发送病毒给他人。

XSS 攻击主要分为两类：一类是来自内部的攻击，主要指的是利用 WEB 程序自身的漏洞，提交特殊的字符串，从而使得跨站页面直接存在于被攻击站点上，这个字符串被称为跨站语句。这一类攻击所利用的漏洞非常类似于 SQL Injection 漏洞，都是 WEB 程序没有对用户输入作充分的检查和过滤。上文的 Yamanner 就是一例。

另一类则是来来自外部的攻击，主要指的自己构造 XSS 跨站漏洞网页或者寻找非目标机以外的有跨站漏洞的网页。如当我们要渗透一个站点，我们自己构造一个跨站网页放在自己的服务器上，然后通过结合其它技术，如社会工程学等，欺骗目标服务器的管理员打开。这一类攻击的威胁相对较低，至少 ajax 要发起跨站调用是非常困难的。

实战

我们来看一个简单的攻击实例，下表给出了一个简单的网站 http://10.10.67.25:8080/testxss，该网站的密码和用户名相同，普通用户可以修改 user value，当以 admin 身份登陆时可以通过向 doadmin.jsp 发起请求来修改 admin value。

index.jsp

```
<html>
<body>
    <textarea rows="3" cols="100" readonly="on">
Current User: ${username}
Admin Value: ${adminvalue}
User Value: ${uservalue}
    </textarea>
    <br>
    <a href="login.jsp"/>logout</a><br>
    Login:<br>
    <form action="login.jsp" method="post">
        username: <input type="text" name="u"></input> <br>
        password: <input type="text" name="p"></input> <br>
        <input type="submit" /> password == username :-)
    </form>
    <form action="doadmin.jsp" method="post">
        adminvalue: <input type="text" name="v"></input> <br>
        <input type="submit" />
    </form>
    <form action="doadmin.jsp" method="post">
        uservalue: <input type="text" name="v2"></input> <br>
    <input type="submit" />
    </form>
</body>
```

login.jsp

```
<%
    String u = request.getParameter("u");
    String p = request.getParameter("p");
    if (u != null && p != null && u.equals(p)) {
        session.setAttribute("username", u);
    } else {
        session.removeAttribute("username");
    }
    response.sendRedirect("index.jsp");
%>
```

doadmin.jsp

```
<%
        String u = (String)session.getAttribute("username");
        String v = request.getParameter("v");
        String v2 = request.getParameter("v2");
        if (u != null && u.equals("admin")) {
                if (v != null)
                        application.setAttribute("adminvalue", v);
```

```
        }
        if (u != null && v2 != null)
                application.setAttribute("uservalue", v2);
        response.sendRedirect("index.jsp");
%>
```

容易想到，只要诱骗 admin 用户发起一个到 http://10.10.67.25:8080/testxss/doadmin.jsp 的 http 请求，就能成功攻击。因此我们设计跨站语句如下：

```
hello </textarea> <img src="http://tinyurl.com/2xwfed" style="display:none"> </img>
```

```
hello </textarea> <form id="shit" action="http://10.10.67.25:8080/testxss/doadmin.jsp" method="post" target="myframe"/> <input type="hidden" name="v" value="hacked3"/> </form> <iframe style="display:none" name="myframe"> </iframe><script>document.forms[0].submit()</script>
```

```
hello </textarea> <script language="jscript">v = new ActiveXObject("MSXML2.XMLHTTP.3.0"); v.open("GET","http://10.10.67.25:8080/testxss/doadmin.jsp?v=hacked4"); v.send();alert(v.statusText);</script>
```

以普通用户身份修改 user value 为以上任何一个，当 admin 浏览 index.jsp 时，即可悄无声息的修改 admin value

这里演示了 3 种跨站手法，1 是利用 img、iframe 等 tag 直接发起请求，这适用于无法直接出 script 的情况，其中 http://tinyurl.com/2xwfed 是一个 redirect，指向 http://10.10.67.25:8080/testxss/doadmin.jsp?v=hacked2 ；2 是用 script 提交 post 表单；3 是 ajax 技术。

以上攻击能够成功有 2 个原因：1. 应用程序没有对 user value 做足够多的过滤，导致用户有机会构造一个复杂的跨站语句来触发 admin 的非预期行为；2. 应用程序在响应 admin value 修改请求时没有防范措施来识别这是不是出于用户主动。

漏洞 1 很容易修复，只要像防止 SQL Injection 那样对用户输入的所有内容都过滤即可。漏洞 2 才是问题的根源，即便我们修补了漏洞 1，只要诱使 admin 用户访问包含<img src="http://tinyurl.com/2xwfed"> </img>的页面，仍然能达到目的，而这是一件极容易做到的事。

防范措施

这里给出一些防范 XSS 攻击的措施。必须说明的是，对于 XSS 攻击，并不像 SQL Injection 那样可以有一劳永逸的解决方案——只需要 grep 一下所有的 sql 调用。这是一场长期的斗争，而且往往需要我们采取修改业务流程、产品设计等看似削足适履的手段。

先总结一下常见的攻击手法：

1. 依赖跨站漏洞，需要在被攻击网站的页面种入脚本的手法
1.1. Cookie 盗取，通过 javascript 获取被攻击网站种下的 cookie，并发送给攻击者。
1.1.1. 从 cookie 中提取密码等隐私
1.1.2. 利用 cookie 伪造 session，发起重放攻击
1.2. Ajex 信息盗取，通过 javascript 发起 ajex 请求。
1.2.1. 从 ajex 结果中获取隐私。
1.2.2. 模拟用户完成多页表单。
2. 不依赖跨站漏洞的手法

2.1. 单向 HTTP 动作，通过 img.src 等方法发起跨站访问，冒充被攻击者执行特权操作。但是很难拿到服务器的返回值。

2.2. 双向 HTTP 动作，如果服务器产生一段动态的 script，那么可以用 script.src 的方法发起跨站访问并拿到服务器的返回值。

防范手法如下：

1. 防堵跨站漏洞，阻止攻击者利用在被攻击网站上发布跨站攻击语句

不可以信任用户提交的任何内容，首先代码里对用户输入的地方和变量都需要仔细检查长度和对"<",">",";",""”等字符做过滤；其次任何内容写到页面之前都必须加以 encode，避免不小心把 html tag 弄出来。

这一个层面做好，至少可以堵住超过一半的 XSS 攻击。

2. Cookie 防盗

首先避免直接在 cookie 中泄露用户隐私，例如 email、密码等等。

其次通过使 cookie 和系统 ip 绑定来降低 cookie 泄露后的危险。这样攻击者得到的 cookie 没有实际价值，不可能拿来重放。

3. 尽量采用 POST 而非 GET 提交表单

POST 操作不可能绕开 javascript 的使用，这会给攻击者增加难度，减少可利用的跨站漏洞。

4. 严格检查 refer

检查 http refer 是否来自预料中的 url。这可以阻止第 2 类攻击手法发起的 http 请求，也能防止大部分第 1 类攻击手法，除非正好在特权操作的引用页上种了跨站访问。

5. 将单步流程改为多步，在多步流程中引入效验码

多步流程中每一步都产生一个验证码作为 hidden 表单元素嵌在中间页面，下一步操作时这个验证码被提交到服务器，服务器检查这个验证码是否匹配。首先这为第 1 类攻击者大大增加了麻烦。其次攻击者必须在多步流程中拿到上一步产生的效验码才有可能发起下一步请求，这在第 2 类攻击中是几乎无法做到的。

6. 引入用户交互

简单的一个看图识数可以堵住几乎所有的非预期特权操作。

7. 只在允许 anonymous 访问的地方使用动态的 javascript。

8. 对于用户提交信息的中的 img 等 link，检查是否有重定向回本站、不是真的图片等可疑操作。

9. 内部管理网站的问题

很多时候，内部管理网站往往疏于关注安全问题，只是简单的限制访问来源。这种网站往往对 XSS 攻击毫无抵抗力，需要多加注意。

安全问题需要长期的关注，从来不是一锤子买卖。XSS 攻击相对其他攻击手段更加隐蔽和多变，和业务流程、代码实现都有关系，不存在什么一劳永逸的解决方案。此外，面对 XSS，往往要牺牲产品的便利性才能保证完全的安全，如何在安全和便利之间平衡也是一件需要考虑的事情。