Proposed Methods To Prevent SQL Injection

A. H. Mohmmed
Department of Computer Science, College of Education, University of Al-Mustansiriyah
Received in : 3, October , 2010
Accepted in : 8, February, 2011

Abstract

In the last decade, the web has rapidly become an attractive platform, and an indispensable part of our lives. Unfortunately, as our dependency on the web increases so programmers focus more on functionality and appearance than security, has resulted in the interest of attackers in exploiting serious security problems that target web applications and web-based information systems e.g. through an SQL injection attack.

SQL injection in simple terms, is the process of passing SQL code into interactive web applications that employ database services such applications accept user input such as form and then include this input in database requests, typically SQL statements in a way that was not intended or anticipated by the application developer that attempts to subvert the relationship between a webpage and its supporting database, in order to trick the database into executing malicious code due to the poor design of the application.

The proposed system is based on protection website at run time, before inclusion of user input with database by validating, encoding, filtering the content, escaping single quotes, limiting the input character length, and filtering the exception messages. The proposed solution is effectiveness and scalability in addition it is easily adopted by application programmers. For empirical analysis, we provide a case study of our solution and implement in Html, PHP, My Sql , Apache Server and Jmeter application.

Key words:web site security, Data Base Server, SQL Injestion attack

Introduction

The Internet has brought about many changes in the way organizations and individuals conduct business, and it would be difficult to operate effectively without the added efficiency and communications brought about by the internet [1]. In the last few years, the popularity of web-based applications has grown tremendously. A number of factors have led an increasing number of organizations and individuals to rely on web-based applications to provide access to a variety of services. Today, web-based applications are routinely used in security critical environments, such as medical, financial, and military systems [2]. Because of the popularity of these types of applications many techniques to exploit their security vulnerabilities are potentially quite dangerous. One such technique is called SQL injection [3]. SQL Injection attack has been one of the major threats to the security of web applications and attackers can trick server into executing malicious SQL code which is [4]. occurs when user input is parsed as SQL tokens, thus changing the semantics of the underlying query [3]. SQL injection attacks have been used to extract customer and order information from e-commerce databases or bypass security mechanisms. The intuition behind such attacks is that predefined logical expressions within a predefined query can be altered simply by injecting operations that always result in true or false statements [5].

This paper, presents a runtime technique to prevent SQL injection observe that all SQL injections alter the structure of the query intended by the programmer and by capturing this structure at runtime, we can compare it to the parsed structure after inserting user-supplied input, and evaluate similarity. Evaluated the proposed system based on user input on a set of real-world applications without requiring a call to the database, thus lowering runtime costs and satisfy the following three criteria:

1. prevent the possibility of the attack
2. Minimize the effort required by the programmer
3. Minimize the runtime overhead.

This paper is structured as follows: The next section reviews related work and section 3 describe web server technology and section 4 SQL injection working while section 5 describe proposed approach that characterizes the sanitization process by modeling the way in which an application processes input values and provides details about the implementation of system, section 6 presents the experimental results that show approach is feasible in practice, section 7 concludes the paper.

-Web Server Technology

Web based systems are a composition of infrastructure components, web servers , databases, and of application specific code, such as HTML-embedded scripts and server-side CGI programs[2]. Nowadays, lots of websites are interactive, dynamic and database-driven, which run various web applications in servers with data stored in back-end database. Web 2.0 technologies allow users to do more than just retrieve information. They can access and modify the content and distribute their information in websites such as social networking sites, wikis and blogs. In other words, they can control the database information via a web browser [6].

Web applications accept user input via forms in web pages. This input is posted to the server as name value pairs, both of which are strings. An alternate mechanism to pass information to the server is the query string. The query string is information appended to the end of the URL. On most web servers, a question mark separates the resource from the query string variables. Each name value pair in the query string is separated by an ampersand, and the user is free to edit this input as easily as form inputs. Because it is common for web servers can differentiate between variables passed in the query string and those posted in the form, then will consider both as user input [3].Web-based applications represent a serious security exposure. These applications are directly accessible through rewalls by design, in addition[7]. The infrastructure components are usually developed by experienced programmers with solid security skills, the application specific code is often developed under strict time constraints by programmers with little security training as a result vulnerable web-based applications are deployed and made available to the whole internet, creating easily exploitable entry points for the compromise of entire networks [2].

The Internet has brought about problems as the result of intruder attacks, both manual and automated, which can cost many organizations excessive amounts of money in damages and lost efficiency [1]. Web-based attacks aimed at either obtaining control of the host running the web server application (e.g., through a buffer overflow) The first type of attack is caused by vulnerabilities in the web server software or in a server-side web-based application that allow one to compromise the security of the underlying host ,The second type of attack is [7] offer the vulnerabilities of unauthorized database control and malicious code injection which attackers can take advantage of attackers are trying to get valuable information held in database, this hack is a kind of application attack called SQL injection[6].

-SQL Injection Working

It is very hard to understand the conceptual idea of SQL injection without partially understanding the code that runs in the background [8]. A database computer language designed for the retrieval and management of data in relational database management systems (RDBMS) [6].

Structured Query Language (SQL) is used for many database systems including Microsoft SQL Server, Oracle, MySQL and even Microsoft Access[8]. SQL injection is yet common vulnerability that is the result of lax input validation. Unlike cross-site scripting vulnerabilities that are ultimately directed at site's visitors, SQL injection is an attack on the site itself in particular its database [9].

In 2008, there was a significant increase in the number of websites affected by SQL injection attacks.This increase can be attributed in part to the development of automated tools that allowed attackers to test and compromise sites much faster than older manual methods.There are specific examples of SQL injection events that occurred in april 2008 attacks against microsoft internet information services (IIS) that affected more than half a million websites and in december 2008 microsoft internet explorer 7 (IE7) that was leveraged via SQL injection attacks [10].

SQL injection attacks are a prime example of malicious input that changes the behavior of a program by introduction of query structure into the input strings. An application that does not perform input validation (or employs error-prone validation) is vulnerable to SQL injection attacks. Although useful as a first layer of defense, input validation often is hard to get right, The absence of proper input validation has been cited as the number one cause of vulnerabilities in web applications [11]. A successful SQL injection exploit can read confidential data from the database ,modify database data (INSERT/UPDATE/DELETE), execute administration operations on the database such as shutdown of database management system (DBMS) , recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system, It may also lead to many potential attacks in other forms[6].

-Proposed System

A web application is vulnerable to an SQL injection attack if an attacker is able to insert SQL statements into an existing SQL query of the application. This is usually achieved by injecting malicious input into user fields that are used to compose the query, login page prompts the user to enter her username and password into a form typically used for checking the user login credentials therefore, are prime targets for an attacker. In this example, if the login application does not perform correct input validation of the form fields, the attacker can inject strings into the query that alter its semantics. For example, consider an hacker entering "OR 1=1"-- one of sql injection string as in the tabel(1), the "--" command indicates a comment in Transact-SQL. Hence, everything after the first "--" is ignored by the SQL database engine with the help of the first quote in the input string, the user name string is closed, while the "OR 1=1" adds a clause to the query which evaluates to true for every row in the table. When executing this query, the database returns all user rows, which applications often interpret as a valid login. So in the proposed system all client-supplied data needs to be cleansed of any characters or strings that could possibly be used maliciously as shown in the figure(1).

In this section list proposed methods can be applied to minimize the risk of a SQL injection attack .

1. Validation Input

The first step in any form processing script should check the syntax of input for validity to verify that the input is a valid input in the language. The validation could be anything such as checking whether the entered value for "password" is a number, and is 16 or over, or making sure the username column has only valid characters such as A to Z, a to z and many classes of input have fixed languages such Email addresses, dates etc .

2. Encoding Input

Sometimes input might contain some illegal characters, or it might not always be viable to validate all user input for example, in a search field the user could type anything that they are searching for, including script tags such as <script> ,encoding is the best way to neutralize

harmful data from user input, since encoding translates the harmful characters into their display equivalents for example the < character will be translated into < character. The HtmlEncode method of the server object can be used to encode the harmful characters.

3. Filtering Input

If an application does not filter user input before it is used in a SQL query, then SQL injection vulnerability occurs because server received incorrect input from an untrusted client.So proposed system prevent SQL injection attacks by filtering user input using language-supplied input filtering functions before using untrusted input in a SQL query. For example, let's suppose to filter out special characters such as the following: [ ] { } ; &. to achieve this functionality use the Replace method of the String object this code will replace all the unwanted characters from the users input.

4. Limit the Length of User Input

Make use of the maxlength attribute of the textbox controls will be restricting the number of characters the hacker can type. In proposed system all text boxes and form fields should be always as short as possible for example, if the firstname textbox should only accept 40 characters, then enforce the length in the maxLength attribute. This will restrict the hacker to send small set of commands back to the server

<input type="text" id="txtFirstname" MaxLength="40" runat="server "/>

5. Modify Error Reports

In situations where the attacker has no knowledge of the underlying SQL query or the contributing tables, hacker try to tamper with the SQL statement in order to receive an error message and can gain more information about the SQL statement, and can start tampering with the SQL statement. The proposed system process error reports properly and configure in such a way that error cannot be shown to outside users and display of errors should be restricted to internal users only such as

1. display_errors = Off
2. display_startup_errors = Off
3. log_errors = On
4. log_errors_max_len = 0
5. ignore_repeated_errors = Off
6. ignore_repeated_source = Off
7. track_errors = Off
8. error_log = /var/log/php.log --OR—syslog

6. Low privilege

Limit database permissions and segregate users and never allows to connect as a database administrator in web application will limit the damage potential.

The proposed system has three pages , the code of first page shown in figure(2) is an HTML page with a form element. The form asks the visitor to enter his MySQL user name and password.The name of this page is login.html as shown in figure(3) , when user full the data and click login button the data will send by post method to the second page that called validentry.php which works in hide,code of this page shown in figure(4).

-Evalution

Web-based applications have become a popular means of exposing functionality to large numbers of users by leveraging the services provided by web servers and databases. The wide proliferation of custom developed web-based applications suggests that a approach for providing early warning and real-time blocking of sql injection exploits. The proposed system composed of a number of methods used to prevent sql injection attack. The proposed system evaluated its applicability with respect to several existing web-based applications according to four key metrics:

1- Safety - How well a technique resists being circumvented or defeated when faced with a competent attacker
2- Speed - How high the performance overhead of a technique is
3- Flexibility - How applicable a technique is to a range of different security flaws and vulnerabilities
4- Practicality - How easy it is to apply a technique to modern software settings, where source code access may not be available and legacy code may be present.

The proposed system tested at run time by Apache's JMeter testing,this application allows one to schedule walks through a web application, including detailed web form submission and thus database querying. JMeter can execute script with a configurable number of concurrent users. It also allows the tester to configure the delay between requests for the simulated users.Tabel(2) shows the response times to execute two type of query as in table(3) with number of users.

Conculsion

A criminal can break into a system and wreak havoc on a network or computer system different ways. It is up to the web application developers to do their part in making sure the applications they design are not vulnerable to any known threats, In general proposed system is not limited to any specic platform since it does not rely on any particular language mechanism or technology. This strategy can be instantiated for any existing web application framework.

References

1. goheer,J. (2009), SQL Injection and Cros Site Scripting , Kualitatem Pvt Ltd.
2. Cova, M.; Felmetsger,V. and Giovanni Vigna,(2007), Vulnerability Analysis of Web-based Applications, University of California.
3. Buehrer, G.; Weide,W.; Paolo,A. and Sivilotti, G. (2010), Using Parse Tree Validation to Prevent SQL Injection Attacks" ,Computer Science and Engineering of Ohio State University .
4. Xiang, F.; Qian,K. (2008), SAFELI – SQL Injection Scanner Using Symbolic Execution,School of Computing and Software Engineering Southern Polytechnic State University .
5. Stephen,W. and Keromytis, D. (2010), SQLrand: Preventing SQL Injection Attacks, Department of Computer Science Columbia University.
6. Ronald,L. (2008), SQL Injection, Hong Kong Computer Emergency Response Team Coordination Centre.
7. Kruegel ,K. ; Valeur ,F. and Barbara,S. (2007), An Anomaly-driven Reverse Proxy for Web Applications, University of California.
8. Mrowton,K. (2005) , Introduction to SQL Injection, Lee Lawson .
9. lia,I. (2005), SQL Injection, Ilia_Security.indb
10. Finch,N. (2009), SQL Injection , US-CERT government organization
11. Bisht, P.; Prasad,A. and Venkatakrishnan, V. (2010), Automatically Preparing Safe SQL Queries, Department of Computer Science University of Illinois, Chicago, USA.

Tabel (1): Example of sql Injection	
admin' --	') or ('1'='1—
admin' #	') or ('1'='1
admin'/*	"or "1"="1
' or 1=1--	' or '1'='1
' or 1=1#	Or 1=1--
' or 1=1/*	" or 1=1--
') or '1'='1--	' or 1=1--

Table (2): Compare between Normal SQL Query & SQL Injection

Number of User	Query TYPE	Response Time (ms)
1	Normal SQL Query	70
1	SQL Injection	75
5	Normal SQL Query	387
5	SQL Injection	399
10	Normal SQL Query	792
10	SQL Injection	810

Table (3) : Example of normal sql query and sql injection attack

Normal SQL Query

Select * from mytable where user name = `ahmed` and password =`12345`;

SQL Injection

Select * from mytable where user name = ``OR 1=1; --` and password=`dummy`;

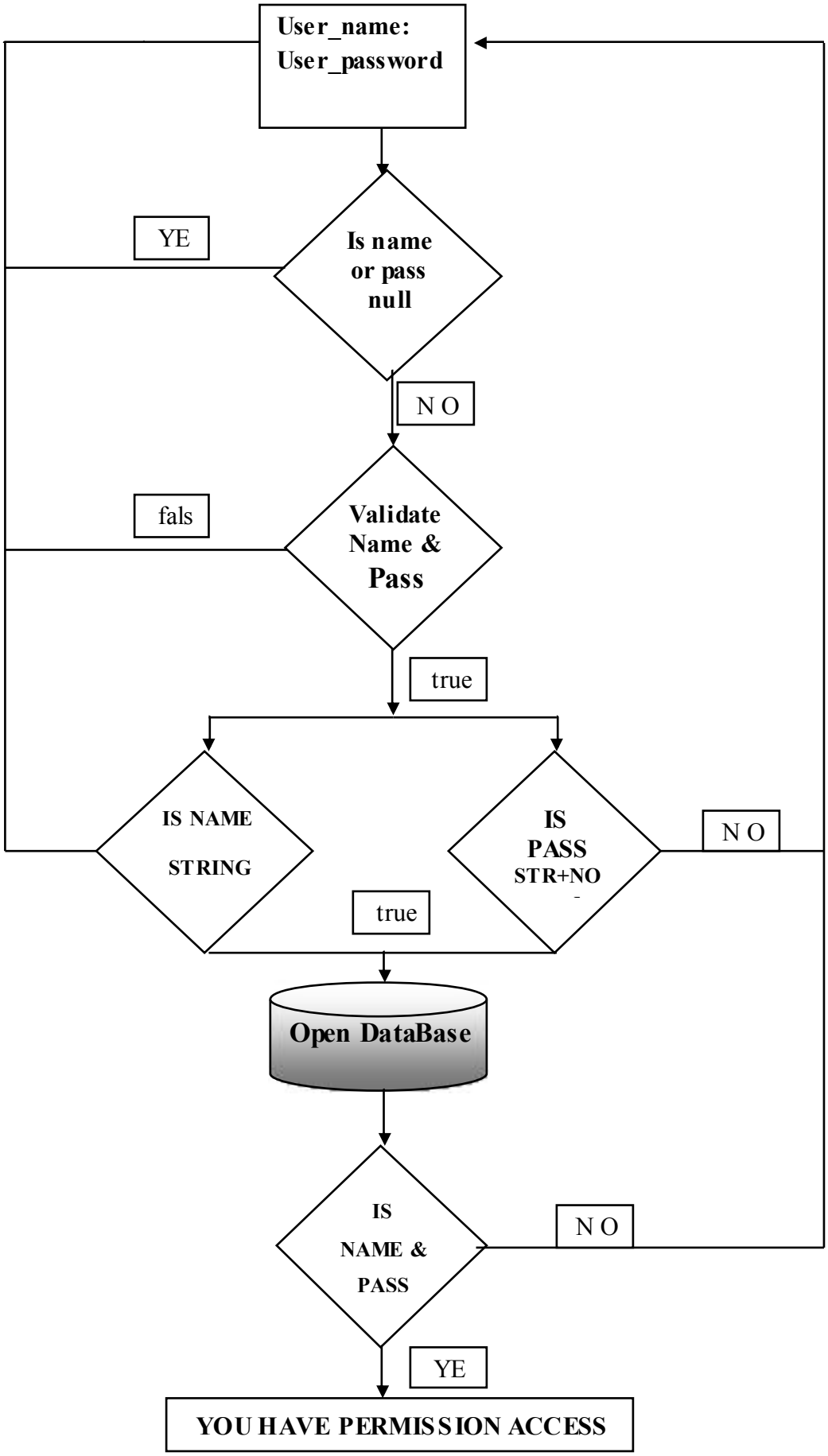

Fig. (1): Proposed System

```
<html>
<head>
<title>user Login </title>
</head>
<body>
<form method="POST" action="validentry.php" name="Login">
<p >enter user name and password to enter the page</p>
<p >user name :<input type="text" name=" user _name"  size="20"></p>
<p>password:<input type="password" name=" user _password"  size="20"></p>
<p><input type="submit" value="send" name="Login"></p>
</form>
</body>
</html>
```

Fig.(2): Code of login

Fig.(3): login page

```
<? php
If ($user_name=="") or ($user_password=="") header("location: login.html");
If not(is_string($ user_name )) header("location: Login.html");
if (!ctype_alnum($_POST[‘ user_password ’])) header("location:  Login.html ");
Input_filtering($ user_name )
Input_filtering($ user_password )
$esc_name = mysql_real_escape_string($ user_name );
$esc_password = mysql_real_escape_string($ user_password );
mysql_connect("localhost", "", "") or die("Could not connect: " . mysql_error());
mysql_select_db("project");
```

```
$result = mysql_query("SELECT * FROM admin where name='$esc_name' && password='$esc_password
‘");
$row = mysql_fetch_array($result, MYSQL_BOTH);
if (($row["username"]==Null)or($row["userpassword"]==Null))
header("location: login.html");
function validate_string( input )
known_bad = array( "select", "insert", "update", "delete", "drop", "union, ";",”"--","xp_",”*”,"'" )
for i = lbound( known_bad ) to ubound( known_bad )
if ( instr( 1, input, known_bad(i), vbtextcompare ) <> 0 )  header("location: admin.php");
next
end function
?>
```

Fig.(4): Code of validentry.php

حماية تطبيقات الويبِ من هجومِ حقن (SQL)

أحمد هاشم محمد
علوم الحاسبات ،كلية التربية،الجامعة المستنصرية
استلم البحث : 3،تشرين الاول ، 2010
قبل البحث: 8،شباط، 2011

الخلاصة

في العقد الأخير،أَصْبَحَ الويب جزءاً لا غنى عنهَ في حياتِنا. ولكن لسوء الحظ،الاعتمادية على الشبكة زادت من عدد المبرمجين الذين يُركّزونَ على تصاميم المواقع أكثرَ من التركيز على الوظيفةِ والحماية مما أدّى إلى زيادة عدد المهاجمين في إِسْتِغْلال مشاكلِ الحماية التي تَستهدفُ تطبيقاتَ الويبِ وأنظمةِ المعلوماتِ على الإنترنتِ ومثال على ذلكِ الهجوم هو (SQL injection).

هجوم (SQL injection) هو عمليةُ تمرير رمزِ لغة الإستفسار البنائيةِ (SQL) إلى تطبيقاتِ الويبِ التفاعليةِ التي تَستعمل قاعدةَ البيانات، اذ يُحاولُ المخترق تَخريب العلاقةِ بين الموقع وقاعدة بياناته المساندة، لكي يَخْدعَ قاعدةَ البيانات إلى تَنفيذ الرمزِ الخبيثِ بسبب التصميمِ السيّئِ للتطبيقِ.

إنّ النظامَ المُقتَرَحَ مستند الى التَدقيق في وقتِ التشغيل، قبل إدراجِ الزبون المتصل بقاعدةِ البيانات باستعمالِ طرائق عديدة لجَعْل التضمين صالح للإستعمالِ وذلك من خلال المصادقة، التَشْفير، تَرشيح المحتوى عن طريق الغاء الإقتباساتِ الوحيدةِ، تحدّيدُ طول الحرفَ المتضمنة، وتصفية رسائل الخطاء . إنّ الحَلَّ المُقتَرَحَ فعال وقابل للتوسع فضلا عن انه يُتبنّى بسهولة مِن قِبل مبرمجي التطبيقِ اذ استعملت لغة HTML PHP, قاعدة بيانات MySql، و الخادم Apache في تصميم الموقع واختبار الحل المقترح.

الكلمات المفتاحية : امنية المواقع ، قواعد البيانات الخادم ، هجوم حقن SQL