

SpringMVC 拦截器防止 SQL 注入

王韬懿

08111302

1120132046

目 录

一 引言.....	3
二 SQL 注入	3
2.1 SQL 注入简介	3
2.2 SQL 注入案例	4
2.2.1 一个简单的 PHP 登录验证 SQL 注入	4
2.2.2 一个 ASP 新闻动态页面的 ID 查询	5
三 SPRINGMVC 拦截器防止 SQL 注入.....	6
3.1 自定义拦截器类实现 HANDLERINTERCEPTOR 接口	6
3.2 配置拦截器	8
四 SQL 注入总结	8
五 课程总结.....	8

一 引言

随着互联网的发展，人们在享受互联网带来的便捷的服务的时候，也面临着个人的隐私泄漏的问题。小到一个拥有用户系统的小型论坛，大到各个大型的银行机构，互联网安全问题都显得格外重要。而这些网站的背后，则是支撑整个服务的核心数据库。可以说数据库就是这些服务的命脉，没有数据库，也就无从谈起这些服务了。

对于数据库系统的安全特性，主要包括数据独立性、数据安全性、数据完整性、并发控制、故障恢复等方面。而这些里面显得比较重要的一个方面是数据的安全性。由于开发人员的设计不周到，以及数据库的某些缺陷，很容易让黑客发现系统的漏洞，从而造成巨大的损失。

接下来本文将会介绍非常常见的一种攻击数据库的方法：SQL 注入，以及使用在项目使用 Java 开发的情况下如何使用 SpringMVC 的拦截器实现防止 SQL 注入的功能。

二 SQL 注入

2.1 SQL 注入简介

通俗的讲，SQL 注入就是恶意黑客或者竞争对手利用现有的 B/S 或者 C/S 架构的系统，将恶意的 SQL 语句通过表单等传递给后台 SQL 数据库引擎执行。比如，一个黑客可以利用网站的漏洞，使用 SQL 注入的方式取得一个公司网站后台的所有数据。试想一下，如果开发人员不对用户传递过来的输入进行过滤处理，那么遇到恶意用户的时候，并且系统被发现有漏洞的时候，后果将是令人难以想象的。最糟糕的是攻击者拿到了数据库服务器最高级的权限，可以对整个数据库服务器的数据做任何操作。

通常情况下，SQL 注入攻击一般分为以下几个步骤：

- 判断 WEB 环境是否可以注入 SQL。对于一个传统的 WEB 网页来说，如果是一个简单的静态页面，比如地址为 `http://www.news.com/100.html` 的网页，这是一个简单的静态页面，一般不涉及对数据库的查询。而在这个网页的网址变成了：`http://www.news.com/news.asp?id=100` 那么，这就是一个根据主键 id 来查询数据的动态网页了。攻击者往往能够在 `id=100` 的后面加上一些自己的 SQL，用来“欺骗”过应用程序。

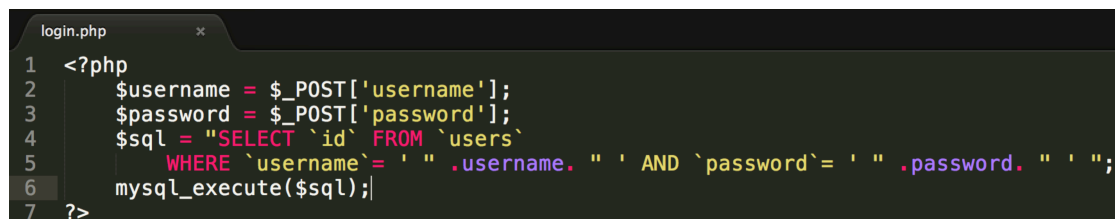
- 寻找 SQL 注入点。当攻击者确定某个页面可以使用 SQL 注入之后，他一般会寻找可以 SQL 注入的点。而这些点往往就是网页或者应用程序中的用于向服务器发送用户数据的表单了。一般的流程是，客户端通过这些表单发送一些用户信息的字段，比如用户名和密码，接着服务器就会根据这些字段去查询数据库。而如果用户输入了一些非法的字符，比如'这个符号，那么在 SQL 解析的时候，解析的结果可能并不是应用开发人员想象中那样。
- 寻找系统的后台。这一点就建立在破坏者对整个系统的了解程度上面了，如果攻击者对整个系统了如指掌，那么实现攻击也就是一件很简单的事情了。
- 入侵和破坏。当攻击者攻破系统之后，整个系统从某种意义上来讲已经失去意义了。

2.2 SQL 注入案例

2.2.1 一个简单的 PHP 登录验证 SQL 注入

比如一个公司有一个用来管理客户的客户管理系统，在进入后台进行管理的时候需要输入用户名和密码。

假设在客户端传给服务器的字段分别为用户名 `username` 和密码 `password`，那么如果用来处理登录的服务器端代码对用户的输入做以下处理：



```
login.php
1 <?php
2 $username = $_POST['username'];
3 $password = $_POST['password'];
4 $sql = "SELECT `id` FROM `users`
5 WHERE `username` = ' ' . username . ' ' AND `password` = ' ' . password . ' ' ";
6 mysql_execute($sql);
7 ?>
```

上面的 PHP 代码就是首先获取客户端 POST 过来的填写在表单里面的用户名和密码，接着要 MySQL 去执行下面这条 SQL 语句：

```
SELECT `id` FROM `users` WHERE `username` = username AND `password` = password;
```

可以看到上面的代码没有对用户的输入做任何过滤，这是非常容易遭到黑客攻击的。

比如，一个用户在输入用户名的输入框里面输入了

```
user';SHOW TABLES;--
```

那么就会解析为下面的 SQL：

```
SELECT `id` FROM `users` WHERE `username` = 'user';SHOW TABLES;--  
AND `password` = password;
```

可以看到被解析的 SQL 被拆分为了 2 条有用的 SQL:

(1)SELECT `id` FROM `users` WHERE `username` = 'user';

(2)SHOW TABLES;

而后面验证密码的部分就被注释掉了。这样攻击者就轻松的获得了这个数据库里面的所有表的名字。同样的道理，如果攻击者在输入框里面输入：

```
';DROP TABLE [table_name];--
```

那么，这一张表就被删除了，可见后果是非常严重的。

而用户如果在用户名输入框里面输入了：user'or 1=1--

那么会被解析成：

```
SELECT `id` FROM `users` WHERE `username` = 'user'  
or 1=1-- AND `password` = password;
```

这里可以看到 1=1 永远为真，这样不用输入用户名就直接可以登入系统了

。

2.2.2 一个 ASP 新闻动态页面的 id 查询

比如有一个新闻网站的动态页面是这种格式：

http://www.news.com/news.asp?id=100 那么当用户在浏览器的地址框里面输入 http://www.news.com/news.asp?id=100;and user>0

那么如果这个网站的数据库用的是 SQL Server 的话，那么 SQL Server 在解析的时候，由于 user 是 SQL Server 的一个内置变量，它的值就是当前连接数据库的用户，那么 SQL 在执行到这里的时候会拿一个类型为 nvarchar 的和一个 int 类型的比较。比较过程中就必然涉及类型的转化，然而不幸的是，转化过程并不是一帆风顺，出错的时候 SQL Server 将会给出类似将 nvarchar 值”aaa”转为为 int 的列时发生语法错误。这个时候攻击者就轻松的获得了数据库的用户名。

三 SpringMVC 拦截器防止 SQL 注入

为了有效的减少以及防止 SQL 注入的攻击，开发人员在开发的时候一定不要期待用户的输入都是合理的，当用户输入完毕之后，应该严谨的对用户提交的数据进行格式上的检查以及过滤，尽可能的减少被注入 SQL 的风险。

一般来说，不同的服务器端语言都有不同的解决方案。比如拿中小型企业采用得最多的 PHP 语言来说，PHP 的官方就为开发者提供了这样的一些函数，比如 `mysql_real_escape_string()` 等。

接下来会详细介绍如何通过 SpringMVC 的拦截器实现防止 SQL 注入的功能。这里以我的大数据人才简历库的拦截器为例。

3.1 自定义拦截器类实现 HandlerInterceptor 接口

```
package com.data.job.util.interceptor;

import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.Enumeraion;

/**
 * 防止 SQL 注入的拦截器
 *
 * @author tyee.noprom@qq.com
 * @time 2/13/16 8:22 PM.
 */
public class SqlInjectInterceptor implements HandlerInterceptor {

    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object o) throws Exception {
        Enumeraion<String> names = request.getParameterNames();
```

```
        while (names.hasMoreElements()) {
            String name = names.nextElement();
            String[] values = request.getParameterValues(name);
            for (String value : values) {
                value = clearXss(value);
            }
        }
        return true;
    }

    public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object o, ModelAndView modelAndView) throws Exception {

    }

    public void afterCompletion(HttpServletRequest request, HttpServletResponse
response, Object o, Exception e) throws Exception {

    }

    /**
     * 处理字符转义
     *
     * @param value
     * @return
     */
    private String clearXss(String value) {
        if (value == null || "".equals(value)) {
            return value;
        }
        value = value.replaceAll("<", "&lt;").replaceAll(">", "&gt;");
        value = value.replaceAll("\\(", "&#40;").replaceAll("\\)", "&#41;");
        value = value.replaceAll("'", "&#39;");
        value = value.replaceAll("eval\\((.*)\\)", "");
    }
}
```

```

        value = value.replaceAll("[\\\\"'\\s]*javascript:(.*)"\\\\"'",
            "\\\"");
        value = value.replace("script", "");
        return value;
    }
}

```

3.2 配置拦截器

```

<!-- 拦截器配置-->
<mvc:interceptors>
    <!-- SQL 注入拦截-->
    <mvc:interceptor>
        <mvc:mapping path="**"/>
        <bean
class="com.data.job.util.interceptor.SqlInjectInterceptor"></bean>
    </mvc:interceptor>
</mvc:interceptors>

```

四 SQL 注入总结

安全问题从古至今都有着非常高的关注度，如今互联网高速发达，各种网络应用层出不穷。不管是现在炙手可热的云计算和大数据，还是传统的 B/S 或者 C/S 架构的网络应用，数据的安全性是至关重要的。从应用层面来讲，网络应用的开发者可以通过完善软件的架构，尽量减少因为 BUG 而导致的数据泄漏的问题。开发者可以不断的审视与完善自己的系统，站在攻击者的角度上去开发某些关键的安全性要求比较高的环节，如银行支付部分等，这样能够在一定程度上减少 SQL 注入等应用层面攻击数据库的风险。

五 课程总结

两个月的 J2EE 课程时间说长也不长，说短也不短。由于自己以前做过 PHP 的企业项目，也用 Java 写过安卓，因此对 J2EE 上手比较容易。从最初的 JSP+Servlet+JavaBean 架构的电话簿程序到如今使用 SpringMVC+Mybatis+Spring 架构的大数据人才简历库，我收获颇多。

即使自己对于 Web 开发还是比较熟悉的，但是比起 PHP 来说，用 Java 写网站还是有些不熟悉。从考试结束连续看了将近一周的 SpringMVC 的书籍，自己对于 SpringMVC 的项目才渐渐熟悉了起来。虽然对于 PHP 开发网站比较熟悉，但是对于开发大型的企业级网站，我还是比较看好 Java，因此以后会多用 Java 开发网站，努力构建强大的 Web 系统。

最后感谢老师两个多月的辛勤付出。