12-2016

Early SQL Injection Detection

Redwan A. Alward
St. Cloud State University, red_alward@yahoo.com

Early SQL Injection Detection

by

Redwan Alward

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

December, 2016

Thesis Committee:
Jim Q. Chen, Chairperson
Dennis Guster
Balasubramanian Kasi

Abstract

Computer security is a moving target that moves or increases with the growth of technology. Organizations during the 21st century have to create and/or adopt new technologies in order to stay in business and be competitive. These new technologies involve thousands of lines of code using programming languages, crossing servers, and database engines. Along with the growth of technology, organizations' IT professionals are trying to prevent any data breach to valuable data from hackers by locking all vulnerable doors that hackers might use to access a system. While IT professionals are trying to lock all vulnerable doors, hackers need only one door to hack a given system using one of the hacking methods available. One of the most used hacking methods and most security concerning is SQL Injection that hackers use to bypass a system by gaining unauthorized access to retrieve or modify valuable data such as Social Security Numbers, bank information, health records, etc. SQL Injection can be achieved through injecting SQL commands into a SQL statement via a web page. There is a number of SQL Injection methods used to gain unauthorized access into a given system; however, SQL Injection through Sign-in/Log-in process is the most used technique with 63% of all SQL injection types used [1]. Therefore, this research focuses on SQL Injection through Sign-in/Log-in process and presents a new way of alerting the system admin of any SQL Injection attempts and blocks, as well as any further access attempts by the same user (abuser).

Acknowledgments

First, I would like to express my sincere gratitude to my advisor Dr. Jim Chen for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. Besides my advisor, I would like to thank the rest of my thesis committee, Dr. Dennis Guster and Dr. Balasubramanian Kasi, for their encouragement and insightful comments.

I would also like to thank my colleagues from my internship at TelCom Construction, Inc. for their wonderful collaboration. You supported me greatly and were always willing to help me. I would particularly like to single out my supervisors at TelCom Construction Inc., Mr. Randy Linn and Mr. Lucas Bohnenkamp. Randy and Lucas, I want to thank you for your excellent cooperation and for all of the opportunities I was given to conduct my research and further my thesis at TelCom Construction Inc.

In addition, I would like to thank my classmate, Hazem Farra, for his valuable guidance and help. You definitely provided me with information I was looking for in regards to hosting my test website for this research.

I must express my very profound gratitude to my parents and to my wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Lastly, I would like to further thank my sunshine and the love of my life, my wife. You have provided me with strength when I was weak and encouraged me when I felt down. Throughout the

course of my study, you have patiently and strongly supported me and stood by my side. You handled my moodiness that is due to pressure and extra study with the utmost tolerable fashion. You turned every hard moment into a moment of encouragement and support. It is through your love, support and unconditional patience that I am able to achieve everything. For that, I thank you and hope that I make you proud.

Thank you very much, everyone!

Table of Contents

Page

Chapter Page

List of Tables

List of Figures

Figure Page

Chapter I: Introduction

Introduction

Computer security is one of the most important issues all organizations spend time and money in research and development (R&D) to produce more secure system(s) to protect their data centers. Depending on the size of the organization, large organizations host their computer security team within the organization while small size organizations outsource security to a third party that is specialized in computer security. Whether a given organization hosts a computer security team within its structure or outsources it, the main point is its data security.

SQL injection attack is one of the most-used methods to bypass a system which allows hackers to retrieve, read, modify, steal, and/or damage sensitive data. SQL injection itself has a number of ways that can be undertaken by an abuser to reach valuable data. This research discusses SQL injection through the log-in process which is the most-used method among all SQL injection methods and presents an alerting system as early as submitting the username and password from the client (front-end application) side. The main focus of this research is introducing an early alert to the system admin in case an SQL injection is attempted and blocks current and future access to the system by analyzing the user's input and the system's file logs.

Problem Statement

A successful SQL injection attack might cause significant damages to the organization that is being attacked. Some of these damages could result to the loss of an entire table of valuable information or get unlimited access to read unauthorized information since this attack enables the abuser to delete, modify, read, and insert unauthorized records, such as a social security number (SSN), medical records, date of birth, etc. While there are a number of SQL injection methods, this research addresses the root cause of sign-in/log-in processes. Log-in is the first step that anyone takes to access a given system and to be able to navigate through what the individual is authorized to do. During the log-in process, there are usually two processes that take place by the application and they are hidden from the end-user. These two processes are authentication and authorization [2]. SQL injection takes place during the authentication process by bypassing the system and sometimes getting admin rights that allow full control over the database. This issue still exists today and the cause is not having the proper SQL injection detection that is able to catch any attempts from the client side to avoid any illegal text that is entered by the user to be passed to the server side. Today's technology needs a system that is capable of handling the users' input at the application level and pass only clean input to the server for authentication and authorization processes. Not only that, but also there is a need of alerting system admin(s) in case SQL injection attempted to react faster and deploy whatever plan the organization has in place regarding this issue such as blocking the abuser's IP address, redirect the abuser to the organization's Honeypot, etc.

Nature and Significance of the Problem

There are a number of studies that have been done to develop defense methods to prevent systems from SQL injection attacks; however, it is still a continuous problem that occurs today and grows with the growth of technology. SQL injection through log-in process still exists and further significant research should be done to prevent attacks and serve as a reminder while developing applications. SQL injection through the log-in process is the most important in all SQL injection methods, as these attacks are used with 63% of all SQL injection attack methods [1]. This study presents a new way of preventing SQL injection at log-in. It is useful because it not only prevents any SQL injection attempts, but also presents an alert functionality that alerts system admins to take actions at the beginning stages of attacks.

Study Questions

Addressing a security concern is a major component during the early stages of system development and requires addressing the three most crucial elements of security: confidentiality, integrity, and availability, or CIA, during all phases of system development. Therefore, one of the most important skills that a system needs to acquire in computer security is the ability to detect, prevent, and recover.

After understanding the SQL Injection problem, Cloud Security, and Intrusion Detection and Prevention, I asked myself the question that led me to this research. Why does SQL Injection still exist when we understand the dilemma enough to prevent it, but also the ability to alert the system admins for further actions? Also, how can we retrieve users' information from the front-end application for our determination of whether to allow or block the attempted access without passing the entered information (by the user) to the server?

Definition of Terms

Table 1: Definition of terms used in this document

Term	Definition
SQL	Structured Query Language is a query language used for accessing and modifying information in a database (retrieved from: http://techterms.com).
SQL Injection	SQL injection is a type of security exploit in which the attacker adds Structured Query Language (SQL) code to a Web form input box to gain access to resources or make changes to data (retrieved from: http://searchsoftwarequality.techtarget.com).
Data center	A data center (sometimes spelled datacenter) is a centralized repository, either physical or virtual, for the storage, management, and dissemination of data and information organized around a particular body of knowledge or pertaining to a particular business (retrieved from: http://searchsoftwarequality.techtarget.com).
Data breach	A data breach is an incident in which sensitive, protected or confidential data has potentially been viewed, stolen or used by an individual unauthorized to do so. Data breaches may involve personal health information (PHI), personally identifiable information (PII), trade secrets or intellectual property (retrieved from: http://searchsoftwarequality.techtarget.com).
Authentication	Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be.
Authorization	Authorization is the process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use (such as access to which file directories, hours of access, amount of allocated storage space, and so forth) (retrieved from: http://searchsoftwarequality.techtarget.com).
Confidentiality	Confidentiality of information refers to protecting the information from disclosure to unauthorized parties.
Integrity	Integrity of information refers to protecting information from being modified by unauthorized parties.
Availability	Availability of information refers to ensuring that authorized parties are able to access the information when needed (retrieved from: http:// security.blogoverflow.com).

Summary

This chapter introduced an introduction to computer security along with the explanation of the SQL Injection problem. Then, it narrowed the explanations of specific problems this study focuses on, like SQL injection through the log-in process. The “Study Questions” section includes some questions that have been developed and raised while taking security classes as a graduate student. These questions are the main reasons for which the topic was chosen. For an easy reference, this chapter defined various terms used in the research. The next chapter covers the background and literature related to SQL Injection.

Chapter II: Background and Review of Literature

Introduction

On August of 2013, the Pew Research Centre reported that 51% of U.S. adults, and 61% of internet users, bank online [3]. Online service is a technology that has been used heavily across the world in both the government and private sectors making people’s lives easier to manage. As this technology is making our day-to-day practices easier, it is a weapon that could be used against us if technology vulnerabilities are not handled correctly. There are a number of technology vulnerabilities that hackers are targeting in order to reach sensitive data. One of these vulnerabilities is SQL injection. Protecting an organization’s data requires full understanding of SQL injection; thus, it is one of the concerns that arises within any dynamic web application development team to prevent anyone from breaching the organization’s data.

Using an online service requires a log-in process as the first step an individual has to go through in order to be able to use the service; then, navigates through their account according to their level of authorization. This chapter discusses the log-in process and its vulnerabilities.

Background Related to the Problem

- Log-in Process Overview (Alward, 2015) [2]

The log-in process is the first step an end-user goes through to access his/her account through a web-based interface known as the “authentication process.”

There are many types of authentications, or levels of authentication, used by organizations today. Some of these levels are known to be one-, two-, three-, and four-layer authentications and they differ as the following:

One-layer authentication: User enters username and password at the same time; e.g., Capital One bank (capitalone.com).

Two-layer authentication: User enters username first, then enters password; e.g., Gmail (gmail.com).

Three-layer authentication:User enters username first, then answers a security question, then enters password.

Four-layer authentication: User enters username first, then answers a security question, then a picture is displayed (a picture which was chosen by the user at the time he/she created his/her account from a list of pictures through the organization), then enters password; e.g. TCF bank (tcfbank.com).

An overview of the log-in process as Alward (2015) [1] explained in previous research is as follows:

> Log-in/Sign-in process using a web system/application is usually done through two aspects of web development, front end known as interface where the end-user (user) uses and experiences to enter his/her password and browse throughout the application. Back end, or the server side, is the other aspect of web development where it translates how the site works, updates and changes, which is everything the user can’t see, like databases [4]. Moreover, looking at these two aspects of web development from the software engineering side, front-end and back-end are distinctions which refer to the separation of concerns between a presentation layer and a data access layer respectively [1] within the OSI model (Open Systems Interconnection model). (p. 3)

- Problem Overview

SQL Injection problem arises with the use of dynamic SQL where query structure has to be formed when the program runs based on the user selection, which makes dynamic queries powerful in terms of performance with both server and client sides. We must first understand what “based on the user selection” means. Carfax.com, for example, is a commercial, web-based service that lists used cars. The site allows people to search and compare prices before they choose one to buy. Carfax.com has a wide-filtering feature that allows visitors to narrow their searches. If we take a look at the filtering query that retrieves information from the company’s database, we will find that the query structure changes every time the query is executed. This means the numbers and names of columns and conditions within the *Where* clause are changing every time the query is executed; thus, dynamic query should be used. However, dynamic SQL is the root problem to SQL injection because it leaves the server vulnerable to any attack by an abuser where he/she can add database keywords to inject the query and get unauthorized information. Let us discuss a similar example to illustrate the issue that focuses on the log-in process. Consider an application log-in process; for example, a query for a log-in process that authenticates users:

```sql
SELECT UserName FROM tblUser

WHERE UserName = '' AND UserPassword = ''
```

Let us assume that I am an authorized user with *Redwan* as my username and *Password123* as my password. The query would look like the following when it is executed:

```sql
SELECT UserName FROM tblUSer WHERE UserName = 'Redwan' AND
UserPassword = 'Password123'
```

When the above query is executed, I will be able to access the application because I have supplied the right username and password. However, what happens if I enter ‘’ *or 1=1* -- as the username and *123* as the password [5]. The query would look like the following:

```sql
SELECT UserName FROM tblUser WHERE UserName = '' or 1=1 --AND
UserPassword = '123'
```

When the above query is executed, I will be able to bypass the authentication logic, assuming there is no data validation process, because no matter what I supply as the username, 1=1 is true; it returns all records within tblUser. Also, supplying any text as the password would not make any difference within the above query because it is canceled out by using the two minus signs (--). Looking at the above query explains what SQL injection is when I am able to enter some text that can be a part of the query structure. Also, it explains how a query can be injected to bypass an application authentication process. Therefore, SQL injection is a mobile problem that has to be handled the correct way and be addressed at the early stages of application development, which is the main reason for this research.

Literature Related to the Problem

1. SQL Injection Attacks in Web Application

Gandhi and JwalantBaria (2013) [5] proposed the hash values technique and they claimed that "…this technique will be able to prevent SQL Injection Attack completely" (p. 191). This technique is where the username and password should be hashed and inserted into another two columns in the "Authorized User" table. These two columns, then, will be used during the log-in process to authenticate users after hashing the users' entries, usernames, and passwords through the client side before the dynamic SQL is executed by the server. These steps are shown in Figure 1.

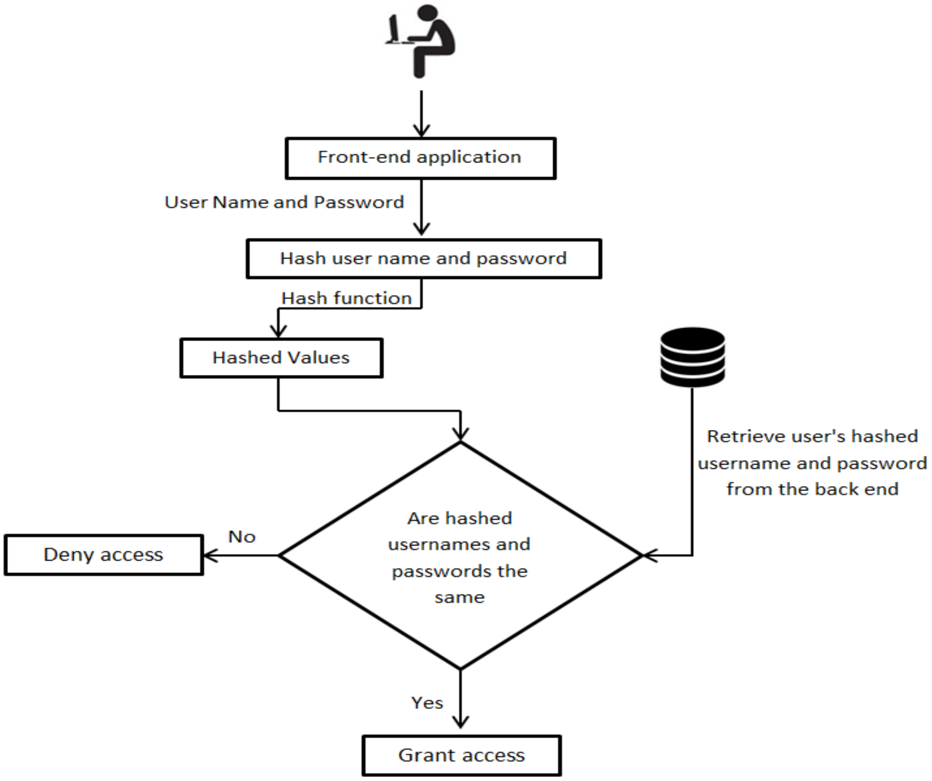

Front-end application
User Name and Password
Hash user name and password
Hash function
Hashed Values
Retrieve user's hashed username and password from the back end
Are hashed usernames and passwords the same
No
Deny access
Yes
Grant access

Figure 1: SQL injection attacks in web application–proposed architecture by Gandhi and JwalantBaria (2013) [5]

2. Evaluation of SQL Injection Detection and Prevention Techniques

Chia-Jun Lin, Chen, Hwang, and Fu-Hau Hsu (2011) proposed a novel technique [6], TransSQL, to solve the notorious SQL injection problem. Their approach indicates that TransSQL automatically duplicates SQL databases and stores the duplicated databases in an LDAP form. TransSQL is used to monitor the connections between the protected web application and the related SQL database. Then, every time an SQL request is sent to the SQL database, TransSQL creates an LDAP-equivalent request of the SQL request and sends the LDAP request to the LDAP database. This proposed technique prevents SQL injection attacks by querying both the SQL database and LDAP to ensure the equivalent contents of the SQL database and LDAP. If they are not equivalent, SQL injection attack tactics are available to use.

Literature Related to the Methodology

1. Preventing SQL Injection Attack Using Pattern Matching Algorithm

Kharche, Patil, Gohad, and Ambetkar (2015) proposed a scheme with two modules defined as Static Phase and Dynamic Phase as shown in Figure 2 [7]. In the Static Phase, they maintain a list of known anomaly patterns known as the Static Pattern List. The Static Phase, then, checks the user generated SQL queries by applying a Static Pattern Matching Algorithm searching for matching queries within the Static Pattern List. If the query matches 100% with any of the patterns from the Static Pattern List, then the query is attacked with SQL injection and access is denied. If there is no match, Phase 2 (Dynamic Phase) of the proposed method takes place. During the Dynamic Phase, a search for any form of new anomalies is triggered by calculating the anomaly score value of the user-generated SQL query using an Aho–Corasick algorithm. If the anomaly score value is more than the threshold value (50% of the query), an alarm is prompted, access is granted, and the query will be passed to the system admins for manual review by adding it to the Static Pattern List.

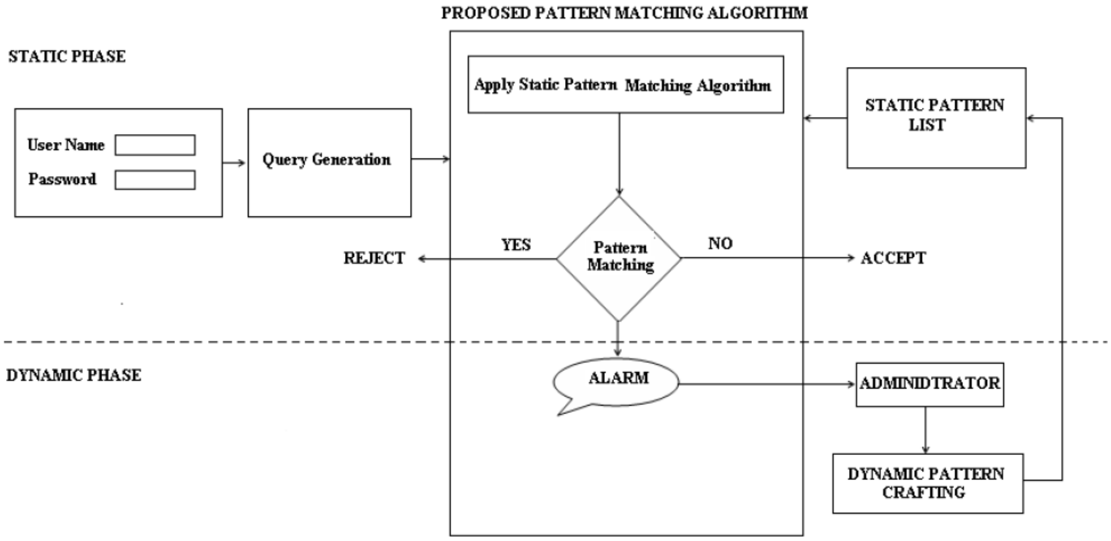


Figure 2: Preventing SQL injection attack using pattern matching algorithm proposed architecture by Kharche, Patil, Gohad, and Ambetkar (2015) [7]

2. Early Detection of SQL Injection Attacks

Shahriar, North, and Chen proposed a Shadow Query approach [8] to test the user input within the client-side before passing values to the server-side. The authors used PHP code in the implementation at the front-end and used JavaScript to code the verification function (Shadow Query). The SQL query present in the server-side script is simplified to generate Shadow Query. This is done by replacing tables, columns, logical operators, and variable values with symbolic values to prevent revealing sensitive information at the client-side. Using this approach, they deploy the system in an Apache web server (version 2.2) with MySQL server as the back-end database after creating necessary tables with data sets and tested their method by applying three types of attack inputs: tautology, union, and piggy-backed queries. The results of this approach were promising as the system detected all SQL injection attacks with a zero rate of false warning; however, there was not an alerting functionality within the system.

Summary

A database is the main system that stores, organizes, and manages a large amount of information within a single software application. Organizations use databases to increase efficiency of its business operations and this reduces overall costs. Bank of America, for example, stores all customers' information in its database, including usernames and passwords, which is used to authenticate and authorize users when they log-in using the web interface to manage their accounts. Therefore, hackers know the importance of this valuable system and databases, and are trying to find doors to break in. The log-in process is one way of breaking into a database by bypassing an unsecure system when dynamic SQL used, which is the root of the SQL injection problem. Also, this chapter has discussed four of the studies done in regards to detection SQL injection; however, none of them discussed or implemented a functionality that is capable of raising an alert as soon as an attack is attempted. The next chapter lays out the proposed approach that is going to prevent the SQL injection problem and alert the system admins as soon as an SQL injection is captured.

Chapter III: Methodology

Introduction

SQL Injection through the log-in process was introduced briefly in Chapter I and explained with examples in Chapter II. Also, Chapter II included four approaches similar to SQL injection in *Literature Related to the Problem* (p. 12) and *Literature Related to the Methodology* (p. 17). Chapter III includes the proposed *Approach* (p. 21), *Hardware and Software Environment* (p. 23)*, Work in Progress* (p. 23), and *Time Line* (p. 23).

Approach

After searching for answers to the study questions mentioned in Chapter I, this approach captures a user's IP address and text entered (username and password) by adding a top security level, which starts prior to the application and begins the authentication process. One advantage is that the information can be utilized to differentiate between blocked and unblocked users from accessing the system. Shall the user be listed on the blocked list, this first stage of security will not allow the user to pass his/her information to the second security level (authentication process). The following is a list of the logical steps the proposed system takes to determine who should be listed in the blocked users' log; also shown in Figure 3:

1. Get user's IP address, username, and password.
2. Using the user's IP address, scan the blocked users' log to determine if the IP address is listed.

3. If the IP address is listed on the blocked users' log, display "Username and/or Password Incorrect" message to the user and end the session.
4. If not, pass the username and password to the validation process; if any illegal text is found, log the user's information into the blocked users' log, display "Username and/or Password Incorrect" message, and send an alert to the system admin.
5. If no illegal text is found, start the authentication process.

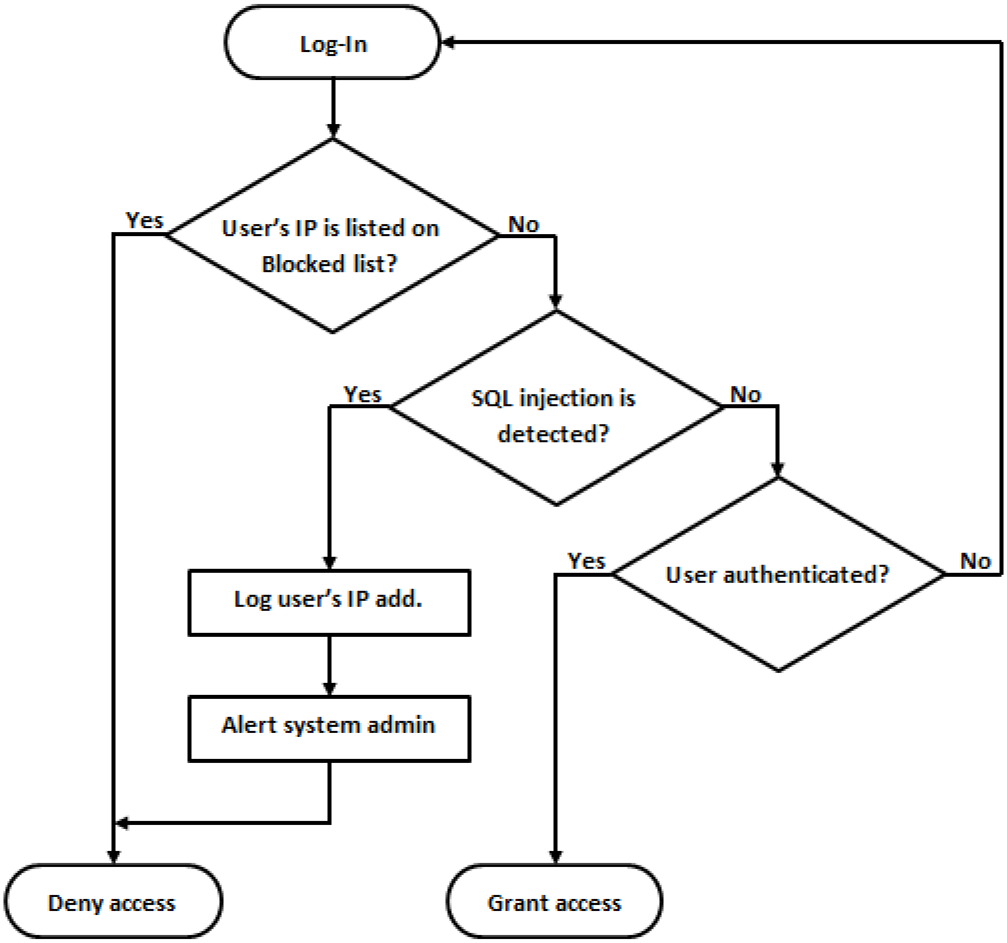


Figure 3: Proposed approach

Using the process mentioned above, system admin gets an alert as soon as any illegal text is found within the supplied text by the user, keeping in mind that the entire process is done at the client-side with the exception of determining if the user should be blocked or not. The process of determining if the user should be blocked or not is done by passing only information that was captured by the front-end application (not what the user entered) to the server-side to scan it against the system's log (blocked users' log); this way, the system makes sure that no illegal text is passed to the server [9].

Entities and processes that will be used during the implementation are:

1. Table used for authorized users.
2. Table to log all activities and blocked users.
3. Two text boxes; one for the username and the other one for the password.
4. Two buttons; one for log-in and the second to create an account.
5. Validation function.
6. Database-stored procedures.

Hardware and Software Environment

- Microsoft environment
- Microsoft Visual Studio
- VB.net
- Microsoft SQL Server
- Microsoft Office

Summary

This chapter included the layout of the logical design of the proposed approach and the tools that are used to accomplish the physical system. About 60% of this research is yet to be completed by designing the physical system for testing. I am looking forward to prove to the committee that my proposed project works with zero defects. The next chapter includes complete report after testing the system to deliver to the committee.

Chapter IV: Data Presentation and Analysis

Introduction

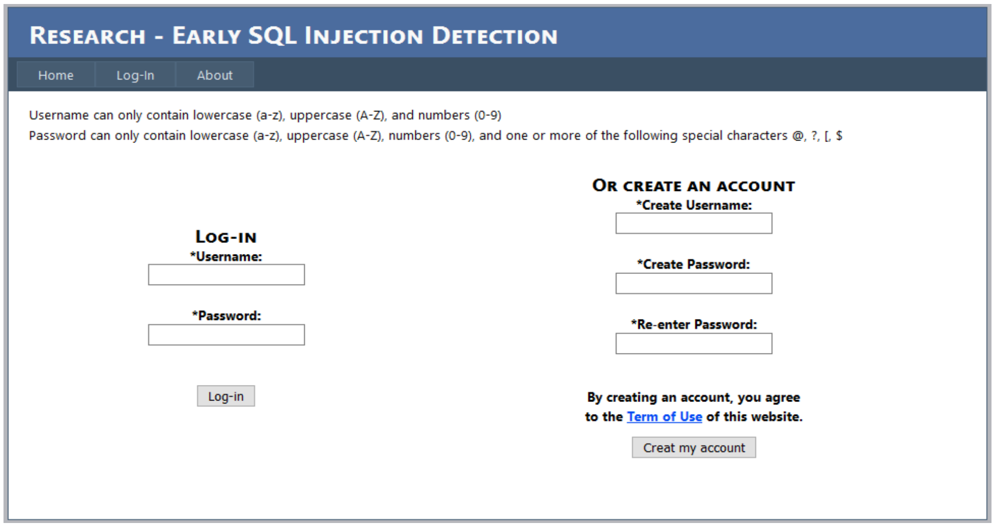
I tested the introduced method for eight days using a web-based system. The system used three major entities defined as authorized users, user blocked list, and system log. This chapter presents the environment and components used for the system, the raw data collected, and data analysis.

Data Presentation

A preliminary web-based system was designed to test the introduced methodology and to collect data using the Windows environment with the following components:

1. Front-end was developed in ASP.NET using VB (Visual Basic) programming language and hosted on a IIS 10.0 web server.
2. Back-end was developed using Microsoft SQL Server 2012.
3. Gmail account was created to send emails and text messages for the alert functionality using SMTP (Simple Mail Transfer Protocol) via VB.

The system included “Log-In” and “Create New Account” functionalities.



Username can only contain lowercase (a-z), uppercase (A-Z), and numbers (0-9)
Password can only contain lowercase (a-z), uppercase (A-Z), numbers (0-9), and one or more of the following special characters @, ?, [, $

By creating an account, you agree
to the Term of Use of this website.

Figure 4: System's interface

As stated within the introduced methodology, the first security level of the system checks for the user's IP address by passing it to the SQL server and searches for it in the system's blocked users' list. If the user's IP address is found with a True Locked Indicator (because the system admin has the ability to manage the log), an "Account is disabled" message is displayed to the user. However, if the user's IP Address is not found, or found with a False Locked Indicator, the system moves on to the second layer (validates users' input for possible SQL injection), and if the second layer is passed, moves on to the third layer (authentication).

The website was live for eight days (10/20/2016-10/27/2016) and was tested from different locations. Before looking at the data collected, the following are explanations of column headers that will be included in all or some of the next three tables:

1- **UserName** is the user’s username that he/she entered in the Username field.

2- **UserPassword** is the user’s password that he/she entered in the Password field.

3- **Date** is the date when the record was entered.

4- **Time** is the time when the record was entered.

5- **IP Address** is the user’s IP Address.

6- **LockedIndicator** is the indicator that determines if the user is allowed to access the system or not.

   a. If **True,** deny access.

   b. If **False**, grant access.

7- **Action** is the action that was taking by the user.

   a. **LogIn:** means the record was inserted when the user logged in.

   b. **Create:** means the record was inserted when the user created a new account.

8- **Successful** indicates whether the user was successful in accessing the system, whether by logging in or creating a new account.

   a. **True:** means the user was logged in or created a new account successfully.

   b. **False:** means the user was unsuccessful to log in or to create a new account.

9- **Message:** displays the message that was sent to the user (front-end application).

Data Collected

1- Users Data (8 records):

The following data was collected when the user successfully created an account.

Table 2: All users' data from created accounts

ID	UserName	UserPassword	Date	Time	IPAddress
3	redtest1	redtwst1	10/20/2016	6:25:56 PM	199.17.22.66
4	redtest2	redtest2	10/20/2016	6:38:46 PM	199.17.55.161
5	jchen	apple999	10/20/2016	11:26:17 PM	68.119.24.182
6	redtest3	redtest3	10/21/2016	12:54:41 PM	63.230.81.241
7	red-test	redtest	10/21/2016	7:48:51 PM	192.168.1.1
8	redtest4	redtest4	10/21/2016	7:53:09 PM	192.168.1.1
9	Redtest6	redtest6	10/23/2016	1:03:33 PM	8.42.164.20
10	redtest5	redtest5	10/24/2016	8:01:54 AM	24.52.25.146

2- System's Users Blocked List (30 records):

The following data was collected when the system detected SQL injection attempts. This data serves the system admin as a log to look at when he/she gets the alert to view and analyze for decisions. Also, here is where the system admin is able to turn off blocking a particular user in cases, like false negatives (incorrectly rejected), occurred.

Table 3: BlockedUser log

ID	UserName	Password	Date	Time	Locked Indicator	UserIP
1	' or 1=1 --	redtest2	10/20/2016	6:46:05 PM	TRUE	199.17.55.161
2	' or 1=1 --	test	10/20/2016	7:28:47 PM	TRUE	199.17.22.67
3	jchen	' or 1=1	10/21/2016	10:16:56 AM	TRUE	199.17.38.71
4	redtest3	'or1=1--	10/21/2016	12:57:37 PM	TRUE	63.230.81.241
5	Redtest3	'or1=1 --	10/21/2016	1:18:50 PM	TRUE	66.191.131.210
6	' or 1=1 --	hjdj	10/21/2016	1:31:14 PM	TRUE	64.83.197.92
7	'Or 1=1 --	test	10/22/2016	8:26:08 PM	TRUE	174.219.13.149
8	Redtest6	'or1=1--	10/23/2016	1:04:26 PM	TRUE	8.42.164.20
9	'Or 1=1 --	hhhhhh	10/23/2016	1:13:54 PM	TRUE	207.238.29.50
10	Test	'or1=1--	10/23/2016	2:12:53 PM	TRUE	63.229.207.136
11	Test' or 1=1 --	gvbvvb	10/23/2016	2:34:45 PM	TRUE	65.103.10.49
12	Test	'or1=1--	10/23/2016	2:37:02 PM	TRUE	63.231.253.15
13	Test	' or 3=3--	10/23/2016	2:39:10 PM	TRUE	75.146.149.45
14	Test	'or1=1--	10/23/2016	5:31:46 PM	TRUE	209.181.233.89
15	redtest5	'or 9=9 --	10/24/2016	8:02:30 AM	TRUE	24.52.25.146
16	Test	'or1=1--	10/25/2016	7:50:38 PM	TRUE	208.54.80.158
17	' select * from tbluser--	ghjhg	10/25/2016	7:53:08 PM	TRUE	12.23.136.106
18	Test	'or1=1--	10/25/2016	8:05:01 PM	TRUE	63.230.83.177
19	Test	'or1=1--	10/25/2016	8:21:55 PM	TRUE	204.96.25.180
20	Test ' or 1=1 --	test	10/25/2016	8:26:52 PM	TRUE	63.230.81.14
21	' or 1	hgfhg	10/27/2016	6:27:14 PM	TRUE	199.17.55.169
22	' or 1=1 --	test	10/27/2016	6:33:36 PM	TRUE	199.17.22.85
23	test ' or 5=5 --	test	10/27/2016	6:35:01 PM	TRUE	199.17.22.86
24	test	'or1=1--	10/27/2016	6:37:14 PM	TRUE	199.17.22.83
25	test	' delete table tbltest;--	10/27/2016	6:38:41 PM	TRUE	199.17.22.82
26	test	'or 1=1 --	10/27/2016	6:41:01 PM	TRUE	199.17.22.92
27	test	'or1=1--	10/27/2016	6:46:19 PM	TRUE	199.17.22.106
28	' select * from tblUser --	gfdsf	10/27/2016	6:49:48 PM	TRUE	199.17.29.3
29	test	'or1=1--	10/27/2016	6:52:46 PM	TRUE	199.17.29.96
30	test	'or 1=1--	10/27/2016	6:53:19 PM	TRUE	199.17.29.65

3- System's Log (142 records):

The following data presents a system log file that contains all events that were performed by users; the following data was collected at any time as a user:

a. Logs in; whether he/she was successfully logged in or not based on the following facts:
    i. Correct username and password.
    ii. Incorrect username and password.
    iii. User is blocked.
b. Creates new account; whether he/she was successful in creating new account or not based on the following facts:
    i. New account was created.
    ii. Username already used.
    iii. User is blocked.

Table 4: System's log

ID	UserName	Password	IP Address	Action	Successful	Date	Time	Message
9	Test	test	208.54.83.129	LogIn	TRUE	10/20/2016	10:50:38 AM	Login Succeeded.
10	Test	test	174.219.0.230	LogIn	TRUE	10/20/2016	12:15:49 PM	Login Succeeded.
11	thesis	test	199.17.55.161	LogIn	TRUE	10/20/2016	12:53:22 PM	Login Succeeded.
12	test	test	172.98.67.90	LogIn	TRUE	10/20/2016	2:28:28 PM	Login Succeeded.
13	test	test	199.17.22.66	LogIn	TRUE	10/20/2016	6:25:18 PM	Login Succeeded.
14	redtest1	redtwst1	199.17.22.66	Create	TRUE	10/20/2016	6:25:56 PM	Account created.
15	redtwst1	redtest1	199.17.22.66	LogIn	FALSE	10/20/2016	6:26:08 PM	Invalid Username or password.
16	redtest1	redtest1	199.17.22.66	LogIn	FALSE	10/20/2016	6:26:18 PM	Invalid Username or password.
17	redtest1	redtest1	199.17.22.66	LogIn	FALSE	10/20/2016	6:26:29	Invalid

ID	UserName	Password	IP Address	Action	Successful	Date	Time	Message
							PM	Username or password.
18	test	test	199.17.22.66	LogIn	TRUE	10/20/2016	6:26:41 PM	Login Succeeded.
19	redtest1	redtest1	199.17.22.66	LogIn	FALSE	10/20/2016	6:26:51 PM	Invalid Username or password.
20	redtest1	redtwst1	199.17.22.66	LogIn	TRUE	10/20/2016	6:37:09 PM	Login Succeeded.
21	redtest2	redtest2	199.17.55.161	Create	TRUE	10/20/2016	6:38:46 PM	Account created.
22	redtest2	redtest2	199.17.55.161	LogIn	TRUE	10/20/2016	6:38:55 PM	Login Succeeded.
23	' or 1=1 --	redtest2	199.17.55.161	LogIn	FALSE	10/20/2016	6:46:05 PM	Account is disabled.
24	test	test	199.17.55.161	LogIn	FALSE	10/20/2016	6:46:39 PM	Account is disabled.
25	newacco	newpass	199.17.55.161	Create	FALSE	10/20/2016	6:46:55 PM	Account is disabled.
26	test	test	199.17.55.161	LogIn	FALSE	10/20/2016	6:47:01 PM	Account is disabled.
27	test	test	199.17.22.66	LogIn	TRUE	10/20/2016	6:48:04 PM	Login Succeeded.
28	Test	test	199.17.55.161	LogIn	FALSE	10/20/2016	6:48:42 PM	Account is disabled.
29	Test	test	199.17.55.161	LogIn	FALSE	10/20/2016	6:49:56 PM	Account is disabled.
30	test	test	199.17.22.67	LogIn	TRUE	10/20/2016	7:28:31 PM	Login Succeeded.
31	' or 1=1 --	test	199.17.22.67	LogIn	FALSE	10/20/2016	7:28:47 PM	Account is disabled.
32	test	test	199.17.22.67	LogIn	FALSE	10/20/2016	7:28:55 PM	Account is disabled.
33	jchen	apple999	68.119.24.182	Create	TRUE	10/20/2016	11:26:17 PM	Account created.
34	jchen	apple999	68.119.24.182	LogIn	TRUE	10/20/2016	11:26:43 PM	Login Succeeded.
35	jchen	apple999	199.17.38.71	LogIn	TRUE	10/21/2016	10:14:11 AM	Login Succeeded.
36	jchen	' or 1=1	199.17.38.71	LogIn	FALSE	10/21/2016	10:16:56 AM	Account is disabled.
37	redtest3	redtest3	63.230.81.241	Create	TRUE	10/21/2016	12:54:41 PM	Account created.
38	redtest3	redtest3	63.230.81.241	LogIn	TRUE	10/21/2016	12:57:27 PM	Login Succeeded.
39	redtest3	'or1=1--	63.230.81.241	LogIn	FALSE	10/21/2016	12:57:37 PM	Account is disabled.
40	Redtest3	redtest3	174.219.132.31	LogIn	TRUE	10/21/2016	1:05:21 PM	Login Succeeded.
41	Redtest3	redtest3	66.191.131.210	LogIn	TRUE	10/21/2016	1:18:12 PM	Login Succeeded.
42	Redtest3	'or1=1 --	66.191.131.210	LogIn	FALSE	10/21/2016	1:18:50 PM	Account is disabled.
43	Redtest3	redtest3	66.191.131.210	LogIn	FALSE	10/21/2016	1:19:02 PM	Account is disabled.
44	Redtest3	redtest3	64.83.197.92	LogIn	TRUE	10/21/2016	1:30:46 PM	Login Succeeded.
45	' or 1=1 --	hjdj	64.83.197.92	LogIn	FALSE	10/21/2016	1:31:14 PM	Account is disabled.
46	Redtest3	redtest3	64.83.197.92	LogIn	FALSE	10/21/2016	1:31:33	Account is

ID	UserName	Password	IP Address	Action	Successful	Date	Time	Message
							PM	disabled.
47	test	test	192.168.1.1	LogIn	TRUE	10/21/2016	7:07:26 PM	Login Succeeded.
48	red-test	redtest	192.168.1.1	Create	TRUE	10/21/2016	7:48:51 PM	Account created.
49	test	test	192.168.1.1	LogIn	TRUE	10/21/2016	7:52:41 PM	Login Succeeded.
50	redtest4	redtest4	192.168.1.1	Create	TRUE	10/21/2016	7:53:09 PM	Account created.
51	redtest4	test	192.168.1.1	Create	FALSE	10/21/2016	7:53:49 PM	Username already used.
52	Test	test	199.17.55.161	LogIn	FALSE	10/22/2016	4:32:27 PM	Account is disabled.
53	Redtest5	redtest5	199.17.55.161	Create	FALSE	10/22/2016	4:33:30 PM	Account is disabled.
54	Test	test	174.219.13.149	LogIn	TRUE	10/22/2016	8:19:26 PM	Login Succeeded.
55	Test	test	174.219.13.149	LogIn	TRUE	10/22/2016	8:19:39 PM	Login Succeeded.
56	'Or 1=1 --	test	174.219.13.149	LogIn	FALSE	10/22/2016	8:26:08 PM	Account is disabled.
57	Test	test	174.219.13.149	LogIn	FALSE	10/22/2016	8:26:21 PM	Account is disabled.
58	Test	test	8.42.164.20	LogIn	TRUE	10/23/2016	1:02:48 PM	Login Succeeded.
59	Redtest6	redtest6	8.42.164.20	Create	TRUE	10/23/2016	1:03:33 PM	Account created.
60	Redtest6	dgtgh	8.42.164.20	LogIn	FALSE	10/23/2016	1:03:49 PM	Invalid Username or password.
61	Redtest6	redtest6	8.42.164.20	LogIn	TRUE	10/23/2016	1:03:57 PM	Login Succeeded.
62	Redtest6	'or1=1--	8.42.164.20	LogIn	FALSE	10/23/2016	1:04:26 PM	Account is disabled.
63	Redtest6	redtedt6	8.42.164.20	LogIn	FALSE	10/23/2016	1:04:37 PM	Account is disabled.
64	Test	test	8.42.164.20	LogIn	FALSE	10/23/2016	1:04:59 PM	Account is disabled.
65	Test	test	207.238.29.50	LogIn	TRUE	10/23/2016	1:13:33 PM	Login Succeeded.
66	'Or 1=1 --	hhhhhh	207.238.29.50	LogIn	FALSE	10/23/2016	1:13:54 PM	Account is disabled.
67	Test	test	207.238.29.50	LogIn	FALSE	10/23/2016	1:14:04 PM	Account is disabled.
68	Test	test	174.219.15.144	LogIn	TRUE	10/23/2016	1:39:40 PM	Login Succeeded.
69	Test	test	8.42.164.20	LogIn	FALSE	10/23/2016	1:44:42 PM	Account is disabled.
70	Test	test	63.229.207.136	LogIn	TRUE	10/23/2016	2:12:28 PM	Login Succeeded.
71	Test	'or1=1--	63.229.207.136	LogIn	FALSE	10/23/2016	2:12:53 PM	Account is disabled.
72	Test	test	63.229.207.136	LogIn	FALSE	10/23/2016	2:13:02 PM	Account is disabled.
73	Test	test	65.103.10.49	LogIn	TRUE	10/23/2016	2:31:34 PM	Login Succeeded.
74	Test	test	65.103.10.49	LogIn	TRUE	10/23/2016	2:34:15 PM	Login Succeeded.

ID	UserName	Password	IP Address	Action	Successful	Date	Time	Message
75	Test	hfth	65.103.10.49	LogIn	FALSE	10/23/2016	2:34:19 PM	Invalid Username or password.
76	Test' or 1=1 --	gvbvvb	65.103.10.49	LogIn	FALSE	10/23/2016	2:34:45 PM	Account is disabled.
77	Test	test	65.103.10.49	LogIn	FALSE	10/23/2016	2:35:04 PM	Account is disabled.
78	Test	test	63.231.253.15	LogIn	TRUE	10/23/2016	2:36:42 PM	Login Succeeded.
79	Test	'or1=1--	63.231.253.15	LogIn	FALSE	10/23/2016	2:37:02 PM	Account is disabled.
80	Test	test	63.231.253.15	LogIn	FALSE	10/23/2016	2:37:08 PM	Account is disabled.
81	Test	test	75.146.149.45	LogIn	TRUE	10/23/2016	2:38:45 PM	Login Succeeded.
82	Test	' or 3=3--	75.146.149.45	LogIn	FALSE	10/23/2016	2:39:10 PM	Account is disabled.
83	Trst	test	209.181.233.89	LogIn	FALSE	10/23/2016	5:31:17 PM	Invalid Username or password.
84	Test	test	209.181.233.89	LogIn	TRUE	10/23/2016	5:31:32 PM	Login Succeeded.
85	Test	'or1=1--	209.181.233.89	LogIn	FALSE	10/23/2016	5:31:46 PM	Account is disabled.
86	Test	test	209.181.233.89	LogIn	FALSE	10/23/2016	5:36:39 PM	Account is disabled.
87	test	test	192.168.1.1	LogIn	TRUE	10/23/2016	10:07:45 PM	Login Succeeded.
88	test	test	24.52.25.146	LogIn	TRUE	10/24/2016	8:01:25 AM	Login Succeeded.
89	redtest5	redtest5	24.52.25.146	Create	TRUE	10/24/2016	8:01:54 AM	Account created.
90	reddest5	redtest5	24.52.25.146	LogIn	FALSE	10/24/2016	8:02:04 AM	Invalid Username or password.
91	redtest5	redtest5	24.52.25.146	LogIn	TRUE	10/24/2016	8:02:15 AM	Login Succeeded.
92	redtest5	'or 9=9 --	24.52.25.146	LogIn	FALSE	10/24/2016	8:02:30 AM	Account is disabled.
93	redtest5	redtest5	24.52.25.146	LogIn	FALSE	10/24/2016	8:02:49 AM	Account is disabled.
94	test6	test6	24.52.25.146	LogIn	FALSE	10/25/2016	12:06:18 PM	Account is disabled.
95	Test	test	208.54.80.158	LogIn	TRUE	10/25/2016	7:50:04 PM	Login Succeeded.
96	Test	'or1=1--	208.54.80.158	LogIn	FALSE	10/25/2016	7:50:38 PM	Account is disabled.
97	Test	test	208.54.80.158	LogIn	FALSE	10/25/2016	7:50:44 PM	Account is disabled.
98	Test	test	12.23.136.106	LogIn	TRUE	10/25/2016	7:52:08 PM	Login Succeeded.
99	' select * from tbluser--	ghjhg	12.23.136.106	LogIn	FALSE	10/25/2016	7:53:08 PM	Account is disabled.
100	test	test	12.23.136.106	LogIn	FALSE	10/25/2016	7:53:22 PM	Account is disabled.
101	Test	test	63.230.83.177	LogIn	TRUE	10/25/2016	8:03:54 PM	Login Succeeded.
102	Test	'or1=1--	63.230.83.177	LogIn	FALSE	10/25/2016	8:05:01 PM	Account is disabled.

ID	UserName	Password	IP Address	Action	Successful	Date	Time	Message
103	Test	test	63.230.83.177	LogIn	FALSE	10/25/2016	8:05:11 PM	Account is disabled.
104	Test	test	12.23.136.106	LogIn	FALSE	10/25/2016	8:06:11 PM	Account is disabled.
105	Test	test	204.96.25.180	LogIn	TRUE	10/25/2016	8:21:40 PM	Login Succeeded.
106	Test	'or1=1--	204.96.25.180	LogIn	FALSE	10/25/2016	8:21:55 PM	Account is disabled.
107	Test	test	204.96.25.180	LogIn	FALSE	10/25/2016	8:22:03 PM	Account is disabled.
108	Test	test	63.230.81.14	LogIn	TRUE	10/25/2016	8:26:03 PM	Login Succeeded.
109	Test ' or 1=1 --	test	63.230.81.14	LogIn	FALSE	10/25/2016	8:26:52 PM	Account is disabled.
110	Test	test	63.230.81.14	LogIn	FALSE	10/25/2016	8:27:05 PM	Account is disabled.
111	Test	test	12.23.136.106	LogIn	FALSE	10/25/2016	8:29:51 PM	Account is disabled.
112	test	test	199.17.55.169	LogIn	TRUE	10/27/2016	6:26:59 PM	Login Succeeded.
113	' or 1	hgfhg	199.17.55.169	LogIn	FALSE	10/27/2016	6:27:14 PM	Account is disabled.
114	test	test	199.17.55.169	LogIn	FALSE	10/27/2016	6:27:39 PM	Account is disabled.
115	test1	test1	199.17.55.169	LogIn	FALSE	10/27/2016	6:27:48 PM	Account is disabled.
116	test222	test222	199.17.55.169	Create	FALSE	10/27/2016	6:28:06 PM	Account is disabled.
117	test222	test222	199.17.55.169	Create	FALSE	10/27/2016	6:29:45 PM	Account is disabled.
118	test	test	199.17.55.169	LogIn	FALSE	10/27/2016	6:29:57 PM	Account is disabled.
119	test	test	199.17.22.85	LogIn	TRUE	10/27/2016	6:33:23 PM	Login Succeeded.
120	' or 1=1 --	test	199.17.22.85	LogIn	FALSE	10/27/2016	6:33:36 PM	Account is disabled.
121	test	test	199.17.22.85	LogIn	FALSE	10/27/2016	6:33:44 PM	Account is disabled.
122	test	test	199.17.22.86	LogIn	TRUE	10/27/2016	6:34:47 PM	Login Succeeded.
123	test ' or 5=5 --	test	199.17.22.86	LogIn	FALSE	10/27/2016	6:35:01 PM	Account is disabled.
124	test	test	199.17.22.86	LogIn	FALSE	10/27/2016	6:35:08 PM	Account is disabled.
125	test	test	199.17.22.83	LogIn	TRUE	10/27/2016	6:37:03 PM	Login Succeeded.
126	test	'or1=1--	199.17.22.83	LogIn	FALSE	10/27/2016	6:37:14 PM	Account is disabled.
127	test	test	199.17.22.83	LogIn	FALSE	10/27/2016	6:37:18 PM	Account is disabled.
128	test	test	199.17.22.82	LogIn	TRUE	10/27/2016	6:38:12 PM	Login Succeeded.
129	test	' delete table tbltest;--	199.17.22.82	LogIn	FALSE	10/27/2016	6:38:41 PM	Account is disabled.
130	test	test	199.17.22.82	LogIn	FALSE	10/27/2016	6:38:48 PM	Account is disabled.
131	test	test	199.17.22.92	LogIn	TRUE	10/27/2016	6:40:50 PM	Login Succeeded.
132	test	'or 1=1 --	199.17.22.92	LogIn	FALSE	10/27/2016	6:41:01	Account is

ID	UserName	Password	IP Address	Action	Successful	Date	Time	Message
							PM	disabled.
133	test	test	199.17.22.92	LogIn	FALSE	10/27/2016	6:41:07 PM	Account is disabled.
134	test	test	199.17.22.93	LogIn	TRUE	10/27/2016	6:41:57 PM	Login Succeeded.
135	test	test	199.17.22.106	LogIn	TRUE	10/27/2016	6:46:07 PM	Login Succeeded.
136	test	'or1=1--	199.17.22.106	LogIn	FALSE	10/27/2016	6:46:19 PM	Account is disabled.
137	test	test	199.17.22.106	LogIn	FALSE	10/27/2016	6:46:25 PM	Account is disabled.
138	test	testtskj	199.17.29.3	LogIn	FALSE	10/27/2016	6:49:22 PM	Invalid Username or password.
139	test	test	199.17.29.3	LogIn	TRUE	10/27/2016	6:49:26 PM	Login Succeeded.
140	' select * from tblUser --	gfdsf	199.17.29.3	LogIn	FALSE	10/27/2016	6:49:48 PM	Account is disabled.
141	test	test	199.17.29.3	LogIn	FALSE	10/27/2016	6:49:56 PM	Account is disabled.
142	test	test	199.17.29.3	LogIn	FALSE	10/27/2016	6:50:01 PM	Account is disabled.
143	test	test	199.17.29.96	LogIn	TRUE	10/27/2016	6:52:34 PM	Login Succeeded.
144	test	'or1=1--	199.17.29.96	LogIn	FALSE	10/27/2016	6:52:46 PM	Account is disabled.
145	test	test	199.17.29.96	LogIn	FALSE	10/27/2016	6:52:49 PM	Account is disabled.
146	test	test	199.17.29.65	LogIn	TRUE	10/27/2016	6:53:10 PM	Login Succeeded.
147	test	'or 1=1--	199.17.29.65	LogIn	FALSE	10/27/2016	6:53:19 PM	Account is disabled.
148	test	test	199.17.29.65	LogIn	FALSE	10/27/2016	6:53:23 PM	Account is disabled.
149	test	test	199.17.29.3	LogIn	FALSE	10/27/2016	6:55:02 PM	Account is disabled.
150	test	test	192.168.1.1	LogIn	TRUE	10/27/2016	8:02:21 PM	Login Succeeded.

4- Alerting system admin:

   a. 30 text messages were received by cell phone.

   b. 30 emails were received by email.

A custom message was sent to the system admin in the event that SQL injection was detected.

Message title: “SQL Injection Detected.”

Message body: “Abuser's information has been logged.”

Data Analysis

Microsoft Excel 2007 and writing queries in SQL Server 2012 were the tools used to analyze the collected data between 10/20/2016 and 10/27/2016. Based on the analysis:

1- 8 accounts were created from 7 unique IP addresses.

2- 30 unique IP addresses were blocked.

   a. 4 of the 30 IP addresses were listed among the 7 unique IP addresses.

   b. 26 of the 30 IP addresses were different than the 7 unique IP addresses.

3- 30 alerts were sent by the system.

   a. 30 alerts were received by the system admin.

4- 58 of the 142 log records had successfully accessed the system.

   a. 50 of the 58 successfully logged into the system.

   b. 8 of the 58 successfully created new accounts.

5- 84 of the 142 log records were unsuccessful to access the system.

   a. 9 of the 84 had invalid username or password.

   b. 1 of the 84 tried to create an account with a username that already had been used.

   c. 74 of the 84 were completely blocked.

      i. The 74 records had 30 unique IP addresses.

         1. 1 IP address was blocked 8 times.

2. 1 IP address was blocked 6 times.
3. 3 IP addresses were blocked 4 times each.
4. 1 IP address was blocked 3 times.
5. 21 IP addresses were blocked 2 times each.
6. 3 IP addresses were blocked 1 time each.

6- 0 IP addresses were granted access with SQL Injection attempt.

a. All IP addresses were logged and blocked at the time SQL injection occurred and denied all future access by the same IP addresses.

Summary

In this chapter, the environment and components used to design the system for testing, raw data collection, and data analysis were presented. This chapter included all of the raw records collected in the three entities. For better understanding, this chapter also included an explanation of the raw data and the analysis, as there were no IP addresses that were granted access with the SQL injection attempts. The next chapter discusses the promising results, conclusion, and recommendations for future study.

Chapter V: Results, Conclusion, and Recommendations

Introduction

The physical design reacted as expected during the planning of the logical design. This chapter includes the results the system facilitated and explains how the first security layer played a crucial role in securing the system from abusers. After discussing the results, this chapter also includes recommendations based on the results. Also, this chapter includes future studies that can be done to strengthen the method.

Results

Computer security is one of the major concerns in the 21st century and it increases with the growth of technology. This research focused on one of the most used techniques by hackers—SQL injection. Additionally, this study introduced a method of identifying users' IP addresses as a tool to block abusers, utilizing security layers, and alerting system admins as soon as SQL injection was detected.

The results of the physical design of the introduced methodology are promising, as it blocked all abusers from accessing the system. The system blocked access for all 30 IP addresses that were logged when they first tried to hack the system using SQL injection technique. None of 30 IP addresses were able to access the system again, as the system's first security layer was able to decipher abusers from legitimate users through their IP addresses. Therefore, the rate of allowing an abuser to access the system was 0.

Abusers were blocked from accessing the system by analyzing their IP addresses within the client-side; therefore, no other information was passed to the server-side (database), so that server resources are not wasted. According to the log, one IP address was blocked 21 times from accessing the system, which was the highest on record. The following table shows how many times every IP address tried to log-in or create an account during the testing period:

Table columns definition:

IPAddress: User's IP address.

Successful: **1 = True**, if user successfully logged in or created an account.

0 = False, if user was blocked from accessing the system.

Number of Activity: the number of how many times a user tried to access the system.

Table 5: Number of times every IP address tried to log-in or create an account

	IPAddress	Successful	Number Of Activity
1	12.23.136.106	1	1
2	12.23.136.106	0	4
3	172.98.67.90	1	1
4	174.219.0.230	1	1
5	174.219.13.149	1	2
6	174.219.13.149	0	2
7	174.219.132.31	1	1
8	174.219.15.144	1	1
9	192.168.1.1	1	6
10	192.168.1.1	0	1
11	199.17.22.106	1	1
12	199.17.22.106	0	2
13	199.17.22.66	1	5
14	199.17.22.66	0	4
15	199.17.22.67	1	1
16	199.17.22.67	0	2
17	199.17.22.82	1	1
18	199.17.22.82	0	2
19	199.17.22.83	1	1
20	199.17.22.83	0	2
21	199.17.22.85	1	1
22	199.17.22.85	0	2
23	199.17.22.86	1	1
24	199.17.22.86	0	2
25	199.17.22.92	1	1
26	199.17.22.92	0	2
27	199.17.22.93	1	1
28	199.17.29.3	1	1
29	199.17.29.3	0	5
30	199.17.29.65	1	1
31	199.17.29.65	0	2
32	199.17.29.96	1	1
33	199.17.29.96	0	2
34	199.17.38.71	1	1
35	199.17.38.71	0	1
36	199.17.55.161	1	3
37	199.17.55.161	0	8
38	199.17.55.169	1	1
39	199.17.55.169	0	6
40	204.96.25.180	1	1
41	204.96.25.180	0	2
42	207.238.29.50	1	1
43	207.238.29.50	0	2
44	208.54.80.158	1	1
45	208.54.80.158	0	2
46	208.54.83.129	1	1
47	209.181.233.89	1	1
48	209.181.233.89	0	3
49	24.52.25.146	1	3
50	24.52.25.146	0	4
51	63.229.207.136	1	1
52	63.229.207.136	0	2
53	63.230.81.14	1	1
54	63.230.81.14	0	2
55	63.230.81.241	1	2
56	63.230.81.241	0	1
57	63.230.83.177	1	1
58	63.230.83.177	0	2
59	63.231.253.15	1	1
60	63.231.253.15	0	2
61	64.83.197.92	1	1
62	64.83.197.92	0	2
63	65.103.10.49	1	2
64	65.103.10.49	0	3
65	66.191.131.210	1	1
66	66.191.131.210	0	2
67	68.119.24.182	1	2
68	75.146.149.45	1	1
69	75.146.149.45	0	1
70	8.42.164.20	1	3
71	8.42.164.20	0	5

Conclusion

This research presented a new method to keep hackers from reaching a given system. This method was thought of after studying some key points of computer security along with understanding the importance and role of IP addresses in communication over the internet. Since any three-way handshake has to include source and destination IP addresses, why IP addresses are not used to identify users' behavior as a security layer when SQL injection is attempted is a mystery.

IP addresses within the .NET Framework can be accessed by using the HttpContext Class. This study uses IP addresses along with examining the user's input to identify if the system should grant or deny access. As results of this study were gathered, all IP addresses were blocked from accessing the system where SQL injection attacks were presented. Not only that, but there was zero access granted to any IP address which was listed in the system's log, whether SQL injection attacks were detected or not. This step was taken to consider security as a top priority because it is believed access should be denied to anyone who is using the same IP address, and this was presented in the analysis when there were 74 access attempts from 30 blocked IP addresses. In other words, if Bob and Alice are in the same LAN and Bob tries SQL injection to attack the system, Bob and Alice will be blocked from accessing the system since they share the same IP address (both are in the same LAN).

The analysis includes 10 records out of the 74 that tried to access the account with usernames or passwords that are not found in the authorized users' table (incorrect username or password). Yet, 6 of the 10 logged records indicated the system was attacked from these addresses that were not found in the authorized users' table, which concludes that someone can use SQL injection even if he/she is not the legitimate user. Thus, IP addresses are tools this study used to prevent any illegal users from accessing the system.

Future Study

Research in the computer security sector will have to be conducted to strengthen the introduced method. The system's Users Blocked List was built dynamically, as it validated each text entered by users. This study used public IP addresses in determining abusers from legitimate users to allow only the latter to access the system. However, it blocks users based on the network (LAN) level, not the user level since public IP addresses can be shared by more than one user. This method can be further developed to block users on the user level by device fingerprinting.

Using device fingerprint (also known as machine fingerprint or browser fingerprint) allows the system to identify individuals or devices for security purposes. Device fingerprint collects information about the device when it is connected to the internet. There are a number of methods used during device fingerprinting such as browser configurations, OSI, operating systems, wireless settings, etc. Device fingerprint is already tested and used, especially by banks (online banking).

As this study used IP addresses as security functionality to block abusers from accessing the system when SQL injection was detected, IP address methods can be applied to combat any other hacking methods, as IP addresses are available in all internet communications. Therefore, this is a tool that is available and security professionals should make a use of it.

References

[1] Imperva, “Web application attack report #5,” October 1, 2014, **Error! Hyperlink reference not valid.** [December 10, 2015].

[2] R. Alward, “SQL injection,” St. Cloud State University, St. Cloud, MN, 2015.

[3] S. Fox, “51% of U.S. adults bank online,” 2013, http://www.pewinternet.org/2013/08/07/51-of-u-s-adults-bank-online/ [February 22, 2016].

[4] C. Kavourgias, “What's the difference between the front-end and back-end?,” January 28, 2015, http://blog.digitaltutors.com/whats-difference-front-end-back-end/ [December 10, 2015].

[5] M. Gandhi and JwalantBaria, “SQL injection attacks in web application,” *International Journal of Soft Computing and Engineering (IJSCE)*, ISSN: 2231-2307, vol. 2, iss. 6, January 2013.

[6] K. Lin, S. Chen, Y. Hwang, and H. Fu-Hau Hsu, “TransSQL: A Translation and validation-based solution for SQL-injection attacks,” *First International Conference on Robot, IEEE*, 2011.

[7] S. Kharche, J. Patil, K. Gohad, and B. Ambetkar, “Preventing SQL injection attack using pattern matching algorithm,” Maharashtra Institute of Technology, Pune, https://arxiv.org/ftp/arxiv/papers/1504/1504.06920.pdf.

[8] H. Shahriar, S. North, and W. Chen, “Early detection of SQL injection attacks,” *International Journal of Network Security & Its Applications (IJNSA)*, vol. 5, no. 4, July 2013, http://airccse.org/journal/nsa/5413nsa04.pdf.

[9] R. Alward, “Early SQL injection detection,” St. Cloud State University, St. Cloud, MN, 2015.