开发代码安全规范

防止SQL注入和XSS跨站攻击代码编写规范

修订历史

版本 发布日期 作者 审核者 改版记录

1.0 2015-12-01 正式版

目 录

概述.............................................................2

适用范围.........................................................3

一、第一类漏洞类型-SQL注入(SQL INJECTION)及规范 .................3

1.1名词解释: ....................................................3

1.2经典案例说明: ................................................3

1.3代码实例分析: ................................................6

1.4防止SQL注入攻击的代码安全规范总结: ..........................8

二、第二类漏洞类型-XSS跨站脚本攻击及规范 ........................9

2.1名词解释: ....................................................9

2.2经典案例说明: ................................................9

2.3 防止XSS跨站脚本攻击的代码安全规范总结: .....................10

三、安全操作实践 ................................................11

概述

在技术高速发展的今天，Web应用被广泛使用，伴随而来的是各种安全隐患，主要是编程人员的安全意识较淡薄，缺乏安全编程经验，上线前安全检测不全面。因此，给心怀不轨之人以机会，对公司和个人财产安全造成威胁。本规范希望给编程人员一个较清晰的安全概念，在代码编写时提高警惕。

适用范围

xx集团及其分子公司业务系统的所有开发人员，包括系统外包的第三方开发人员。

一、第一类漏洞类型-SQL注入，SQL Injection，及规范

1.1 名词解释:

SQL注入攻击:通过把SQL命令插入到Web表单递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。

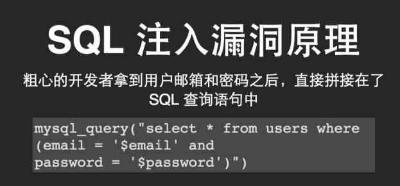
具体来说，它是利用现有应用程序，将(恶意)的SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入(恶意)SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。

1.2 经典案例说明:

例1:用户登录界面及标准输入格式:



Web与数据库连接调用方式:



通过“‘空’or’1’=’1’”这类非法输入，进行恶意SQL注入



以上例子是开发人员直接把用户输入当作可信部分直接和SQL语句拼接造成的SQL漏洞。

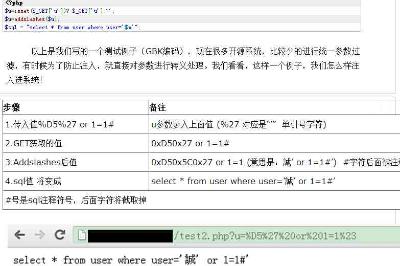
例2:即使开发人员利用PHP内置的过滤函数后，还是有可能出现问题:



调用PHP函数可以改善查询和调用，并限制输入类型，但仍然无法避免注入:



利用GBK转译编码方式依旧可以执行‘空’or‘1=1’类注入语句



因此，使用PHP函数规范输入时，还必须注意字符集选择问题:



1.3 代码实例分析:

Web应用存在着多种多样的SQL注入漏洞，下面以生产实例进行分析。

实例1:参数传递 SQL 语句片断

数据来源:某线上业务被拦截数据

数据日期:2015-10-21

数据内容:HTTP 请求的 URLPATH

/WRIROOT?wri=671&DBName=dev&pagenum=15&page=1&SortValue=%20order%20by%20FeeSta

mp%20%20Desc&XXXXX=WWW

拦截处理截图:



案例分析:

在这个案例中，URL 参数 SortValue 的值为 ' order by FeeStamp Desc'，这是一个典型的 SQL 注入点。对于业务需求来说，传递 order by FeeStamp Desc 可能方便了后端的处理，但是如果被恶意攻击，那么可以传不符合业务预期的参数，对后端数据库造成损害。如:

order by FeeStamp Desc union select username, password, 1, 2, 3 from users

\*注:此处假设存在 users 数据表，并且有username 和 password 的列数据。

:参数传递完整 SQL 语句 实例2

数据来源:某线上业务被拦截数据

数据日期:2015-10-21

数据内容:HTTP 请求的 BODY

拦截处理截图:



案例分析:在这个案例中，HTTP Body 是常见的 application/x-www-form-urlencoded 类型。其中 DBName 参数的值是 dev，sql 参数的值是:

select a.payoutid,a.value,b.basket\_id,b.inserttime,a.suppliername,a.currentrank from tabpayout\_sheet a left join TabPayOut\_Basket b on a.payoutid=b.payoutid where a.payoutid in (605641737)

这里开发人员为了业务的便利性，直接从客户端将完整 SQL 语句通过 HTTP 请求发送给后端。这也是非常典型的 SQL 注入点，对于恶意攻击的黑客来说，可以构造恶意 SQL 语句请求给后端，从而达到恶意获取数据、修改数据、毁坏数据的目的，甚至如果权限设置不合理的话，恶意攻击的黑客可能从该注入点获取服务器管理权限。

1.4 SQL注入危害:

敏感数据被获取(cookie盗取)

网络钓鱼

获取 web 用户的网页内容

Session Riding(CSRF 攻击)

获取用户的键盘击键数据

web 僵尸

1.5防止SQL注入攻击的代码安全规范总结:

1、所有的查询语句都使用数据库提供的参数化查询接口，参数化的语句使用参数而不是将用户输入变量嵌入到SQL语句中。当前，几乎所有的数据库系统都提供了参数化SQL语句执行接口，使用此接口可以非常有效的防止SQL注入攻击。

2、对进入数据库的特殊字符('"\<>&\*;等)进行转义处理，或编码转换。

3、确认每种数据的类型。比如:数字型的数据就必须是数字，数据库中的存储字段必须对应为int型。

4、数据长度应该严格规定，能在一定程度上防止比较长的SQL注入语句无法正确执行。

5、每个数据层的编码统一，建议全部使用UTF-8编码，上下层编码不一致有可能导致一些过滤模型被绕过。

6、严格限制用户的数据库的操作权限，给此用户提供仅仅能够满足其工作的权限，从而最大限度的减少注入攻击对数据库的危害。

7、SQL注入种类，方式变换繁多，并不能做全数说明，所提内容均为基础说明，以提醒警示为主。

二、第二类漏洞类型-XSS跨站脚本攻击及规范

2.1 名词解释:

XSS跨站脚本攻击:为不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆，故将跨站脚本攻击缩写为XSS。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，从而达到恶意攻击用户的特殊目的。

2.2 经典案例说明:

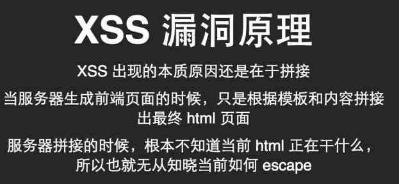
Web应用与数据库正常的调用链接方式如下:



如果插入一条恶意JS脚本



当插入的恶意JS脚本被调用，就形成了XSS攻击。



2.3 XSS危害

绕过防火墙进行攻击

绕过web应用程序的验证过程

非法越权操作数据库内容

随意篡改网页内容

添加系统帐户或数据库帐户

上传和下载非法文件

本地溢出并获取系统最高权限

安装木马后门/僵尸网络

2.4 防止XSS跨站脚本攻击的代码安全规范总结:

1、要假定所有输入都是可疑的，必须对所有输入中的script、iframe等字样进行严格的检查。这里的输入不仅仅是用户可以直接交互的输入接口，也包括HTTP请求中的Cookie中的变量，HTTP请求头部中的变量等。

2、不仅要验证数据的类型，还要验证其格式、长度、范围和内容。

3、不要仅仅在客户端做数据的验证与过滤，关键的过滤步骤在服务端进行。

4、对输出的数据也要检查，数据库里的值有可能会在一个大网站的多处都有输出，即使在输入做了编码等操作，在各处的输出点时也要进行安全检查。

5、在发布应用程序之前测试所有已知的威胁。

6、XSS跨站脚本种类，方式变换繁多，并不能做全数说明，所提内容均为基础说明，以提醒警示为主。

三、安全操作实践

代码漏洞无处不在，本规范无法穷尽开发安全规范操作的各个方面，旨在督促编程人员高度重视。开发实践中，网络和应用安全设备(例如:WAF、IPS等)会将存在严重代码漏洞的语句或脚本进行拦截，开发人员从基础架构的设备管理员得到安全设备的拦截处理日志，据此修改为合规的代码，正常业务才会得以执行。安全建设需要大家相互配合共同努力，不断完善代码，修复漏洞。