基于 PHP 的 web 应用中的 SQL 注入及防御措施

王 娅
（厦门软件职业技术学院 福建 厦门 361024）

DOI:10.16009/j.cnki.cn13-1295/tq.2019.06.067

【摘要】互联网的发展给人民的生活带来了极大的便利，大量的数据都可以通过互联网获取，人们在便捷实用网络的同时，也面临着信息被窃取的危害。在众多网络攻击方式中，SQL 注入攻击使用频率高，成为了当前主要的攻击方式之一。基于 PHP 的 web 应用程序在使用广泛，因此 PHP 的 web 应用的安全问题受到广泛关注。本文首先对 sql 注入的方式进行了说明，然后给出了在使用 PHP 进行开发时的防护建议。

【关键词】SQL 注入；PHP；web 应用程序；安全；预处理

【中图分类号】TP31 【文献标识码】A 【文章编号】1009-5624（2019）06-0115-03

1 引言

“互联网 +“概念的提出，使得互联网在政治、经济、文化以及社会生活中发挥着越来越重要的作用。人们在便捷使用网络的同时，黑客也同样越来越活跃，形成了不少黑色产业链。网络安全问题不容忽视。

PHP，是英文超级文本预处理语言 Hypertext Preprocessor 的缩写。它常与开源免费的 Web 服务 Apache 和数据库 MYSQL 配合使用于 Linux 平台上（简称 LAMP）和 WINDOWS 平台上（简称 WAMP）。PHP 作为一种服务器端脚本编程语言，因其多平台特性以及开源免费、易学易用、开发效率高等特点，成为目前 W e b 应用开发的主流语言之一，且被广泛应用于门户、电子商务、微博、论坛等网站的开发。目前全球 5000 万互联网网站中，有 60% 以上使用着 PHP 技术。在 Web 应用的安全问题已经变得越来越重要的今天，如何对 PHP Web 应用程序进行安全防护已经成为当前研究的一个方向。

我们都知道 s q l 语言是一种解释型语言，它的数据由程序员编写的代码和用户提交的数据组成。程序员在进行 w e b 开发时，如果没有没有对用户输入数据的合法性进行全面的判，绑定变量。攻击者就可以精心构思 s q l 语句，或者通过系统报错，返回对自己有用的数据。更甚者，攻击者可以绕过网页或 W e b 应用程序的身份验证和授权，完全控制 W e b 应用程序后面的数据库服务器，来添加、修改和删除数据库中的记录，这便是我们所说的 S Q L 注入。需要注意的是，当前绝大多数的防火墙无法就 S Q L 注入发出警报。如此一来，若网络管理员未定期对 I I S 日志进行检查和查看那么在较长的时间段内，SQL 注入都是非常隐蔽的，无法被发现，因此 SQL 注入攻击有着非常大的危害性，容易造成敏感信息的泄露。

表 4-1 两种挖掘算法的比较

算法性能	准确度（Accuracy）	精度（Precision）	召回率（Recall）	RUC
贝叶斯算法	77.22%	35.06%	27.21%	0.681
决策树算法	76.89%	31.85%	31.01%	0.697

实验结论：本论文主要采用决策树算法、贝叶斯算法进行数据挖掘模型构建，通过分析学生一卡通中所包含的各属性数据记录，用训练集进行模型训练、用训练得到的分析预测模型对测试集进行预测及性能评估。通过比较两种算法模型的预测结果，可以发现在准确度（Accuracy）、精度（Precision）方面贝叶斯算法更优一些，但在召回率（Recall）、RUC 方面决策树算法要更优一些。

5 结语

数字校园的建设是一项具有重大意义的事业。校园一卡通的推广与应用是建设数字校园、实现校园信息化的重要一环，校园一卡通极大地方便了学生在校的生活和学习，方便了学校对学生的管理。一卡通数据中包含有学生就餐消费记录、图书馆图书借阅记录、超市消费记录、学业成绩情况等，本文将数据挖掘分析技术应用在校园一卡通数据分析中，基于 Rapidminer studio 平台，采用决策树与贝叶斯算法进行挖掘模型构建，通过分析一卡通记录，从中筛选出品学兼优、努力学习但是家庭贫困的学生供相关部门参考，再结合相关佐证材料，尽可能将助学金发放给这部分同学，帮助他们完成学业。

【参考文献】

[1] 徐翔，王煦法．协同过滤算法中的相似度优化方法 [J]．计算机工程，2016，36（6）：52-54.

[2] 李聪，梁昌勇，马丽．基于领域最近邻的协同过滤推荐算法 [J]．计算机研究与发展，2015，45（9）：1532-1538.

[3] T.Hofmann.Probabilistic Latent semantic analysis[c]. Proceedings of the Fifteenth Conference on Uncertainty in Artifieial Intelligence,2015.

[4] Wang Q,Xu J,Li h,er al. Regularized latent semantic indexing//Proceedings of the 34th international ACM SIGIR conference on Research and development in information Retrieval.ACM,2015,79(3):685-694.

[5] 王志新，王晓强．基于数字化校园的一卡通系统的设计研究 [J]．山西青年，2016（14）:189.

基金项目：2018 年度昆明冶金高等专科学校科研基金项目，基于校园一卡通的数据挖掘与应用（2018XJZK09）

作者简介：陈云川（1989-），男，汉族，云南玉溪人，硕士研究生，专业教师（助教），研究方向：数据挖掘与分析预测。

2 SQl注入攻击

SQl注入按照参数类型分为：数字注入和字符串注入。接下来，我们使用下面的数据来说明sql注入攻击。

```
CREATE TABLE users (
    id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    username varchar(64) NOT NULL,
    password varchar(64) NOT NULL,
    phone char(11) NOT NULL,
);
```

数据使用md5的方式对密码进行加密

```
INSERT INTO users VALUES
(null, 'tom',md5('abcd'),'18632124567'),
(null, 'peter', md5('1234') ,'18632124567'),
(null, 'sim',md5('7899'),'18632124567'),
(null, 'mary',md5('8a6c'),'18632124567');
```

数据在数据库中的显示如下：

←T→	id	username	password	phone
编辑 复制 删除	6	tom	e2fc714c4727ee9395f324cd2e7f331f	18632124567
编辑 复制 删除	7	peter	81dc9bdb52d04dc20036dbd8313ed055	18632124567
编辑 复制 删除	8	sim	7c792a8279211dece3b4df04719c818a	18632124567
编辑 复制 删除	9	mary	eaeba049bc1949c0c00db1906ebe44a7	18632124567

2.1 数字型注入

当输入的参数为整形时，如果存在注入漏洞，可以认为是数字型注入。

接下来，我们来模拟以下场景：根据url传递的参数id，查找相应用户的信息。实现该功能的代码如下：

```
$id = isset($_GET['id']) ? $_GET['id'] : '';
if(empty($id))
{
    exit('id不能为空');
}
$link = mysqli_connect($config['host'],$config['user'],$config['psd'],$config['dbname']);
mysqli_set_charset($link,'utf8');
$sql = 'select * from users where id = '.$id;
echo $sql;
$rs = mysqli_query($link ,$sql);
$data = mysqli_fetch_all($rs , MYSQLI_ASSOC);
echo '<pre>';
print_r($data);
```

上述代码在仅仅是对url上传递的参数做了一个是否为空的判断，在正确输入的情况下，我们可以获取要查找的用户的信息，但是如果我们在传递参数的时候传递这样一个 -1 or 1=1 的数据，则会构造出这样一条SQL语句：select * from users where id = -1 or 1=1。

我们选择的是满足where条件的表中的所有记录，很明显id=-1为假，对于或运算来说，会判断第二个表达式，而1=1是一个恒等式，顾where的条件恒为真。

执行这条语句的后果将会暴露这张表中的所有用户的信息。代码运行后的效果如下图所示：

2.2 字符型注入

当输入的参数为字符串时，称为字符型。字符型和数字型最大的一个区别在于，数字型不需要单引号来闭合，而字符串一般需要通过单引号来闭合的。

假设上述数据表存放的是网站后台会员的信息，我们通过表单验证用户的登录状态。实现上述功能的代码如下：

```
if(!empty($_POST)){
    $user = isset($_POST['user']) ? trim($_POST['user']) : '';  //获取表单输入的用户名
    $psd = isset($_POST['psd']) ? trim($_POST['psd']) : '';     //获得表单输入的密码
    if(empty($user) || empty($psd))
    {
        exit('请输入用户名和密码');
    }
    $link = mysqli_connect($config['host'],$config['user'],$config['psd'],$config['dbname']);
    mysqli_set_charset($link,'utf8');
    $sql = 'select * from users where username = "'.$user.'" and password = "'.md5($psd).'"';
    echo $sql;
    echo '<br/>';
    if(!$rs= mysqli_query($link,$sql))
```

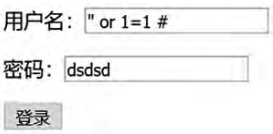
```
{
    exit(mysqli_error($link));
}
while($row=mysqli_fetch_row($rs)) {

    echo ' 登录成功';
    echo '<pre>';
    print_r($row);
}
mysqli_free_result($rs);
mysqli_close($link);
```

上述代码对于用户在表单中输入的数据也仅仅是作了非空判断，输入正确的用户名和密码则会显示登录成功，为了测试方便，我们将对应的 sql 语句和该用户的信息一并输出。

尽管对于用户的密码，我们采用了 md5 加密，然而，我们却可以构造 sql 语句，绕过用户名和密码的验证。

我们在输入框中输入信息，如下图所示：



则会在在不知道用户名和密码的情况下实现了用户登录。并获取了所有后台用户的信息。运行后的效果如下图所示：

```
select * from users where username = '" or 1=1 #' and password = 'cef468eeda569cc1b16b45fd53200b9c'
登录成功
Array
(
    [0] => 6
    [1] => tom
    [2] => e2fc714c4727ee9395f324cd2e7f331f
    [3] => 18632124567
)
登录成功
Array
(
    [0] => 7
    [1] => peter
    [2] => 81dc9bdb52d04dc20036dbd8313ed055
    [3] => 18632124567
)
登录成功
Array
(
    [0] => 8
    [1] => sim
    [2] => 7c792a8279211dece3b4df04719c818a
    [3] => 18632124567
)
登录成功
Array
(
    [0] => 9
    [1] => mary
    [2] => eaeba049bc1949c0c00db1906ebe44a7
    [3] => 18632124567
)
```

3 SQL注入的防范措施-预处理

通过上述两个实例，可以看到，sql 注入攻击不是针对系统 BUG 来开展，而是针对程序员在编写程序时的不严密。要想防止 sql 注入攻击，一种方法是对输入的数据进行检验：开发者可以在编写代码时对输入的数据进行类型的检验，对特殊字符进行过滤来避免特殊内容的输入。另外一种方法就是将 SQl 语句的解析和数据的传递分开，这就是我们接下来要说明的 PHP 的预处理机制。

下图演示了预处理语句和传统方式的区别。

3.1 相比于直接执行 SQL 语句，预处理语句有两个主要优点：

（1）节省资源：预处理语句对于 sql 语句只解析一次，后续仅仅传递变化的参数。

（2）防止 SQL 注入：一条传统的 SQL 语句被分成了语句和数据两个部分分别执行，保证了数据的合法性。

3.2 预处理语句的工作原理如下：

（1）预处理：创建 SQL 语句模板并发送到数据库。预留的值使用占位符 "?" 标记 。

（2）参数绑定：将参数绑定到预处理语句中，这里需要指定传入参数的类型。

（3）执行：将绑定的值传递给占位符，执行数据库的操作。如果参数的值不一样，执行语句可以多次执行。

3.3 PHP 中 MYSQLi 扩展提供了面向对象和面向过程两种风格的预处理语句。在面向过程编程时，预处理相关函数有：

（1）mysqi_prepare() 函数用于预处理一个待执行的 SQL 语句。当函数执行后，成功返回预处理对象，失败返回 false。

（2）mysqli_stmt_bind_param() 函数用于将变量作为参数绑定到预处理语句中。

该函数执行成功时返回 true，失败时返回 false。

（3）mysqli_stmt_execute() 函数用于执行预处理。当函数执行成功后，返回 true，执行失败返回 false。

（4）mysqli_stmt_get_result（） 函数用于获得结果集。当执行 select、show、describe 或 explain 查询时，需要对查询后的结果集进行处理。

【参考文献】

[1] 传智播客 . PHP+MYSQL 网站开发项目式教程 [M]. 北京：人民邮电出版社，2016 年 8 月 .

[2] 叶根梅 . 基于 PHP 的 Web 网站表单安全防御策略 [J]. 淮阴师范学院学报（自然科学版），2019，18(01):30-34.

[3] 丁允超，范小花 . SQL 注入攻击原理及其防范措施 [N]. 重庆科技学院学报（自然科学版），2012，14(05):136-139.

作者简介：王娅（1983-），女，汉族，山东菏泽人，硕士，讲师、工程师，研究方向：计算机应用方向，大数据技术。