Machine Learning for SQL Injection Prevention on Server-Side Scripting

Krit Kamtuo and Chitsutha Soomlek
Department of Computer Science
Khon Kaen University
Khon Kaen, Thailand
krit.kamtuo@kkumail.com and chitsutha@kku.ac.th

Abstract— **SQL injection is the most common web application vulnerability. The vulnerability can be generated unintentionally by software developer during the development phase. To ensure that all secure coding practices are adopted to prevent the vulnerability. The framework of SQL injection prevention using compiler platform and machine learning is proposed. The machine learning part will be described primarily since it is the core of this framework to support SQL injection prediction by conducting 1,100 datasets of vulnerabilities to train machine learning model. The results indicated that decision tree is the best model in term of processing time, highest efficiency in prediction.**

Keywords-SQL Injection; Machine Learning; Web application vulnerability

I. Introduction

The most common web application vulnerability found in the National Security Agency (NSA) report is SQL injection [1]. An attacker uses specifically crafted inputs that are considered as database queries to gain unauthorized access and execute SQL database via a web application. As the results, an intruder can alter data, reveal confidential information, and attack the internal database. Typically, web application developers try to prevent SQL injection by implementing input sanitization or use personal expertise on coding. However, web application is still vulnerable if the input sanitization or filtering is not being able to either fulfill or predict the trend of vulnerability over the web application code.

Lee demonstrated the types of SQL injection attacks and examples in SQL Injection commands [2] as follows:

A. Illegal/Logically Incorrect Queries

This attack derives the CGI tier replies error message by inserting a crafted SQL commands such as query 1.

- Query 1:
  *SELECT * FROM user WHERE id= '1111' AND password= '1234' AND CONVERT (char, no) --;*

B. Union Queries

This type of SQL injection uses the ''Union'' operator which performs unions between two or more SQL queries. The attack performs unions of malicious queries and a normal SQL query with the ''union'' operator. Query 2 shows an example.

- Query 2
  *SELECT * FROM user WHERE id= '1111' UNION SELECT * FROM member WHERE id= 'admin'--' AND password= '1234';*

All subsequent strings after -- are recognized as comments, and two SQL queries are processed in this example. The result of the query process shows the administrator's information in the DBMS.

C. Piggy-Backed Queries

This attack inserts malicious SQL queries into a normal SQL query. It is possible because many SQL queries can be processed if the operator '';'' is added after each query. Query 3 is an example. Note that the operator '';'' is inserted at the end of query.

- Query 3:
  *SELECT * FROM user WHERE id= 'admin' AND password= '1234'; DROP TABLE user; --;*

The result of query 3 is to delete the user table.

- Query 4:
  *CREATE PROCEDURE DBO @userName varchar2, @pass varchar2, AS EXEC ("'SELECT * FROM user WHERE id='" + @userName + " 'and password='" + @password + "'); GO*

Query 4 is also vulnerable to attacks such as Piggy-backed queries. However, the vulnerability in stored procedures is not tested in this study.

Typically, the vulnerability can be produced unintentionally by software developers under a web application development phase using unsecured coding practices [3]. To guarantee that all secure coding practices are adopted to prevent the vulnerability, before they are unintentionally added into the production code, the framework of SQL injection prevention in Illegal/Logically Incorrect Queries, Union Queries, and Piggy-backed Queries on server-side scripting using compiler platform and machine learning is proposed. The machine learning part will be described primarily in this paper.

978-1-5090-4420-7/16/$31.00 ©2016 IEEE

II. RELATE WORK

A. *SQL injection and cross-site scripting vulnerabilities prediction and detection*

Shar and Tan have found the vulnerability detection approaches based on static and dynamic taint analysis techniques produce too many false alarm and too complex in commercialization perspective [4]. Therefore, Shar and Tan proposed the framework called "PhpMinerI" for SQL injection (SQLI) and cross site scripting (XSS) vulnerabilities prediction in PHP server-side script using machine learning [4].

C4.5, Naïve Bayes (NB), and Multi-Layer Perceptron (MLP) were used as the machine learning models in the framework to predict and detect the vulnerabilities on eight PHP standard open-source web applications to evaluate efficacy of detection and prediction in the vulnerabilities. The benchmark of each machine learning model revealed the best machine learning model due to indication of the highest accuracy and lowest false alarm is MLP for prediction SQL injection and XSS on average probability of detection in SQL injection at 93%, probability of false alarm in SQL injection at 11%, probability of detection in XSS at 78%, and probability of false alarm in XSS at 6% [4].

MLP is one of machine learning algorithms which is supposed to be more effective than traditional testing if the model in machine learning is effectively trained [2].

However, Shar and Tan did not specify the types of SQL injection that their research can resolve.

B. *A novel method for SQL injection attack detection based on removing SQL query attributes values*

Lee proposed a method to remove the attribute values of SQL queries at runtime using dynamic method and compares them with the SQL queries analyzed in advance of using static method [2]. The result is rule-based method to remove the attribute in SQL queries for SQL injection analysis. However, the method cannot validate SQL syntax before detecting SQL injection.

C. *Microsoft Azure Machine Learning*

Microsoft Azure Machine Learning is cloud based predictive service that provides full-managed model predictive analytics and predictive models as web services [10].

Microsoft Azure Machine Learning provides tools for creating predictive analytics solutions by creating analytics workflow module. Data can retrieve from the various type of data sources to build analytics and predictive models. Finally, the validated analytics and predictive models are deployed as web services which can connect to applications or websites, or provide insights in business intelligence [10].

III. RESEARCH METHODOLOGY

The framework is designed for SQL commands datasets extraction to mark as input attribute. The input attribute will be sent to the machine learning models as well as prediction of SQL injection is reported. To explain the overview of the framework, Figure 1. demonstrate overview of the framework

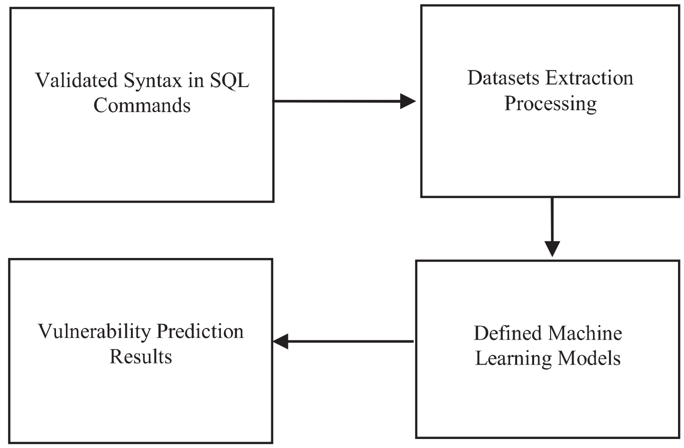

Figure1. Overview of the framework

This section explains the methodologies used for building the proposed framework in machine learning part. The framework is designed to support the ability to validate SQL syntax, detect, and predict SQL injection in web application development. The techniques used for SQL injection detection and prediction, e.g., machine learning models, research methods, and plan are described.

A. *SQL Injection Commands Datasets Extraction*

SQL commands consist of variables that can be classified into 2 categories:

- Fixed variable
  A fixed variable is a variable that receives value internally within server-side scripts. This type of variable does not directly involve with SQL injection.
- Dependent variable
  A dependent variable is a variable that receives value from user input, thus, there is a possibility that SQL injection could happen as shown in a demonstration of dependent variable in SQL command in below.

*"SELECT * from db_user where username = '$input_username' and password = '$input_password'"*

This command receives two variables from the user:
$input_username and $input_password.

If the user injects the command:
$input_username = "krit"
$input_password = " " ; drop table users --

Then, these SQL commands will be included into:
*SELECT * from db_user where username = 'krit' and password = '_'; drop table users --*

Finally, the code has been injected by another SQL command.
*SELECT * from db_user where username = 'krit' and password = '_'; drop table users --*

Input code attributes will be marked manually and extracted for machine learning model analysis which will form the characteristics of the input. Thus, the attributes of vulnerable SQL commands are proposed by 20 input attribute types showing in the TABLE I. in below.

TABLE I. INPUT ATTRIBUTE TYPES

No.	Attribute	Description
1.	Single Line Comment	Single line comment. Ignores the remainder of the statement.
2.	Semicolon	A query termination.
3.	Three Single quote	Three single quote (‘’’) in SQL query
4.	Two Single Quotes	Two Single Quotes (‘’) in SQL query
5.	Separated Two Single Quotes	Separated Two Single Quotes (‘ ’) in SQL query.
6.	Number equals to the same number, e.g., 0=0	Condition which is always return true.
7.	Number equals to the same number, e.g., 1=1	Condition which is always return true.
8.	Character equals to the same character, e.g., ‘x’= ‘x’	Condition which is always return true.
9.	Variable equals to the same variable, e.g., a=a	Condition which is always return true.
10.	Character equals to the same character, e.g., ‘a’= ‘a’	Condition which is always return true.
11.	Double quote	Double quote (“) in SQL query.
12.	Comment delimiter	Comment delimiter (/*) in SQL Query. Text within comment delimiters is ignored
13.	Semicolon and SET IDENTITY_INSERT commands	; SET IDENTITY_INSERT commands in SQL query
14.	Semicolon and TRUNCATE TABLE commands	; TRUNCATE TABLE commands in SQL query
15	Semicolon and DROP TABLE command	; DROP TABLE commands in SQL query
16.	Semicolon and UPDATE command	; UPDATE commands in SQL query
17	Semicolon and INSERT INTO command	; INSERT INTO commands in SQL query
18	Semicolon and DELETE command	; DELETE command in SQL query
19	UNION command	UNION commands in SQL query
20	PiggyBackedQuery or IllegalQuery or UnionQuery	To indicate type of SQL injection, e.g., Illegal query, Union query, and Piggy-backed query

The attributes are collected from HTML code and SQL code in existing CMS applications, e.g., WordPress [5], Drupal [6], Joomla [7], and Simple Machine Forum [8]. Finally, vulnerable SQL commands are extracted to the dataset. 0 and 1 stand for availability of each attribute in sample SQL query, 1 means available attribute and 0 means unavailable attribute.

The samples of dataset in each vulnerability type are performed as Figure 1. below:

The selected models will be analyzed and compared to find the model that can produce high accuracy and low false alarm in web vulnerabilities prediction and detection. The models will be analyzed through static code analysis to identify a potential design, globalization, interoperability, performance, security, and other categories of potential problems. The analysis measures the prediction models performance as follows:

SQL Query	SingleLine Comment	Semicolon	Three Single Quote	Two Single Quote	Seperated Two Single Quote	True Case Zero	True Case One	True Case CharX	True Case VarA	True Case charA	Double Quote	Multiple Line Comment	SET IDENTITY_ INSERT	TRUNCATE TABLE	DROP table	UPDATE	INSERT into	DELETE	union	Union Query
SELECT * from test.members where user_name=' ' or 0=0 #'	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SELECT * FROM customers WHERE username = ''; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --' and password = ''	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
SELECT * FROM product WHERE PCategory=''; DROP table Username --'	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
SELECT * FROM newsletter WHERE email = ' 'sqlvuln'	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SELECT * from test.members where user_name=' 1' AND non_existant_table = '1 '	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1. The samples of dataset

B. Preprocessing

Since, the cost of an invulnerable is four times larger than the cost of a vulnerable. To address for this, generation of a new balanced dataset that reflects this cost function is performed by using Synthetic Minority Over-Sampling Technique (SMOTE) [9] with SMOTE percentage value is 252.

C. Machine learning model analysis for SQL Injection prediction and detection

1,100 samples of vulnerable SQL commands will be used to train machine learning models which are designed for SQL injection, prediction, and detection. The initial properties in each machine learning model are initiated per the default values of the machine learning module in Microsoft Azure Machine Learning, the cloud-based predictive analytics [10].

The machine learning models used in this research are summarized in TABLE II.

TABLE II. SELECTED MACHINE LEARNING MODELS

Model Name	Properties
Support Vector Machine (SVM)	1. Number of iterations = 1,000 2. Lambda = 0.001
Boosted Decision Tree	1. Maximum number of leaves = 80 2. Learning rate = 0.1 3. Number of trees construct = 100
Artificial Neural Network	1. Number of hidden nodes = 100 2. Learning Rate = 0.1 3. Number of learning iterations = 100 4. The initial learning weight = 0.1
Decision Jungle	1. Resampling method = Bagging 2. Number of decision DAGs = 8 3. Maximum depth of the decision DAGs = 32 4. Maximum width of the decision DAGs = 128

- Probability of detection (*Pd*) = *tp* / (*tp* + *fn*).
- Probability of false alarm (*Pf*) = *fp* / (*fp* + *tn*).
- Precision (*Pr*) = *tp* / (*tp* + *fp*).
- Accuracy (*Acc*) = (*tp* + *tn*) / (*tp* + *fp* + *fn* +*tn*).
- Processing time

Pd measures efficacy of prediction model in finding the actual vulnerability. *Pr* measures the correctness of actual vulnerabilities which are predicted in percentage. *Pf* measures the possibility of false alarm. *Acc* measures the number of correctness that the models predict the vulnerabilities correctly. Processing time indicates how long each model computes in seconds. In addition, *tp, tn, fp,* and *fn* are variable from confusion matrix as described in the TABLE III.

TABLE III. A CONFUSION MATRIX [2]

Predicted by machine learning model	**Actual**	
	Vulnerable	***Non-Vulnerable***
Vulnerable	True positive (*tp*)	False positive (*fp*)
Non-Vulnerable	False negative (*fn*)	True negative (*tn*)

D. Training and Testing

10 x 10 cross-validation methods are decided for training and testing the machine learning models. The dataset from source code is randomly divided into ten parts. Then, the machine learning model is trained on the nine parts and tested on the remaining part. This entire process is iterated for ten times [4]. The advantage of this method is that every data point gets to be in a test set exactly once, and gets to be in a training set [11].

IV. Evaluation

A. Correlation of input attributes

Linear correlation is used to prove correlation within 19 input attributes of 1,100 vulnerable SQL command datasets in Table II respectively as well as vulnerability. The results of correlation are separated by type of SQL injection. As the results, every attribute shows the dependency in positive trends in Figure 2 but not every attribute performs in the same way in Figure 3. and Figure 4. since the factors of input attribute are difference due to vulnerability type.

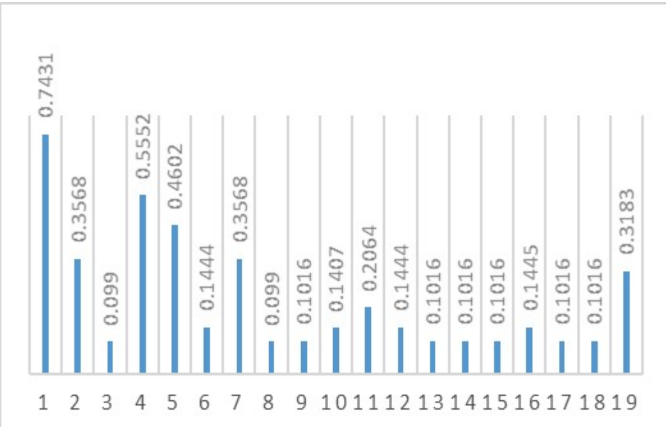

Figure 2. Correlation of input attributes in illegal/logically incorrect queries

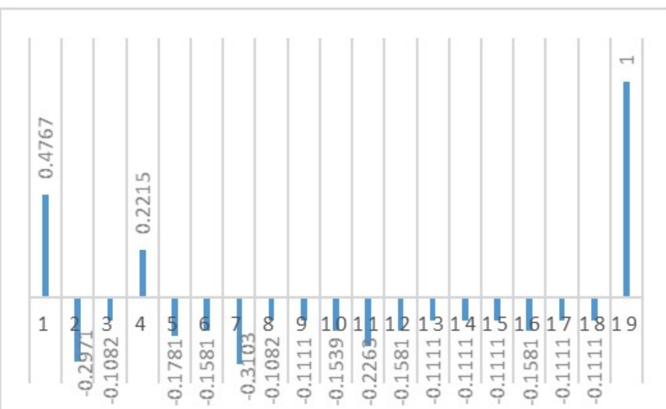

Figure 3. Correlation of input attributes in union queries

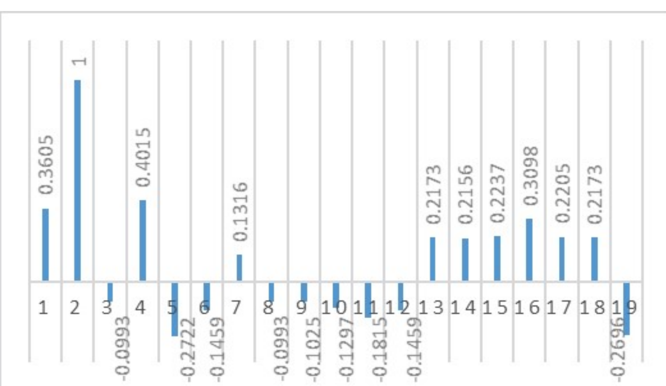

Figure 4. Correlation of input attributes in Piggy-backed queries

B. Evaluation machine learning models

Evaluation is performed in triplicates and the results are expressed as Mean±SEM (Standard Error of Mean); which is the standard deviation of the overall possible samples. The results of prediction, in terms of probability of detection (*Pd*), probability of false alarm (*Pf*), precision (*Pr*), accuracy Acc, and processing time, perform Support Vector Machine (SVM) is the best model for prediction in the vulnerability type of Illegal/Logically Incorrect queries as well as Decision Jungle is the best model for prediction in Union queries, and Piggy-backed queries. TABLE IV, V, and VI summarize the results of evaluation model in each vulnerability type as below.

TABLE IV. Evaluation model of Illegal/Logically Incorrect Queries

Machine learning model	*Pd*	*Pf*	*Pr*	*Acc*	Processing Time (Seconds)
SVM	Mean = 0.9898	Mean = 0	Mean = 1.0000	Mean = 0.9930	Mean = 2.6453
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0053
Boosted Decision Tree	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 5.2210
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0020
Artificial Neural Network	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 5.7633
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0102
Decision Jungle	Mean = 0.9865	Mean = 0	Mean = 1.0000	Mean = 0.9906	Mean = 2.6819
	SD =0	SD = 0	SD = 0	SD = 0	SD = 0.0102

TABLE V. Evaluation model of Union Queries

Machine learning model	*Pd*	*Pf*	*Pr*	*Acc*	Processing Time (Seconds)
SVM	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 2.8930
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0073
Boosted Decision Tree	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 5.2270
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0093
Artificial Neural Network	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 5.3862
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0135
Decision Jungle	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 2.4621
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0460

TABLE VI. EVALUATION MODEL OF PIGGY-BACKED QUERIES

Machine learning model	***Pd***	***Pf***	***Pr***	***Acc***	**Processing Time (Seconds)**
SVM	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean =2.6743
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0440
Boosted Decision Tree	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 2.7692
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0334
Artificial Neural Network	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 2.7658
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0032
Decision Jungle	Mean = 1.0000	Mean = 0	Mean = 1.0000	Mean = 1.0000	Mean = 2.2735
	SD = 0	SD = 0	SD = 0	SD = 0	SD = 0.0242

V. CONCLUSIONS AND FUTURE WORK

The goal is to proposed the framework of SQL injection prevention in Illegal/Logically Incorrect Queries, Union Queries, and Piggy-backed Queries on server-side scripting using compiler platform and machine learning. The machine learning part is described primarily in this paper since it is main core for SQL injection prediction part. 1,100 samples of vulnerable SQL commands are trained in four machine learning models: Support Vector Machine (SVM.), Boosted Decision Tree, Artificial Neural Network, and Decision Tree. As shown in the results of training models, Decision jungle performs as the best machine learning model which related to the processing time on average of Pd = 0.9955 SD = 0.0078, Pf = 0 SD = 0, Pr = 1.000 SD = 0, Acc = 0.9968 SD = 0.0054, and Processing Time = 2.4725 Seconds SD = 0.2044. The conduction of future experiment will be reflected to build the compiler platform on Integrated Development Environment (IDE) which should be able to validate the SQL syntax as well as support prediction and detection the SQL injection using Machine Learning application in server-side scripts within the development phase.

VI. REFERENCE

[1] National Security Agency. (2014, December). Defending Against the Exploitation of SQL Vulnerabilities to Compromise a Network. Available: https://www.iad.gov/iad/library/ia-guidance/tech-briefs/defending-against-theexploitation-of-sql-vulnerabilities-to.cfm.

[2] Lee I., Jeong S., Yeo S., and Moon J., A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 55(1–2), 2012, pp. 58-68.

[3] Zhu J., Xie J., Lipford H.R., and Chu B., Supporting secure programming in web applications through interactive static analysis. Journal of Advanced Research, **5**(4), 2014, pp. 449-462.

[4] Shar L.K. and Tan H.B.K., Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. Information and Software Technology, 55(10), 2013, pp. 1767-1780.

[5] WordPress. (2016, July). Blog Tool, Publishing Platform, and CMS. Available: http://www.wordpress.org.

[6] Drupal. (2016, July). A CMS platform for great digital experiences. Available: http://www.drupal.com

[7] Joomla. (2016, July). Joomla! The CMS Trusted by Millions for their Websites. Available: http://www.joomla.org.

[8] Simple Machines. (2016, July). Simple Machines Forum. Available: http://www.simplemachines.org.

[9] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research ,2002, pp. 321–357

[10] Microsoft Corporation. (2015, November). Machine Learning on Microsoft Azure, Available: https://azure.microsoft.com/en-us/services/machine-learning/

[11] Michael W Browne, Cross-Validation Methods, Journal of Mathematical Psychology, Volume 44, Issue 1, 2000, pp. 108-132