Web 方式攻击介绍

网络越来越发展的今天，针对 web 进行的攻击也越来越多，下面我们就简单介绍几种主流的 web 攻击方式

一.跨站脚本攻击(XSS)

跨站脚本攻击（XSS，Cross-site scripting）是最常见和基本的攻击 WEB 网站的方法。攻击者在网页上发布包含攻击性代码的数据。当浏览者看到此网页时，特定的脚本就会以浏览者用户的身份和权限来执行。通过 XSS 可以比较容易地修改用户数据、窃取用户信息，以及造成其它类型的攻击，

例如 CSRF 攻击

常见解决办法:确保输出到 HTML 页面的数据以 HTML 的方式被转义

出错的页面的漏洞也可能造成 XSS 攻击.

比如页面/gift/giftList.htm?page=2 找不到,出错页面直接把该 url 原样输出,如果攻击者在 url 后面加上攻击代码发给受害者,就有可能出现 XSS 攻击

二. 跨站请求伪造攻击（CSRF)

跨站请求伪造（CSRF，Cross-site request forgery）是另一种常见的攻击。攻击者通过各种方法伪造一个请求，模仿用户提交表单的行为，从而达到修改用户的数据，或者执行特定任务的目的。为了假冒用户的身份，CSRF 攻击常常和 XSS 攻击配合起来做，但也可以通过其它手段，例如诱使用户点击一个包含攻击的链

接。

解决的思路有:

1.采用 POST 请求,增加攻击的难度.用户点击一个链接就可以发起 GET 类型的请求。而 POST 请求相对比较难，攻击者往往需要借助 javascript 才能实现

2.对请求进行认证，确保该请求确实是用户本人填写表单并提交的，而不是第三者伪造的.具体可以在会话中增加 token,确保看到信息和提交信息的是同一个人

三.Http Heads 攻击

凡是用浏览器查看任何 WEB 网站，无论你的 WEB 网站采用何种技术和框架，都用到了 HTTP 协议.HTTP 协议在 Response header 和 content 之间，有一个空行，即两组 CRLF（0x0D 0A）字符。这个空行标志着 headers 的结束和 content 的开始。“聪明”的攻击者可以利用这一点。只要攻击者有办法将任意字符“注入”到 headers 中，这种攻击就可以发生

以登陆为例:有这样一个 url:

http://localhost/login?page=http%3A%2F%2Flocalhost%2Findex

当登录成功以后，需要重定向回 page 参数所指定的页面。下面是重定向发生时的 response headers.

HTTP/1.1 302 Moved Temporarily

Date: Tue, 17 Aug 2010 20:00:29 GMT

Server: Apache mod_fcgid/2.3.5 mod_auth_passthrough/2.1 mod_bwlimited/1.4 FrontPage/5.0.2.2635

Location: http://localhost/index

假如把 URL 修改一下，变成这个样子：

http://localhost/login?page=http%3A%2F%2Flocalhost%2Fcheckout%0D%0A%0D%0A%3Cscript%3Ealert%28%27hello%27%29%3C%2Fscript%3E

那么重定向发生时的 reponse 会变成下面的样子：

HTTP/1.1 302 Moved Temporarily

Date: Tue, 17 Aug 2010 20:00:29 GMT

Server: Apache mod_fcgid/2.3.5 mod_auth_passthrough/2.1mod_bwlimited/1.4 FrontPage/5.0.2.2635

Location: http://localhost/checkout<CRLF>

<CRLF>

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

这个页面可能会意外地执行隐藏在 URL 中的 javascript。类似的情况不仅发生在重定向（Location header）上，也有可能发生在其它 headers 中，如 Set-Cookie header。这种攻击如果成功的话，可以做很多事，例如：执行脚本、设置额外的 cookie（<CRLF>Set-Cookie: evil=value）等。

避免这种攻击的方法，就是过滤所有的 response headers，除去 header 中出现的非法字符，尤其是 CRLF。

服务器一般会限制 request headers 的大小。例如 Apache server 默认限制 request header 为 8K。如果超过 8K，Aapche Server 将会返回 400 Bad Request 响应：

对于大多数情况，8K 是足够大的。假设应用程序把用户输入的某内容保存在 cookie 中,就有可能超过 8K.攻击者把超过 8k 的 header 链接发给受害者,就会被服务器拒绝访问.解决办法就是检查 cookie 的大小,限制新 cookie 的总大写，减少因 header 过大而产生的拒绝访问攻击

四.Cookie 攻击

通过 Java Script 非常容易访问到当前网站的 cookie。你可以打开任何网站，然后在浏览器地址栏中输入：javascript:alert(doucment.cookie),立刻就可以看到当前站点的 cookie (如果有的话)。攻击者可以利用这个特性来取得你的关键信息。例如，和 XSS 攻击相配合，攻击者在你的浏览器上执行特定的 Java Script 脚本，取得你的 cookie。假设这个网站仅依赖 cookie 来验证用户身份，那么攻击者就可以假冒你的身份来做一些事情。

现在多数浏览器都支持在 cookie 上打上 HttpOnly 的标记,凡有这个标志的 cookie 就无法通过 Java Script 来取得,如果能在关键 cookie 上打上这个标记，就会大大增强 cookie 的安全性

五.重定向攻击

一种常用的攻击手段是“钓鱼”。钓鱼攻击者，通常会发送给受害者一个合法链接，当链接被点击时，用户被导向一个似是而非的非法网站，从而达到骗取用户信任、窃取用户资料的目的。为防止这种行为,我们必须对所有的重定向操作进行审核,以避免重定向到一个危险的地方.常见解决方案是白名单,将合法的要重定向的 url 加到白名单中,非白名单上的域名重定向时拒之,第二种解决方案是重定向 token,在合法的 url 上加上 token,重定向时进行验证.

六.上传文件攻击

1.文件名攻击,上传的文件采用上传之前的文件名,可能造成:客户端和服务端

字符码不兼容,导致文件名乱码问题;文件名包含脚本,从而造成攻击.

2.文件后缀攻击.上传的文件的后缀可能是 exe 可执行程序,js 脚本等文件,这些程序可能被执行于受害者的客户端,甚至可能执行于服务器上.因此我们必须过滤文件名后缀,排除那些不被许可的文件名后缀.

3.文件内容攻击.IE6 有一个很严重的问题 ，它不信任服务器所发送的 content type，而是自动根据文件内容来识别文件的类型，并根据所识别的类型来显示或执行文件.如果上传一个 gif 文件,在文件末尾放一段 js 攻击脚本,就有可能被执行.这种攻击,它的文件名和 content type 看起来都是合法的 gif 图片,然而其内容却包含脚本,这样的攻击无法用文件名过滤来排除，而是必须扫描其文件内容，才能识别。

七.SQL 注入

最常见的攻击方式，所谓 SQL 注入，就是通过把 SQL 命令插入到 Web 表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令，比如先前的很多影视网站泄露 VIP 会员密码大多就是通过 WEB 表单递交查询字符暴出的，这类表单特别容易受到 SQL 注入式攻击.