# 漏洞原理防御（寒假）

原创

七月24 于 2021-12-29 17:30:28 发布 1004 收藏

文章标签： web安全 前端 安全



本文链接：https://blog.csdn.net/weixin_55773382/article/details/122217744



## web常见漏洞总结与防御

### 1，sql注入

被广泛的应用于网站控制，在设计程序上，忽略对输入字符串中夹带的sql指令的检查，被数据库认为是正常的sql指令而运行，从而数据库受到攻击，导致数据库被窃取，更改，删除，以及导致网站被嵌入恶意代码，被植入后门程序等危害

#### 注入的位置：

1，表单提交（post，get）
2，URL参数提交，主要是get请求
3，cookies参数提交
4，http请求中一些可以修改的值，比如:referer,user_agentdeng等，在掌握安全封神台中遇到过，则首先要查看源代码来进行修改
5，一些边缘的输入点，比如mp3文件的一些信息

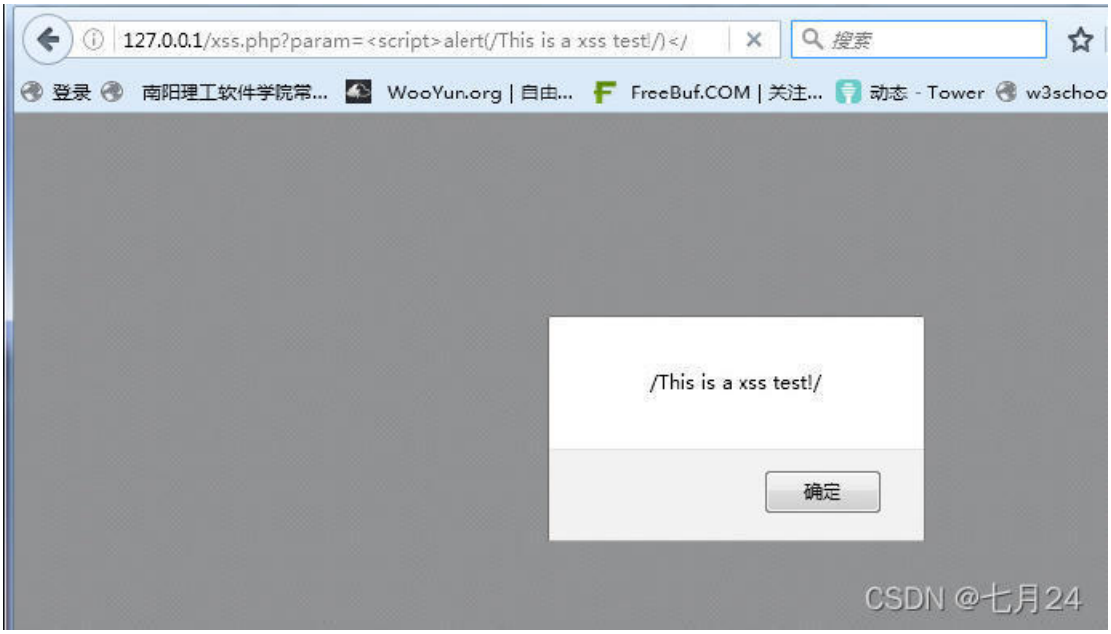
#### 防御

1，使用拼接语句
2，对特殊字符（‘ " <> &…)进行转义，或者编码转换
3，确认每种数据的类型（数字型，字符型。。。。）
4，数据长度严格规定，防止sql注入语句无法正常执行
5，建议使用utf—8编码
6，严格限制用户的数据库的操作，最大限度的减少注入攻击对数据库的危害
7，使用一些专业的sql注入检测工具进行检测，以及及时修补这些sql注入漏洞

### 2，xss原理

#### 1，反射型

用户提交的数据中可以伪造代码来执行，从而实现窃取用户信息的等攻击，需要诱惑用户去点击“一个恶意连接，”才能够进行攻击例子：在你的浏览器中插入一段恶意的javascript脚本，窃取你的隐私信息，冒充的身份进行操作




[alt](　　提交一段html代码，在这个页面就先执行了，有弹窗跳出

反射型XSS，是最常用的，使用最广的一种方式。通过给别人发送有恶意脚本代码参数的URL，当URL地址被打开时，特有的恶意代码参数呗HTML解析、执行。
它的特点：是非持久化，必须用户点击带有特定参数的链接才能引起。
反射型XSS，是最常用的，使用最广的一种方式。通过给别人发送有恶意脚本代码参数的URL，当URL地址被打开时，特有的恶意代码参数呗HTML解析、执行。
它的特点：是非持久化，必须用户点击带有特定参数的链接才能引起。

## 2,存储型

把用户输入的数据存储再服务器中，这种xss居于很强的稳定性
存储型的攻击脚本被存储到了数据库或者文件中，服务端在读取了存储的内容回显了。就是存储型。这种情况下用户直接打开正常的页面就会看到被注入
流程如下：
坏人把恶意的XSS代码提交网站—>网站把XSS代码存储进数据库—>当页面再次被其他正常用户请求时，服务器发送已经被植入XSS代码的数据给客户端—>客户端执行XSS代码

## 3， DOM型

通过修改页面的dom节点形成的xss

## 总结

以为xss就是弹窗，其实是错误的，弹窗只是测试形式上的存在性和使用性
当插入js代码时，可以是

或者<img src=x onerror=appendChild(creatElement(‘script’)).src=‘js_url’>等等，只要能够运行js就可以，通过它，可以获取cookies，控制用户的动做等
比如：我们在网站的留言区输入下边的代码

当管理员进入后台浏览器留言时，就会触发，然后管理员的cookies和后台地址还有管理员浏览器版本等等都可以获取到

## 防御

在输入的时时候没有做到严格过滤，对输入中的script，iframe等字样进行严格检查。而且在输出时，也要进行检查，转义，替换等，防范的方法就是做到严格检查，过滤，在输出的时候，对某些·特殊字符进行转义，替换的等等，在发布应用程序时，测试以知威胁

# 3，CSRF

## 攻击原理

CSRF：跨站请求伪造
可以这么理解，攻击者盗用了你的身份，以你的名义发送恶意请求，对于服务器来说完全是合法的，但是却完成攻击者所期望的一个操作，以我的名义发送邮件，发消息，盗取你的账号，甚至用来购买商品，虚假货币等等，
1，用户访问一个正常网站例如www.baidu.com，用户信息通过验证后，该网站产生cookies信息并返回给浏览器，此时用户登录网站成功，可以正常发送请求到网站
2,用户正在同一个浏览器访问了攻击者伪造的恶意网站
3，恶意网站可以让用户自动提交一个请求到目标网站（WWW.BAIDU.COM），恶意代码就会被执行

## 漏洞检测：

检测CSRF最简单的方法就是抓取一个正常请求的数据包，去掉Referer字段后从新提交，如果提交还有效，基本上就可以确定时CSRF漏洞了
检测工具：CSRFtester,CSRF Request Builder等
使用CSRFTester进行测试时，首先需要抓取我们在浏览器中访问过的所有链接以及所有的表单等信息，然后通过在CSRFTester中修改相应的表单等信息，重新提交，这相当于一次伪造客户端请求。如果修改后的测试请求成功被网站服务器接受，则说明存在CSRF漏洞，当然此款工具也可以被用来进行CSRF攻击。

## 防御

CSRF能够攻击成功的原因在于所有参数都是可以被攻击者猜到的，攻击者只有预测出恶意请求的URL的所有参数，才能成功的构造一个伪造的请求，比如要删除目标网站上的一篇文章，攻击者必须知道删除操作的URL以及参数值这个问题
则我们使用token来解决
http://host/path/delete?username=abc&item=123&token=[random(seed)]
1，正在请求地址添加token并验证 CSRF 攻击之所以能够成功，是因为黑客可以完全伪造用户的请求，该请求中所有的用户验证信息都是存在于 cookie 中，因此黑客可以在不知道这些验证信息的情况下直接利用用户自己的 cookie 来通过安全验证。要抵御 CSRF，关键在于在请求中放入黑客所不能伪造的信息，并且该信息不存在于 cookie 之中。可以在 HTTP 请求中以参数的形式加入一个随机产生的 token，并在服务器端建立一个拦截器来验证这个 token，如果请求中没有 token 或者 token 内容不正确，则认为可能是 CSRF 攻击而拒绝该请求。
在URL后边加上token，这个token必须是随机的，由于token地存在，攻击者无法再构造出一个完整的url来实施CSRF攻击，token需要同时存放在请求和session中，当再次提交请求时，只需要验证请求中的token与用户session中的token是否相同，如果不相同，则是发生了CSRF攻击
token 可以在用户登陆后产生并放于 session 之中，然后在每次请求时把 token 从 session 中拿出，与请求中的 token 进行比对，但这种方法的难点在于如何把 token 以参数的形式加入请求。对于 GET 请求，token 将附在请求地址之后，这样 URL 就变成 http://url?csrftoken=tokenvalue。 而对于 POST 请求来说，要在 form 的最后加上 ，这样就把 token 以参数的形式加入请求了。但是，在一个网站中，可以接受请求的地方非常多，要对于每一个请求都加上 token 是很麻烦的，并且很容易漏掉，通常使用的方法就是在每次页面加载时，使用 javascript 遍历整个 dom 树，对于 dom 中所有的 a 和 form 标签后加入 token。这样可以解决大部分的请求，但是对于在页面加载之后动态生成的 html 代码，这种方法就没有作用，还需要程序员在编码时手动添加 token。
该方法还有一个缺点是难以保证 token 本身的安全。特别是在一些论坛之类支持用户自己发表内容的网站，黑客可以在上面发布自己个人网站的地址。由于系统也会在这个地址后面加上 token，黑客可以在自己的网站上得到这个 token，并马上就可以发动 CSRF 攻击。为了避免这一点，系统可以在添加 token 的时候增加一个判断，如果这个链接是链到自己本站的，就在后面添加 token，如果是通向外网则不加。不过，即使这个 csrftoken 不以参数的形式附加在请求之中，黑客的网站也同样可以通过 Referer 来得到这个 token 值以发动 CSRF 攻击。这也是一些用户喜欢手动关闭浏览器 Referer 功能的原因。
2，验证Referer字段，它记录了该http请求的来源地址，在通常情况下，访问一个安全受限制页面的请求来自用一个网站比如需要访问 http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory，用户必须先登陆 bank.example，然后通过点击页面上的按钮来触发转账事件。这时，该转帐请求的 Referer 值就会是转账按钮所在的页面的 URL，通常是以 bank.example 域名开头的地址。而如果黑客要对银行网站实施 CSRF 攻击，他只能在他自己的网站构造请求，当用户通过黑客的网站发送请求到银行时，该请求的 Referer 是指向黑客自己的网站。因此，要防御 CSRF 攻击，银行网站只需要对于每一个转账请求验证其 Referer 值，如果是以 bank.example 开头的域名，则说明该请求是来自银行网站自己的请求，是合法的。如果 Referer 是其他网站的话，则有可能是黑客的 CSRF 攻击，拒绝该请求。
3，在http头中定义属性并验证
也是使用token并进行验证，不同的是，不把tokrn以参数的形式至于http请求中，而是把它放到http头中自定义的属性里，通过XMHttpRequest这个类，可以一次性给所有该类请求加上csrftoken这个http头属性，并把token值放入其中，这样子就解决了在请求中加入token的不便，同时，通过XMLHttpRequest请求的地址不会被记录到浏览器的地址栏，也不用担心token或肉过Referer泄露到其他网站去
这种方法局限性性大，并非所有的请求都适合用这个类发起，而且通过该类请求得到的页面不能被浏览器所记录下，从而进行前进，后退，刷新，收藏等操作都不能实现，给用户带来不便。同时，要采用这种方法来进行防护，要把所有请求都改为 XMLHttpRequest 请求，这样几乎是要重写整个网站

例子： CSRF攻击实例得意得意得意

受害者 Bob 在银行有一笔存款，通过对银行的网站发送请求 http://bank.example/withdraw?account=bob&amount=1000000&for=bob2 可以使 Bob 把 1000000 的存款转到 bob2 的账号下。通常情况下，该请求发送到网站后，服务器会先验证该请求是否来自一个合法的 session，并且该 session 的用户 Bob 已经成功登陆。

黑客 Mallory 自己在该银行也有账户，他知道上文中的 URL 可以把钱进行转帐操作。Mallory 可以自己发送一个请求给银行：http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory。但是这个请求来自 Mallory 而非 Bob，他不能通过安全认证，因此该请求不会起作用。

这时，Mallory 想到使用 CSRF 的攻击方式，他先自己做一个网站，在网站中放入如下代码： src=”http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory ”，并且通过广告等诱使 Bob 来访问他的网站。当 Bob 访问该网站时，上述 url 就会从 Bob 的浏览器发向银行，而这个请求会附带 Bob 浏览器中的 cookie 一起发向银行服务器。大多数情况下，该请求会失败，因为他要求 Bob 的认证信息。但是，如果 Bob 当时恰巧刚访问他的银行后不久，他的浏览器与银行网站之间的 session 尚未过期，浏览器的 cookie 之中含有 Bob 的认证信息。这时，悲剧发生了，这个 url 请求就会得到响应，钱将从 Bob 的账号转移到 Mallory 的账号，而 Bob 当时毫不知情。等以后 Bob 发现账户钱少了，即使他去银行查询日志，他也只能发现确实有一个来自于他本人的合法请求转移了资金，没有任何被攻击的痕迹。而 Mallory 则可以拿到钱后逍遥法外。

## 框架钓鱼漏洞

框架注入攻击针对Internet Exploer5，internet explorer 6,与Internet explorer7攻击中的一种，这种攻击会导致Internet explorer不检查结果框架的目的地网站，因此允许任意代码像js或者VBscript跨框架存储，这种攻击也发生在代码渗透多框架注入，因为脚本并不确认来自于多框架的输入。 这种
其他形式的框架注入会影响所有的不确认不受信任输入的各厂商浏览器和脚本。
如果应用程序不要求不同的框架互相通信，就可以通过完全删除框架名
称、使用匿名框架防止框架注入。但是，因为应用程序通常都要求框架之间相互通
信，因此这种方法并不可行。因此，通常使用命名框架，但在每个会话中使用不同
的框架，并且使用无法预测的名称。一种可行的方法是在每个基本的框架名称后附
加用户的会话令牌，如main_display 。

## 6，文件上传漏洞

文件上传漏洞通常由于网页代码中的文件上传路径变量过滤不严造成
的，如果文件上传功能实现代码没有严格限制用户上传的文件后缀以及文件类型，
攻击者可通过Web 访问的目录上传任意文件，包括网站后门文件（ webshell ），
进而远程控制网站服务器。
因此，在开发网站及应用程序过程中，需严格限制和校验上传的文件，禁
止上传恶意代码的文件。同时限制相关目录的执行权限，防范webshell 攻击。

## 7，应用程序测试脚本泄露

由于测试脚本对提交的参数数据缺少充分过滤，远程攻击者可以利用洞
以WEB 进程权限在系统上查看任意文件内容。防御此类漏洞通常需严格过滤提交
的数据，有效检测攻击。

## 8，私有IP 地址泄露漏洞

IP 地址是网络用户的重要标示，是攻击者进行攻击前需要了解的。获取的方法较多，攻击者也会因不同的网络情况采取不同的方法，如：在局域网内使用 Ping 指令， Ping 对方在网络中的名称而获得IP；在Internet 上使用IP 版的QQ 直接显示。最有效的办法是截获并分析对方的网络数据包。攻击者可以找到并直接通过软件解析截获后的数据包的IP 包头信息，再根据这些信息了解具体的IP。针对最有效的“数据包分析方法”而言，就可以安装能够自动去掉发送数据包包头IP 信息的一些软件。不过使用这些软件有些缺点， 譬如：耗费资源严重，降低计算机性能；访问一些论坛或者网站时会受影响；不适合网吧用户使用等等。现在的个人用户采用最普及隐藏IP 的方法应该是使用代理，由于使用代理服务器后，“转址服务”会对发送出去的数据包有所修改，致使“数据包分析”的方法失效。一些容易泄漏用户IP 的网络软件(QQ 、MSN 、IE 等)都支持使用代理方式连接Internet ，特别是QQ 使用“ ezProxy ”等代理软件连接后， IP 版的QQ 都无法显示该IP 地址。虽然代理可以有效地隐藏用户IP，但攻击者亦可以绕过代理， 查找到对方的真实IP 地址，用户在何种情况下使用何种方法隐藏IP，也要因情况而论。

## 9、未加密登录请求

由于Web 配置不安全， 登陆请求把诸如用户名和密码等敏感字段未加密进行传输， 攻击者可以窃听网络以劫获这些敏感信息。建议进行例如SSH 等的加密后再传输。

## 10、敏感信息泄露漏洞

SQL 注入、XSS、目录遍历、弱口令等均可导致敏感信息泄露，攻击者可以通过漏洞获得敏感信息。针对不同成因，防御方式不同。