# 遍历目录

## 描述

允许用户输入来控制文件系统操作中使用的路径可以使攻击者能够访问或修改通常最终用户无法访问的受保护系统资源。 在某些情况下，用户提供的输入可以直接传递给文件系统操作，也可以连接到一个或多个固定字符串以构建完全限定的路径。

当应用程序不正确地清理用户提供的文件名中的特殊字符序列时，可能会发生路径遍历（或目录遍历）漏洞。例如，攻击者可以指定一个文件名，如“../..”/etc/passwd”，它解析到目标目录之外的文件，攻击者通常不会被授权查看该文件。

此调用包含路径操作缺陷。 函数的参数是使用不受信任的输入构造的文件名。 如果允许攻击者指定全部或部分文件名，则可能会对服务器上的文件（包括webroot之外的文件）进行未经授权的访问，这些文件通常是最终用户无法访问的。暴露程度取决于输入验证程序的有效性（如果有的话）。

## 推荐方法

假设所有用户提供的输入都是恶意的。 验证所有用户提供的输入，以确保它符合预期的格式，尽可能使用集中数据验证例程。 使用黑名单时，请确保清理例程执行足够数量的迭代以删除所有不允许的字符实例，并确保最终结果不危险。

验证所有不受信任的输入，以确保它符合预期的格式，尽可能使用集中数据验证例程。 使用黑名单时，请确保清理例程执行足够数量的迭代以删除所有不允许的字符实例。

# Clickjacking

## 基本概念

点击劫持，也称为“用户界面纠正攻击”，是指攻击者使用多个透明或不透明图层来诱骗用户在打算点击顶层页面时点击其他页面上的按钮或链接。 因此，攻击者“劫持”针对其页面的点击并将其路由到另一个页面，很可能由另一个应用程序，域或两者所有。

使用类似的技术，键击也可能被劫持。 通过精心设计的样式表，iframe和文本框组合，用户可以相信他们输入的是他们的电子邮件或银行帐户的密码，而是键入由攻击者控制的隐形框架。

## 示例

例如，想象一个攻击者建立一个网站，上面有一个按钮，上面写着“点击这里免费获取iPod”。 但是，在该网页的顶部，攻击者已经使用您的邮件帐户加载了iframe，并在“免费iPod”按钮的顶部直接排列了“删除所有邮件”按钮。 受害者试图点击“免费iPod”按钮，但实际上点击了隐藏的“删除所有消息”按钮。 从本质上讲，攻击者“劫持”了用户的点击，因此名称为“Clickjacking”。

Clickjacking最臭名昭着的例子之一是对Adobe Flash插件设置页面的攻击。 通过将此页面加载到不可见的iframe中，攻击者可以欺骗用户更改Flash的安全设置，允许任何Flash动画使用计算机的麦克风和摄像头。

## 防御点击劫持

主要有两个方法来防止点击劫持：

1、发送适当的内容安全策略（CSP）frame-ancestors指令响应标头，指示浏览器不允许从其他域进行框架。（这将替换旧的X-Frame-Options HTTP标头。）

2、在UI中使用防御性代码以确保当前帧是最顶级的窗口。

# SQL注入

## 问题描述

当数据从不受信任的源进入应用程序并用于动态构造SQL查询时，会发生SQL注入漏洞。这允许攻击者操纵数据库查询以访问，修改或删除任意数据。 根据平台，数据库类型和配置，还可以对数据库执行管理操作，访问文件系统或执行任意系统命令。SQL注入攻击还可用于破坏身份验证和授权方案，这将使攻击者能够获得对应用程序受限部分的特权访问。

## 解决方法

1、使用参数化预准备语句而不是动态构造SQL查询。这将阻止数据库将绑定变量的内容解释为查询的一部分，并且是防止SQL注入的最有效方法。

2、使用正过滤器（白名单）验证用户提供的输入，以确保它符合预期的格式，尽可能使用集中数据验证程序。

3、在应用过滤器或正则表达式或将数据提交到数据库之前，规范化所有用户提供的数据。这意味着所有URL编码（％xx），HTML编码（＆＃xx;）或其他编码方案应该减少到应用程序期望的内部字符表示。这可以防止攻击者使用备用编码绕过过滤器的方案。

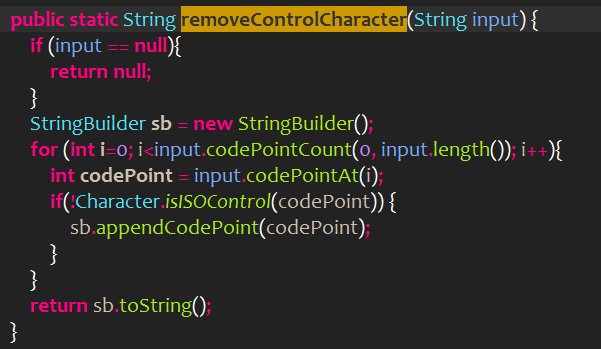
4、使用Hibernate等数据库抽象库时，不要假设API公开的所有方法都会自动防止SQL注入攻击。大多数库包含以不安全的方式将任意查询传递给数据库的方法。

优先使用PreparedSQLStatement，使用它提供的占位符来填充SQL中的参数。

重新定义从外部传入的SQL变量

参考mypage-cn这个项目中的EmailCtrl类中的listMessage方法中的fromRequestMessage(message)这个方法。

removeControlCharacter(String input)主要是这个方法。



# CRLF injection

## 概述

CRLF是”回车 + 换行”（\r\n）的简称，即我们都知道在HTTP协议中，HTTP Header与HTTP Body是用两个CRLF分隔的，浏览器就是根据这两个CRLF来提取HTTP 内容，一旦我们能够控制http头，通过注入一些CRLF这样就可以控制header和body的分割线，这样我们就可以向body或header中注入些东西了，所以CRLF Injection又叫HTTP Response Splitting，简称HRS。

对于HRS最简单的利用方式是注入两个\r\n，之后在写入XSS代码来构造一个xss。

**\n\r（%0d%0a）**

## 命令

表示键盘上的"Enter"键(可以用来模拟[回车键](https://baike.baidu.com/item/%E5%9B%9E%E8%BD%A6%E9%94%AE))

## CRLF注入

就是说黑客能够将[CRLF](https://baike.baidu.com/item/CRLF)命令注入到系统中。它不是系统或[服务器软件](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8%E8%BD%AF%E4%BB%B6)的漏洞，而是网站应用开发时，有些开发者没有意识到此类攻击存在的可能而造成的。

针对这个漏洞黑客能够做什么？就算黑客发现网站存在CRLF注入，他们仍然受到应用结构和这个缺陷的严重程度的限制。对有些站点它将非常严重，而有些站点它只是很小的bug。

HTTP Header CRLF Injection

许多[网络协议](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE)，包括HTTP也使用CRLF来表示每一行的结束。这就意味着用户可以通过CRLF注入自定义HTTP header，导致用户可以不经过[应用层](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E5%B1%82)直接与Server对话。

## 示例

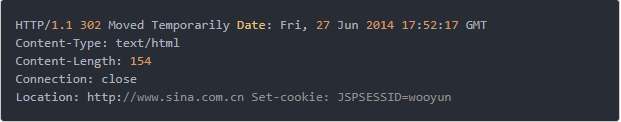
举个例子，一般网站会在HTTP头中用Location:[http://baidu.com](https://link.jianshu.com/?t=http://baidu.com)这种方式来进行302跳转，所以我们能控制的内容就是Location:后面的XXX某个网址。所以一个正常的302跳转包是这样：



但如果我们输入的是

[http://www.sina.com.cn%0aSet-cookie:JSPSESSID%3Dwooyun](https://link.jianshu.com/?t=http://www.sina.com.cn%0aSet-cookie:JSPSESSID%3Dwooyun)

注入了一个换行，此时的返回包就会变成这样



这个时候这样我们就给访问者设置了一个SESSION，造成一个“会话固定漏洞”。 当然HRS并不仅限于会话固定，通过注入两个CRLF就能造成一个无视浏览器Filter的反射型XSS。

比如一个网站接受url参数[http://test.sina.com.cn/?url=xxx](https://link.jianshu.com/?t=http://test.sina.com.cn/?url=xxx)，xxx放在Location后面作为一个跳转。如果我们输入的是

[http://test.sina.com.cn/?url=%0d%0a%0d%0a](https://link.jianshu.com/?t=http://test.sina.com.cn/?url=%0d%0a%0d%0a)<imgsrc=1onerror=alert(/xss/)>

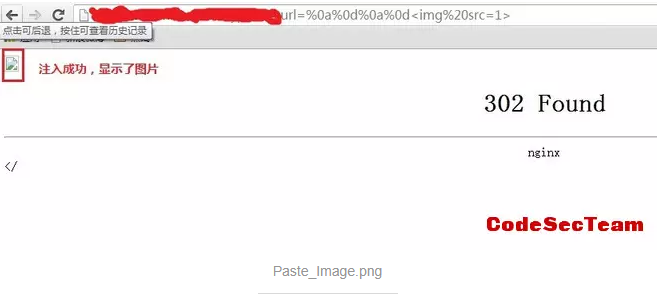
我们的返回包就会变成这样

HTTP/1.1 302 Moved Temporarily Date: Fri, 27 Jun 2014 17:52:17 GMT Content-Type: text/html Content-Length: 154 Connection: close Location: <img src=1 onerror=alert(/xss/)>

之前说了浏览器会根据第一个CRLF把HTTP包分成头和体，然后将体显示出来，于是我们这里这个标签就会显示出来，造成一个XSS。

我们来一个真实案例吧。 新浪某分站含有一个url跳转漏洞，危害并不大，于是我就想到了CRLFInjection，当我测试

[http://xxx.sina.com.cn/?url=%0a%0d%0a%0d%3Cimg%20src=1%3E](https://link.jianshu.com/?t=http://xxx.sina.com.cn/?url=%0a%0d%0a%0d%3Cimg%20src=1%3E)  
的时候，发现图片已经输出在页面中了，说明CRLF注入成功了。



我们试试XSS看看



还想到了一个利用字符编码来绕过XSSFilter的方法，当编码是is-2022-kr时浏览器会忽略%0f，这样我们在onerror后面加个%0f就能绕过filter，前提是注入一个<metacharset=ISO-2022-KR>



当然，在Location:这里注入只有webkit内核浏览器才能够利用，其他浏览器可能会跳转、出错。不过对于chrome的使用量来说，危害已经足够了。

## 解决方法

当然是过滤\r、\n之类的换行符，避免输入的数据污染到其他HTTP头。通过java web filter 过滤特定字符解决此问题，同时也过滤掉SQL注入，XSS 相关的特定敏感字符。

# Credentials Management