doi:10.16018/j.cnki.cn32-1650/n.201902006

一种轻量级的服务端防 SQL 注入攻击方法

付熙徐,龚希章
(上海海洋大学 现代信息与教育技术中心,上海　201306)

摘要:SQL 注入攻击是针对基于数据库的网站和信息系统的一种常见攻击。通过非法的输入,攻击者可以绕开验证、非法获取内容甚至篡改系统数据。通常在客户端的验证可以被攻击者用跳过输入界面直接提交非法数据的方法攻击;而服务端的验证又会严重消耗服务器的资源。为了克服上述缺陷,通过对注入语句的分析,提出了一种轻量级的服务端验证方法,用文本挖掘的方法取得最不常见的字符串替换掉输入中的少数字符以阻止 SQL 注入攻击,同时最小化服务器用于验证输入合法性的资源。

关键词:SQL 注入;最不频繁字符串;信息安全;文本挖掘
中图分类号:TP393.08　　**文献标识码**:A　　**文章编号**:1671 -5322(2019)02 -0028 -05

SQL 注入攻击是通过在应用程序输入中插入部分 SQL 语句,改变应用程序中实际执行的 SQL 语句,以达到绕开验证、篡改数据、控制被攻击系统之目的的攻击[1-2]。由于目前很多应用系统采用的都是基于关系数据库通过 SQL 语句操纵数据的模式,因而 SQL 注入攻击被视为对 Web 应用威胁最大的攻击方式之一。

阻止注入攻击的方式通常是对输入进行判断,从而阻止非法字符的输入。然而,如果需要判断的攻击字符串过多,就容易消耗过多的资源,尤其当验证工作在需要处理大量信息的服务器上进行时,将严重影响服务器的性能。为此,本文对注入攻击的原理和注入攻击语句进行了分析,针对不同的输入设定了不同的处理方法,最终使得需要判断的攻击字符集达到最小,从而最大限度地减轻服务器的压力。

1　相关工作

1.1　SQL 注入攻击

SQL 注入攻击的攻击方法是通过非法输入改变系统中应执行的 SQL 语句。如下面的语句:

select * from user where username = ′user′ and password = ′123′

其目的是返回用户表中用户名为“user”,且密码为“123”的用户,通常用于系统用户验证。正常输入为合法的用户名和密码,但如果在密码部分输入如下字符串则可以达到跳过密码验证的目的。

1’ or ‘1’ = ’1

SQL 语句加入该字符串后,变成如下形式:

select * from user where username = ′user′ and password = ′ 1′ or ′1′ = ′1′

则失去了校验密码的功能。同理,攻击者也可插入 SQL 语句对数据和数据库结构进行修改和删除,从而对数据和系统造成严重破坏;更有甚者,攻击者可以通过数据库系统执行操作系统命令,给系统造成更严重的威胁[1]。另外,由于数据库和 SQL 的广泛使用,注入漏洞的范围也扩展到云和嵌入式系统中[3]。

总的来说,SQL 注入攻击的方式有两种:一种是直接在系统和网站客户端的表单中输入注入攻击语句,另一种是通过伪造的输入页面或网络工具将攻击语句直接提交到应用服务器中进行攻击,如图 1 所示。

1.2　注入攻击防御

注入攻击的防御通常是通过屏蔽敏感字符和

收稿日期:2018-10-08
作者简介:付熙徐(1981—),男,江西抚州人,工程师,博士,主要研究方向为人工智能、数据科学与信息系统。

词汇为主[1,4]。根据注入攻击的方式,防御可分为客户端防御和服务器端防御。

客户端防御是在客户端对数据进行限制和验证,服务器端防御是在服务端进行判断。前者需要判断的可用于攻击的字符串较多[1],也不能防止盲注攻击;后者可以防止盲注攻击,但服务端资源开销较大。

为了提高注入攻击防御的准确率,通常采用数据挖掘的方式对注入攻击进行确认[5]。但由于精准的确认开销较大,为了减轻服务器的压力,可以将精准确认与 SQL 注入的阻止错开进行。另有一些防御方式,如基于正则表达式的防御等,其本质上也是对异常输入的防御[6]。

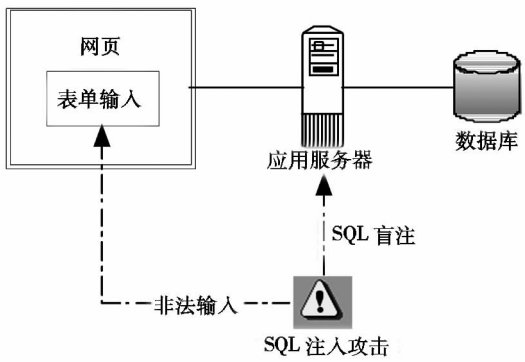

图 1　SQL 注入攻击的方式

Fig. 1　SQL injection methods

1.3　存在的问题

除去由于防御方式不同导致的盲注攻击防御问题和系统开销过大问题以外,对于合法内容的过渡防御、特殊字符串的替换及正确显示、二次注入攻击的防御也是目前各种防御方法需要改进的重要问题。

图 2 显示的是某系统对潜在注入数据进行的拦截。图 2 中,提交的数据并非注入攻击数据,但由于其包含了防御系统禁止的关键字而被禁止输入。事实上,一些包含非法字符的输入仍然是合法有意义的内容,应予以保存和显示。

2　反注入原理和评价标准

2.1　注入攻击语句和反注入关键字符集

通常来说,防止注入攻击的关键问题在于判断哪些字符串可以用于注入攻击,而很多反注入方案使用了较大的关键字符集,致使系统开销较大。本文对注入攻击语句和注入攻击关键字定义如下:

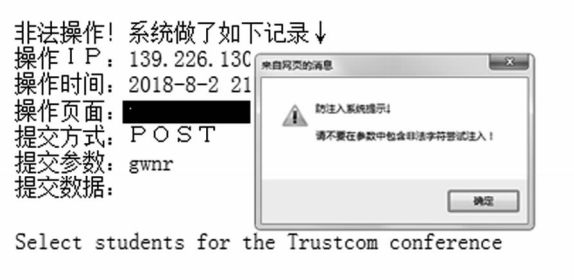

图 2　对 SQL 注入攻击的过度防御

Fig. 2　Excessive defense for SQL injection attack

定义 1 注入攻击语句是一个字符串,将该字符串作为字段的值插入 SQL 语句中会引发 SQL 执行错误或改变 SQL 语句执行的目的。本文将一个注入攻击语句记为 a,注入攻击语句集合计为 A。

定义 2 极小注入关键字符串集是一个字符串的集合 K,k 为 K 的元素,则 K 是符合式(1)的最小集合。

$$(\forall k \in K(k \notin s)) \rightarrow s \subsetneq A \tag{1}$$

2.2　反注入攻击的要求和评价标准

反注入攻击的基本要求一方面是防止入侵者修改 SQL 语句,执行跳过认证、篡改数据库、攻击操作系统等操作;另一方面,系统应该把合法的输入保存到数据库中,并在显示时尽可能显示输入的内容。

对于防止注入攻击,通常的要求是防止所有的注入攻击。对于正确显示的要求,由于可能将文本中出现的字符替换成敏感字符,故不一定可以完全恢复。通常输入的恢复率可以用式(2)来评价。

$$R = \frac{\text{正确恢复的字符数 } c}{\text{所有输入字符数 } n} \tag{2}$$

3　轻量级服务端反注入

3.1　方案总述

在服务器端对于字符型字段,若在程序中 SQL 语句字符字段前后用单引号标识,则一个注入攻击语句可用式(3)表示。

$$L = c^{*\prime}c^{*} \tag{3}$$

其中 c 为任意字符。

很显然 $K = \{'\}$ 是一个极小注入关键字符串集,也就是说,对字符型数据字段,只需判断输入是否含单引号即可确认是否为注入攻击语句;而对于非字符型输入如数字、日期时间、文件等,只

要确定输入类型正确即可防止注入。

轻量级服务端反注入编码流程如图 3 所示,该流程可保证系统不受注入攻击。

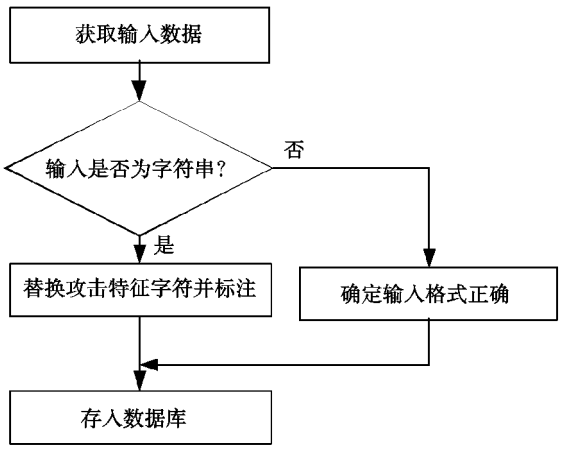

图 3 轻量级服务端反注入编码

Fig. 3 Lightweight server side anti – injection encoding

在显示时,只需将替换字符串替换回注入攻击字符串即可还原输入字符串。

3.2 用最不频繁字符串替换过滤字符

为了提高回显准确率,可以使用最不频繁出现的字符串替换输入中出现的注入攻击关键字符串。按照文本挖掘中的定义,含 n 个字符的字符串称为 n-gram 字符串,n-gram 字符串的集合称为 n-gram 集合[7]。最不频繁字符串集获取算法如下:

输入 候选字符集 $\boldsymbol{C}$,训练文本集 $\boldsymbol{T}$,频数阈值 f,最不频繁字符串数量 n,字符串最大长度 l_{max}

输出 最不频繁字符串集 U

Begin

S = C

l = 1

While l < lmax

For each s in S

For each t in T

查找 u 在 t 中频数 f_t

$f_u = f_u + f_t$

Next t

if fu < f then 将 c 加入 U 中

if U 中元素个数 > = n then return U

Next s

For each s in S

For each c in C

s′ = s + c(取字符连接到字符串 s 后面)

将 s′加入 S

Next c

将 s 从 S 中移除

Next s

Loop

End

该算法复杂度为 $O(n^3)$,但在实际运行中,通常在 1-gram 和 2-gram 中就可以找到足够的最不频繁字符串,因此程序实际复杂度接近 $O(n^2)$。另外,还可以根据出现频数对候选字符和 n-gram 集进行排序或剪枝,使得程序运行更快。

3.3 数据显示和对二阶注入攻击的防御

数据在显示时需将用于替换的字符串替换回原来的注入关键字,只需根据替换对应表完成替换即可。由于原文本中可能含有用于替换的字符串,因此仍然可能会导致部分显示错误。例如方案中用@ 字符替换单引号,但文本中有一个电子邮件地址,该地址中的@ 字符也会被错误地替换为单引号。

由于数据通常都是用于显示,故替换后数据包含一些注入攻击语句也不影响系统安全;但若数据被用于系统的控制、验证等,则可能造成二阶注入攻击[4]。为应对二阶注入攻击,最好的方式是程序再次判断数据是否合法,也可不对用于显示以外功能的数据进行注入关键字符串替换。

4 实验与结果

4.1 实验概述

为测试本文极小关键字法的有效性,建立了一套简单的系统用于测试。系统使用 Apache 2.2 作为 Web 服务器,php 5.5 作为服务端编程语言,MySQL 5.6 作为数据库。系统数据库只含一个表,表名 data,表结构如表 1 所示。

表 1 测试系统数据库结构

Table 1 Database structure of test system

字段名称	字段类型和长度	字段含义和说明
id	int(1)	记录编号,唯一标识
content	varchar(200)	消息内容
inject	bool	是否含注入字符

注:(1) 自增型变量,长度由系统决定

系统中需要输入的量为消息内容,即 content 字段。若成功插入数据库,则读取数据库中内容,恢复其中单引号字符进行显示。

进行测试的注入字符串为

');drop table data; -

若注入成功,则 data 表被删除,系统返回错误;否则,返回下内容:

Content: ');drop table data; -

经测试,该系统可以防御各种形式的注入式攻击,包括从输入界面进行的攻击、盲注以及文献[1,4]中提及的用不同形式替换注入攻击字符的攻击。

4.2　程序性能实验及结果

使用文献[4]中的敏感字符集方法与本文极小关键安法进行编码和效率对比,结果如图 4 所示。输入的文本长度分别为:短文本约为 10 字符左右,中等长度文本在 100 字符左右,长文本在 1 000 字符左右。

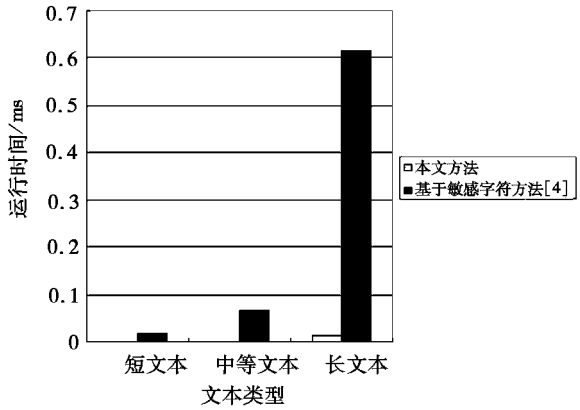

图 4　本文方法与传统防御方法的开销对比

Fig. 4　System cost compare of method in this paper and traditional method

由图 4 可知,由于需要判断的敏感注入字符串较少,极小关键字法系统开销明显要小很多,极小关键字法的效率更高。

4.3　攻击关键字替换及回显

为测试攻击关键字的替换与回显情况,采用两个不同文本库进行测试,一个是美国开放文本语料库[8],主要包含口语、小说、期刊、信件等内容。在该库中随机选取 30 篇材料作为测试集,运行程序后未发现有@ 、#等字符;再随机选取 100 篇材料,使用@ 字符替换单引号,然后将所有@ 字符替换回单引号,恢复率为 100%。

由于实际输入和小说、期刊等内容有所不同,实际输入可能会包含较多的特殊字符,因此,本文还引入了一个微博库进行测试[9]。在微博库中随机选取 20 篇材料作为测试集,总字符数 2 479 580个,其中频率较低的一些字符串如表 2。

从表 2 可知,2 字符组合的字符串中已经可以找出较多的用于替换的字符串了。

表 2　用于替换注入关键字候选字符串

Table 2　Candidate strings to replace injection keys

字符串	类型	训练集出现频率
^	1-gram	12
+	1-gram	32
~	1-gram	39
@	1-gram	107
#	1-gram	109
+ ~	2-gram	0
-^	2-gram	0
~#	2-gram	0
~^	2-gram	0
@ -	2-gram	0
~ -	2-gram	1

为测试替换的恢复率,在微博库中随机选取 100 篇材料,总字符数 12 397 965 个,其中单引号 32 293 个;使用候选字符串替换单引号,然后将所有候选字符串替换回单引号。部分候选字符串恢复率如表 3 所示。

表 3　候选字符串恢复率

Table 3　Recover rate for candidate strings　%

字符串	恢复率 R
+ ~	100
-^	100
~#	100
~^	100
@ -	99.99998
~ -	99.9997

由表 3 可知,2-gram 候选字符串恢复率极高,对内容影响几可忽略不计。

对于不同的系统,由于输入类型的不同,可以采用不同的训练集来获取最不频繁的字符串。对于公文系统只需采用前面的文章小说类训练集即可,而论坛类系统则建议使用字符包含范围更广的训练集如微博、论坛的文字集合等。

5　结束语

通过对反注入攻击方法的讨论,得出了只要禁止字符串定义的边界字符(本文中是单引号)出现在字符型数据中,并对其他类型输入进行类

型判断即可防止 SQL 注入攻击的结论。另外,关于数据的还原和显示,提出了使用最不频繁字符串替换注入关键字符串集的方法。经实验,本文的方案能够有效防止 SQL 注入攻击,并将用户的输入正确地显示出来。该方案也可推广到其他控制操纵语句与数据在同一字符串中的系统,如 DMSQL 等。

参考文献:

[1] DALAI A K, JENA S K. Neutralizing SQL injection attack using server side code modification in web applications[J]. Security and Communication Networks, 2017,2017:1-12.

[2] NAGPAL B, CHAUHAN N, SINGH N. A survey on the detection of SQL injection attacks and their countermeasures[J]. Journal of Information Processing Systems, 2017,13(4):689-702.

[3] WU T Y, CHEN C M, SUN X Y, et al. A countermeasure to SQL injection attack for cloud environment[J]. Wireless Personal Communications, 2017,96(4):5279-5293.

[4] 张慧琳,丁羽,张利华,等. 基于敏感字符的 SQL 注入攻击防御方法[J]. 计算机研究与发展,2016,53(10):2261-2275.

[5] PINZON C I, DE PAZ J F, HERRERO A, et al. IdMAS-SQL:intrusion detection based on MAS to detect and block SQL injection through data mining[J]. Information Sciences, 2013,231:15-31.

[7] 周水庚,俞红奇,胡运发,等. 基于 N-gram 信息的中文文档分类研究[J]. 中文信息学报,2001,15(1):34-39.

[6] 王伟平,李昌,段桂华. 基于正则表示的 SQL 注入过滤模块设计[J]. 计算机工程,2011,37(5):158-160.

[8] American National Corpus Project. Open American National Corpus[DB/OL]. [2018-08-15]. http://www.anc.org/OANC.

[9] The Blog Authorship Corpus[DB/OL]. [2018-08-15]. http://www.cs.biu.ac.il/~koppel/blogs/blogs.zip.

A Lightweight Server Side Method to Prevent SQL Injection

FU Xixu, GONG Xizhang

(Institute of Modern Information and Educational Technology, Shanghai Ocean University, Shanghai 201306, China)

Abstract:SQL injection attack is a common attack against database - based websites and information systems. Through illegal input, attackers can bypass authentication, illegally acquire content and even tamper with system data. In general, client - side validation can be attacked by attackers by directly submitting illegal data by skipping the input interface, while server-side validation can seriously consume server resources. In order to overcome the above defects, a lightweight server-side validation method is proposed by analyzing the injected statements. The method of text mining is used to get the least common strings and replace a few characters in the input to prevent SQL injection attacks, while minimizing the resources that servers use to validate input legitimacy.

Keywords:SQL injection; least frequent string; information security; text mining

(责任编辑:李华云)