基于 jQuery 的 SQL 注入攻击防范实现

刘 毅,刘益和
(内江师范学院 计算机科学学院,四川 内江 641100)

摘 要:现今实现防止 SQL 注入攻击主要是在服务器端实现,实现的方法主要有基于正则表达式的输入验证、敏感字符的改写、部分数据加密。上述方法的缺点主要体现在以下的几个方面,其一,服务端处理数据使系统资源被过分占用,容易造成服务器端拒绝服务;其二,编写动态网页的脚本语言多样化,每种脚本对应一种防止 SQL 注入方法,使程序不具有可读性、标准性。文中着力解决基于服务器端处理 SQL 注入攻击的缺陷,故探寻了一种在客服端实现的基于 jQuery 防止 SQL 注入攻击的方法,其目的是使攻击在客服端就被拦截,同时利用 jQuery 开放性的特点使 web 程序更具有可移植性。

关键词:jQuery;SQL 注入;javascript 框架;jQuery 插件;入侵检测系统
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2011)02-0177-04

JQuery-Based Implementation of SQL Injection Attacks Prevention

LIU Yi, LIU Yi-he
(Computer Science College, Neijiang Normal University, Neijiang 641100, China)

Abstract: The main method of realizing to prevent SQL injection attacks is based on regular expression validation, sensitive character's change, partly data's encryption on the server. As is stated above, disadvantage of those method are mainly embodied in the following aspects, firstly on the server deal with information easily to create system resource are over-occupied, easy to create server-side denial of service. Secondly mutiplex ways to compile web program, each script have each an opinion to prevent attack. Accordingly program is absent of readability, standardization. So discuss a method that is based on jQuery, and the purpose of preventing the SQL injection on the client. As the same time, while taking advantage of the characteristics of jQuery open-ended program to make web more portable.

Key words: jQuery; SQL injection; javascript frame; jQuery plug-in; IDS

0 引 言

SQL 注入攻击(SQL Injection)是近几年前流行的 web 攻击方法之一,为此有经验的程序员都非常注意这方面的问题,八仙过海可谓各有高招,文献[1]描述的是基于正则表示式的方法,文献[2]论述了部分加密防御 SQL 注入攻击,文献[3]论述基于 URL 重写的防止策略,这些方法都是在服务器端实现的敏感字符的编码或者过滤。故文中全新地提出了在客服端解决此问题的方法,所编写的基于 jQuery 防止 SQL 注入攻击的插件之目的就是实现防止 SQL 注入攻击程序的标准化,程序员无论用什么脚本语言(asp、asp. net、jsp、php)编写的动态 web 程序用此插件都具有开放性,只要导入本插件的全局函数到 web 程序中就能实现无论是基于 get 或者是基于 post 的数据提交方式的任意一种的数据验证,把攻击阻挡在"国门之外"。

1 SQL 注入攻击原理分析

SQL 注入攻击利用 SQL(结构化查询语言)结合防火墙对 80 端口的开放性(web 服务器一般开放此端口)组合成一条具有攻击性的 SQL 语句,包括数据查询语句(导致敏感信息的泄露)、数据操作语句(非法对信息进行修改,删除、更新)、控制语句(对 DBMS 的权限,DATABASE 的权限更改)、存储过程(一种新的 SQL 注入方式)。通过对注入点的采集,攻击方可能发动三种方式的攻击形式:第一,获取敏感信息;第二,通过注入点挂马,使 web 服务器成为肉鸡,攻击者通过指令发动攻击获取网站客户的信息,这是现在黑客惯用手法;第三,上传病毒到服务器从而实现 DOS 攻击。

2 SQL 注入攻击的一般方法

SQL 注入利用网站应用程序的漏洞,通过 GET 或者 POST 方式提交数据,如果程序设计存在漏洞(对客服端的输入数据弱验证)就会造成 SQL 注入攻击可能。

收稿日期:2010-06-12;**修回日期**:2010-09-02
基金项目:四川省应用基础研究计划课题(04JY029-096);四川省自然科学重点项目(09ZA055)
作者简介:刘 毅(1988-),男,四川南充人,研究方向为计算机应用技术;刘益和,教授,博士,研究方向为信息安全。

下面阐述 SQL 注入攻击步骤的主要内容[4~12]。

2.1 判断 web 服务器提供 HTTP 服务的 OS

地址栏输入 http://www.xxx.com/showNews.asp?id=1010（xxx 做假设处理）能正常浏览，再把 URL 的字母大小写打散，如 http://www.xxx.com/showNews.asp?id=1010。如果也能正常浏览，就说明三个问题。

（1）本网站是用 asp 脚本语言编写。

（2）该数据库表中很可能有一个字段是 id。

（3）提供 web 服务的操作系统极可能是微软操作系统配 IIS。因为现今流行的 web 服务器是 Linux+Apach（严格区分大小写）和 window server 2003+IIS（不区分大小写）。

明确提供 web 服务的环境很重要，因为知己知彼，百战不殆，不同的环境其日志文件的位置，日记文件记录的信息种类，应用程序的漏洞种类都不同，这对于一个黑客来说很重要，真正的高手是攻击无痕。

2.2 SQL 注入攻击注入点的查找

常见的注入点查找方式[4]：第一，使用工具软件来查找注入点，例如，啊 D 注入工具，桂林老兵，NBSI 可以遍历 web 服务器所有的可以注入攻击漏洞的网页，在此基础上在解密数据表与用户字段和数据。第二，手工测试，例如 http://www.xxx.com/showNews.asp?id=1010;and 1=1 与 http://www.xxx.com/showNews.asp?id=1010;and 1=2 如果前者能正确显示而后者不能正确显示说明存在注入点。

文中阐述一种智能化的注入点查询方式，在明确了 web 采用什么脚本语言之后，编写脚本语言（linux 下面的 shell 脚本，windows 下面构建 bat 文件）在后台循环执行命令（linux 下面是加 & 符号，该命令就在后台执行），命令的构建要使用正则表达式+浏览网页命令+后台执行符+重定向结果信息组成，最后只需要查询重定向文件就能判定那张页面存在注入可能。一般的注入工具也就是利用第三种方式的思想加之利用多线程的方式来加快判断的速度提升攻击的便利性的。

2.3 数据库类型的判定

判断服务器的数据库类型，常用的数据库是 SQL server 2000/2003、Access，mysql，Oracle 中的一种。判断的方法就是利用 DBMS 具有的系统表来判断，假设下面的页面存在注入点，http://www.xxx.com/showNews.asp?id=1010;and select 0<>count(*) from sysobjects，如果页面也能正确显示那可能是 SqlServer，同时如是 http://www.xxx.com/showNews.asp?id=1010;and select 0<>count(*) from msysobjects 能正常显示说明是 access 数据库。

2.4 数据库表名的判断

数据库的表名的设定完全是取决于程序设计者的个人爱好，故完成表名的猜解只能通过爆破法，例如 http://www.xxx.com/showNews.asp?id=1010;and Esists（select * from 猜的表名）或者 http://www.xxx.com/showNews.asp?id=1010;0<> select count(*) from 猜的表名，一般有经验的程序员都会使用不规则的命名法加上复杂的字符，但是在国内很多的系统都是使用见名知意的命名法，给黑客们留下了攻击的可能。有经验的攻击者也不会一一的去试每一个表名，都是写一个脚本让其在后台执行，只要猜解正确就退出程序。

2.5 表字段名与值的猜解

完成表名的猜解接下来的任务就是来确定用户表中的字段名，而要加快猜解首先就是确定字段名，方法很简单。输入 http://www.xxx.com/showNews.asp?id=1010;and exists(select * from 表名)，直到显示正确的结果为止。接下来就是判断字段的长度，常用的方法是二分法，http://www.xxx.com/showNews.asp?id=1010; and31>(select top1 len(字段) from 表名)。最后一步就是逐字符的猜解，简便的方式还是使用脚本字符遍历法，例如 http://www.xxx.com/showNews.asp?id=1010;(select top 1 asc(mid(字段名,1,1)) from 表名)>数字，加入遍历的脚本中。

上述的五个步骤具有方法简单，破坏性大的特点。如果要使攻击具有智能性，可以利用 vbscript 编写基于 XMLHTTP 协议的脚本，其实质是构建 XMLHTTP 对象来操控网页返回信息，这里不再赘述。

3 基于 jQuery 插件实现防止 SQL Injection

jQuery 是 javascript 的框架[5]，融入了 DOM，BOM，CSS，AJAX 使其功能更加强大（本插件就融入了 DOM，AJAX，CSS），同时它既有开放性，对于任何的脚本语言都是支持的，所以任何网站只要加载这个框架和本插件就可以对任何的网页实现 SQL 注入的监控，考虑到客服端验证的弱安全性没有人研究使用它来实现 SQL 注入的验证，在这里笔者用下面的 4 种方式来实现客服端的安全验证。

（1）隐藏客服端的 JS（插件）代码。

（2）加密插件的代码，并使网页的每次回传调用都重新加载页面。

（3）服务器端对存放插件的文件夹，windows 加 NTFS 权限，linux 系统用 chmod 和 chown 命令设置文件夹权限，其目的就是不让插件下载到客服端让其来服务器端执行。

（4）插件中存在只要客服端攻击就立即遮罩客服端浏览器页面的代码，不让攻击者有二次攻击的可能。

jQuery 插件分为类级别的插件开发,jQuery 对象级别的插件开发[3]。本插件采用类级别的插件开发方法,按照其特有的命名规则插件命名为 jQuery. sqlInjection. js,包括三个全局函数。jQuery. sqlInject. sqlDialog 是攻击者攻击时的插件的处理函数,这里会至上而下弹出对话框来遮罩界面,同时可以设置 cookies 使其不能有二次注入的可能。jQuery. sqlInject. sqlGetInj 是防止通过 URL 的方式注入的攻击。jQuery. sqlInject. sqlPostInj 是防止通过 POST 方式提交数据注入的攻击,利用了 $(:input)获取网页中所有的输入框对象,结合 val()方法获取所有的输入信息来进行判断,这是其他语言无可拥有的优点。插件模型如图 1。

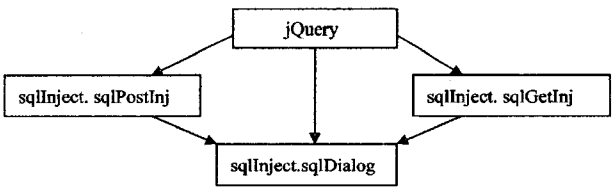

图 1　插件模型

下面是主要的源代码:

```
jQuery. sqlInject = {
sqlDialog:function( title,content,width,height,cssClassName) {
var dialogFirst = true;
if( dialogFirst = = true) {
 $ ("#containerBg"). remove( );
 $ ("#container"). remove( );
   var temp_float = new String;
temp_float = "<div id = \"containerBg\" style = \" height: " + $ ( document). height( ) +"px;filter:alpha( opacity = 0) ;opacity:0;\"></
div>";
temp_float+ = "<div id = \"container\" class = \"container\">";
temp_float+ = "<div class = \"title\"><h4></h4><span>close</
span></div>";
temp_float+ = "<div class = \"content\"></div>";
temp_float+ = "</div>";
 $ ("body"). append( temp_float) ;
dialogFirst = false;
}
 $ ("#container . title span"). click( function( ) {
   $ ("#containerBg"). animate( {opacity: "0"}, "normal", function( ) { $ ( this). hide( ) ;} ) ;
   $ ("#container"). animate( {opacity:0}, "2000", function( )
{ $ ( this). hide( ) ;} ) ;
} ) ;//动画效果,加强页面的体验
 $ ("#container . title h4"). html( title) ;
contentType = content. substring( 0, content. indexOf( ":" ) ) ;
content = content. substring ( content. indexOf ( ":" ) + 1, content.
length) ;
switch( contentType) {
case "text":
   $ ("#container . content"). html( content) ;
   break; //可以用于直接遮罩页面
case "url": // example"url:post? admin. asp? username = "+liuyi
+"&pass = "+liuyi+""
var content_array = content. split( "?" ) ;
   $ ("#container . content"). ajaxStart( function( ) {
 $ ( this). html( "program loading……" ) ;
} ) ;
 $ . ajax( {
     type:content_array[ 0] , url:content_array[ 1] , data:content_
array[ 2] ,
error:function( ) {
 $ ("#container . content"). html( "request error !" ) ;
}, //更新页面的局部区域
     success:function( returninfo) {
        $ ("#container . content"). html( returninfo) ;
     }
} ) ;
   break;
}
 $ ("#containerBg"). show( ) ;
 $ ("#containerBg"). animate( {opacity: "0. 5"}, "slow" ) ;
 $ ("#container"). attr( "class", "container"+cssClassName) ;
 $ ("#container"). css( {display: "block", left:( ( $ ( document).
width( ) ) /2-( parseInt( width) /2) ) + "px", top:( $ ( document).
scrollTop( ) -( height = = "auto"? 300:parseInt( height) ) ) + "px",
width:width, height:height, "background-color": "white"} ) ;
 $ (". content"). css( {padding: "20px 25px"} ) ;
 $ ("#container"). animate( {top:( $ ( document). scrollTop( ) +
200) + "px"}, "slow" ) ;
},
sqlGetInj:function( ) {
var urlvalue = window. location. href. toString( ) ;
var urlarray = urlvalue. split( "?" ) ;
var array1 = urlarray[ 1] ;
var sql_inj = new Array( "'", ";", " ", " or", "and", "not", "select",
"updte", "insert", "drop", "--", "sp_", "create", "function",
"grant", "revoke", "union" ) ;//JSON 数据或 XML 可以代替数
组,直接用 $ . getJSON( )来获取数据更方便
var htmlbj = "<div class = 'urlcontain' >+"array1+"</div>";
 $ ("body"). append( htmlbj) ;
 $ (". urlcontain"). hide( ) ;
for( var i = 0;i<sql_inj. length;i++) {
if( ( $ (". urlcontain"). html( ). indexOf( sql_inj[ i] ) ) >0) {
 $ . sqlInject. sqlDialog( "锁定", "text:you are attrack my website",
400,150, "cssClassName" ) ;}
}
},
sqlPostInj:function( ) {
var valueNum = jQuery( ":input" ). size( ) ;//输入框的个数
var values = new Array( ) ;
```

```
var temp="";
for(var i=0;i<valueNum;i++){
    values[i]=($(":input:eq("+i+")").val());
    temp+=values[i];
}
var sql_post_inj=new Array("'",";"," ","--","or","and",
"not","select","updte","insert","drop","sp_","create",
"function","grant","revoke","union");//JSON 数据或 XML 可
以代替数组,直接用 $.getJSON()来获取数据更方便
var htmlbj="<div class='postcontain'>+"temp+"</div>"
$("body").append(htmlbj);
$(".postcontain").hide();
for(var i=0;i<sql_post_inj.length;i++){
if(($(".postcontain").html().indexOf(sql_post_inj[i].toString
(),0))>0){
$("form").remove();//破坏 DOM 的结构
$.sqlInject.sqlDialog("锁定","text:you are attrack my website",
"400","150","cssClassName");
break;}
}
}
};
```

在任何的网页中只要导入这个插件就能实现过滤敏感 SQL 语句实现防止 SQL 注入,同时锁定网页,为了保持页面的简洁性,可以让其保存为一个页面在其他的页面中通过 include 指令来包含该页面。导入的方法如下:

```
<script language="javascript" src="jquery-1.3.2.min.js"></
script>
<script language="javascript" src="jQuery.sqlInject.js"></script
>
<script language="javascript">
$(function(){
$.sqlInject.sqlPostInj();
$.sqlInject.sqlgetInj();
});
</script>
```

jquery-1.3.2.min.js 是 jQuery 类与对象的封装,jQuery.sqlInject.js 是本插件。插件两个全局函数可以任何需要防止 SQL 注入的事件处理中。

4 结束语

文中对 SQL 注入攻击的原理、方法进行了阐述,提出了从客服端解决 SQL 注入攻击的方案,同时利用插件可扩展接口使用者只要提出自己的处理方案(或利用本插件处理方案)就能方便的把插件效果改变,但其实质是不变的,提出了 SQL 注入攻击的一种开放性解决方案,无论动态网页采用那种语言只要导入本插件就能实现保护。伴随 EXTJS 的发展,可以利用 js 的线程技术(TaskRunnner)和集合来改装本插件使其更趋轻便灵活。

参考文献:

[1] 张 楠.基于规则的检测 SQL 注入攻击方法的研究[J].陕西科技大学学报,2007,25(2):121-123.

[2] 钱林红.部分加密防御 SQL 注入攻击[J].中国科技信息,2008(24):101-102.

[3] 崔学冰. 基于 URL 重写技术的 SQL 注入攻击防御方法[J]. 河南城建学院学报,2009,18(3):63-69.

[4] 田 鼎.病毒与黑客攻防[M].北京:清华大学出版社,2006:198-203.

[5] 曾 顺. 精通 jQuery+javascript[M]. 北京:人民邮电出版社,2009:373-389.

[6] SQL Server 安全回顾[EB/OL]. 2004. http://www.microsoft.com/china/ctc/Newsletter/04/ctc2.htm.

[7] Anley C. Advanced SQL injection in SQL server applications [EB/OL]. 2002. http://www.creangel.com/papers/advanced_sql_injection.pdf. An NGS Software Insight Security Research(NISR) Publication.

[8] 陈小兵,张汉煜,骆力明. SQL 注入攻击及其防范检测技术研究[J]. 计算机工程与应用,2007(11):150-152.

[9] Sam M S. NG,SQLBlock:SQL injection protection by variable nor-malization of SQL statement [EB/OL]. 2010. http://www.iem.pw.edu.pl/~kozlowk3/biblioteczka/www_SQL/SQL_Injection_Protection_by_Variable_Normalization_of_SQL_Statement.pdf.

[10] 徐 陋,姚国祥. SQL 注入攻击全面预防办法及其应用[J]. 微计算机信息,2006(9):18-20.

[11] Finnigan P. SQL injection and Oracle[EB/OL]. 2002-11-21. http://www.oracledeveloper.nl/newforum/files/2002_11_21%20SecurityFocus%20SQL%20Injection%20and%20Oracle.pdf.

[12] Cerrudo C. Manipulating Microsoft SQL server using SQL injection[EB/OL]. 2010. http://injection.rulezz.ru/Manipulating_SQL_Server_Using_SQL_Injection.pdf.

(上接第 176 页)

[11] Krohn M, FreedMan M, Mazieres D. On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution [C]//IEEE Symposium on Security and Privacy. Berkeley, CA:[s.n.], 2004.

[12] Bhattad K, Nayayanan K P. Weakly secure network coding [C]//In:Proc. First Workshop on Network Coding, Theory, and Applications (NetCod). Hanover, USA:[s.n.],2005.

[13] Ho T, Leong B, Koetter R, et al. Byzantine modification detection in multicast networks using randomized network coding [C]//In International symposium on Information Theory (ISIT). Chicago, USA:[s.n.],2004.