Atitit 动态sql的安全性解决方案

[1. 因为安全问题而废弃使用sql查询方式，调用参数化api查询，简直就是因噎废食。。 2](#_Toc26628)

[2. Sql这类dsl的优点 2](#_Toc14664)

[2.1. Sql这类dsl对于项目开发速度提升实在是太好了 2](#_Toc26384)

[2.2. 直接把rest api作为一个sql驱动，对于开发大部分项目来说很合适。。 2](#_Toc17327)

[2.3. 逻辑dsl化sql化，实在不行js化也是可以的，大大提升开发速度啊 3](#_Toc23517)

[2.4. Sql是dsl，比起api高一层次，应该优先使用在查询领域。。最起码在查询的时候作为主力。当然对安全做处理即可。。 3](#_Toc10594)

[2.5. 普通系统80%的开发快速性，扩展性，与灵活性要求大于安全性，就算牺牲一点安全也值得 3](#_Toc24080)

[2.6. SQL注入产生的原因，和栈溢出、XSS等很多其他的攻击方法类似，就是未经检查或者未经充分检查的](#_Toc2561)**[用户输入数据](#_Toc2561)**[，](#_Toc2561)**[意外变成了代码被执行](#_Toc2561)** [3](#_Toc2561)

[2.7. 所以从根本上防止上述类型攻击的手段，还是避免数据变成代码被执行，时刻分清代码和数据的界限。 3](#_Toc818)

[3. Sql注入的情况分析与解决方案 3](#_Toc20425)

[3.1. 单引号问题，直接转义解决即可。 3](#_Toc29686)

[3.2. 禁止注释字符 双杠号的在sql里面检查即可 3](#_Toc4679)

[3.3. 多条语句问题 比如xxx;drop tab1 ,驱动参数里面儿禁止多语句sql执行即可。 4](#_Toc23736)

[3.4. 你的db service api 里面儿,也要禁止多语句sql执行即可 4](#_Toc31588)

[3.5. 此外，执行select 语句，做白名单检查，只允许select开头的sql语句执行 4](#_Toc6868)

[3.6. 查询语句与操作语句的api相分离 4](#_Toc10712)

[3.7. 基于黑白名单的过滤关键字检查，对drop等ddl，以及操作类开头的update ,del 等sql关键字禁止使用在查询语句sql service rest接口中 4](#_Toc21954)

[3.8. 同样采用addslashes转义，也还是存在SQL攻击漏洞。的解决 4](#_Toc9299)

[3.9. 禁止使用char等转义函数在传入的sql中 ，解决但是黑客可以将字符串转为16进制编码数据或使用char函数进行转化，同样能达到相同的目的 5](#_Toc27171)

[3.10. 重要表白名单制度：检查sql是否包含表名，就可以启动TooManyRowsException以及其他查询约束 5](#_Toc19166)

[3.11. 对对sql做词法分析，语法分析，语义分析，对重要表的查询api重点关注，增加TooManyRowsException以及其他查询约束 5](#_Toc32285)

[3.12. 对所有查询sql做TooManyRowsException约束，一般50条就可以了 5](#_Toc24635)

[4. 安全框架 5](#_Toc4543)

[4.1. 权限分级 5](#_Toc21326)

[4.2. 读写分离的sql rest api 5](#_Toc13113)

[4.3. 动态明确的权限授权请求，类似uac 5](#_Toc28893)

[4.4. DEP（Data Excution Protection，数据执行保护 6](#_Toc32307)

[4.5.](#_Toc10270) **[引入了完整性级别（Integrity Level），](#_Toc10270)**[完整性级别分为四级，System（系统），High（高），Medium（中），Low（低）。 6](#_Toc10270)

[4.6.](#_Toc1674) **[ASLR（Address Space Layout Randomization，地址分布随机化）的引入](#_Toc1674)**[。 6](#_Toc1674)

# 因为安全问题而废弃使用sql查询方式，调用参数化api查询，简直就是因噎废食。。

http://www.xxx.com/xxServlet?sql=select \* from table1 where username=’aaa’

这样的格式多么简单快捷啊，一定要大力使用

编程的一大原则就是面向接口，我们界面ui直接面向sql解析接口，传递dsl语句sql进行查询，可以更加的扩展性，与灵活性提升。。

# Sql这类dsl的优点

## Sql这类dsl对于项目开发速度提升实在是太好了

## 直接把rest api作为一个sql驱动，对于开发大部分项目来说很合适。。

特别是内部系统erp crm一类的，可以不用关注安全问题，直接连update 语句都可以传输，用户都是普通小白，就算出了问题也跑不了。。

对于互联网项目，安全才比较重要了。

## 逻辑dsl化sql化，实在不行js化也是可以的，大大提升开发速度啊

## Sql是dsl，比起api高一层次，应该优先使用在查询领域。。最起码在查询的时候作为主力。当然对安全做处理即可。。

至于更新操作，因为毕竟比较少，一般也就20%左右吧，就用后台api模式吧。。。

## 普通系统80%的开发快速性，扩展性，与灵活性要求大于安全性，就算牺牲一点安全也值得

## SQL注入产生的原因，和栈溢出、XSS等很多其他的攻击方法类似，就是未经检查或者未经充分检查的**用户输入数据**，**意外变成了代码被执行**

## 所以从根本上防止上述类型攻击的手段，还是避免数据变成代码被执行，时刻分清代码和数据的界限。

# Sql注入的情况分析与解决方案

## 单引号问题，直接转义解决即可。

## 禁止注释字符 双杠号的在sql里面检查即可

## 多条语句问题 比如xxx;drop tab1 ,驱动参数里面儿禁止多语句sql执行即可。

## 你的db service api 里面儿,也要禁止多语句sql执行即可

## 此外，执行select 语句，做白名单检查，只允许select开头的sql语句执行

## 查询语句与操作语句的api相分离

这样就算执行update语句也会报异常错误。

## 基于黑白名单的过滤关键字检查，对drop等ddl，以及操作类开头的update ,del 等sql关键字禁止使用在查询语句sql service rest接口中

## 同样采用addslashes转义，也还是存在SQL攻击漏洞。的解决

$uid = addslashes(uid);$

sql = "SELECT uid,username FROM user WHERE uid={$uid}";

由于第一句SQL的变量被单引号包含起来，SQL注入的时候，黑客面临的首要问题是必须要先闭合前面的单引号，这样才能使后面的语句作为SQL执行，并且还要注释掉原SQL语句中的后面的单引号，这样才可以成功注入，由于代码里使用了addslashes函数，黑客的攻击会无从下手，但第二句没有用引号包含变量，那黑客也不用考虑去闭合、注释，所以即便同样采用addslashes转义，也还是存在SQL攻击漏洞。  
黑客可以使用单引号,addslashes不个哪转换成个双引号.. 参数’ or xxx走ok兰..

解决方案：框架上面做语句返回条数检测，对此表查询，只允许返回一条数据，超出则报TooManyRowsException

## 禁止使用char等转义函数在传入的sql中 ，解决但是黑客可以将字符串转为16进制编码数据或使用char函数进行转化，同样能达到相同的目的

解决方案：： 检测禁止使用char等转义函数在传入的sql中。。如果发现报异常即可。

## 重要表白名单制度：检查sql是否包含表名，就可以启动TooManyRowsException以及其他查询约束

## 对对sql做词法分析，语法分析，语义分析，对重要表的查询api重点关注，增加TooManyRowsException以及其他查询约束

## 对所有查询sql做TooManyRowsException约束，一般50条就可以了

# 安全框架

## 权限分级

## 读写分离的sql rest api

## 动态明确的权限授权请求，类似uac

让用户的sql运行在普通模式下。。那么如果要运行高权限怎么办？？动态的明确提升权限，req请求 limit=100 ,就可以提升最大限制返回的权限。。防止病毒木马的侵害。。类似于windows的uac

Uac

**受限账户下可以自动检测部分提升请求**（该功能被叫做LUA： Least-privilege User Account，最低权限用户账户，也就是UAC的前身）

## ****DEP（Data Excution Protection，数据执行保护****

它的原理很简单。就是不让内存中的代码直接执行（缓冲区攻击），默认仅为系统程序和显式声明自己支持DEP的程序启用**很快黑客就弄出了绕过DEP的方法**（内存位置预测+注入）。不过对于一般黑客还是提升明显的。

权限与分组

## **引入了完整性级别（Integrity Level），**完整性级别分为四级，System（系统），High（高），Medium（中），Low（低）。

System IL对应什么权限自然不用说了，High和Medium对应什么权限呢？

答案是High对应Administrators组拥有的权限，Medium对应Users组所对应的权限（PS：Users/Medium的权限已在NT6中进行了“不危及系统安全性的”提升，目的是让用户在降权的管理员账户下做更多的事。）。

## **ASLR（Address Space Layout Randomization，地址分布随机化）的引入**。

Vista开始，系统开始引入ASLR，这玩意就是为了**防止绕过DEP的。**

之前的Windows版本上，DLL加载的位置都是固定的，而这就容易给黑客带来可乘之机，而在Vista中，**系统重要的DLL文件和显式声明自己支持ASLR的DLL文件加载到的内存位置都是随机的**。

我们的网页程序api也是固定的话，那就容易有这个问题，也可以搞成动态api即可哈。。

参考

如何从根本上防止 SQL 注入？ - 知乎.html

Windows 7 中的用户帐户控制（UAC）真的有必要吗？ - 知乎.html wids secury histry

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ）

汉字名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

Atiend