A close-up of the sun

Description automatically generated with low confidenceA picture containing text, clipart

Description automatically generated

SQL Attacks Detection and Prevention

Project: Milestone 4: System Administration Guide

Student’s Name: Mubasher Mehnaz Begum

Grand Canyon University

CST-590-0500-Computer Science Capstone Project

Instructor Name: Aiman Darwiche

Detection of SQL injection Attacks and Prevention

System Admin Guide

02/2023

Version 1.2

Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 02/22/2023 | 1.2 | Created | Mubasher M Begum |

Copyright Notice

Original work Copyright 2023 Mubasher Mehnaz Begum

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

**Legal Notice**

Please read this disclaimer ("disclaimer") carefully before using this project (“project”, "application") provided by [Mubasher Mehnaz Begum] ("us", 'we", "our").

All information posted is merely for educational and informational purposes. It is not intended as a substitute for professional advice. Should you decide to act upon any information in this guide, you do so at your own risk.

While the information on this guide has been verified to the best of our abilities, we cannot guarantee that there are no mistakes or errors.

We reserve the right to change this policy at any given time, of which you will be promptly updated. If you want to make sure that you are up to date with the latest changes, we advise you to frequently visit this page.

Table of Contents

[Introduction 7](#_Toc128007572)

[1.0 Purpose 7](#_Toc128007573)

[2.0 System Overview 7](#_Toc128007574)

[3.0 System Configuration 7](#_Toc128007575)

[4.0 System Maintenance 10](#_Toc128007576)

[5.0 System backup 10](#_Toc128007577)

[6.0 Networking, and connectivity 10](#_Toc128007578)

[7.0 Access and authentication: 10](#_Toc128007579)

[8.0 Cabling and critical service 10](#_Toc128007580)

# Introduction

SQL Injection attacks (or SQLi) alter SQL queries and carried out by injecting malicious code by exploiting application vulnerabilities.  Bypassing validation on the feeble login page permits us to get to the information base and take client information without knowing their secret word. All data in the data set can be recovered just utilizing union’s based assault strategy.

# Purpose

The goal of this project is to create a safe channel for user transactions. Its goal is to prevent SQL injection when running queries against the database and to safeguard the database. Using a unique static and dynamic analytic approach to detect SQL injection threats. This approach eliminates SQL query metadata at execution (nonlinear system) and contrasts those to SQL queries previously examined (static method).

# System Overview

The proposed project can be deployed on any server or installed locally.

**The project has following components:**

* Frontend Application: Which presents the User Interface for the Login application. This is written in React (JavaScript).
* Java Rest API: Utilizes Java Spring boot framework for developing a RESTful API that React application calls for “/authenticate” POST call. This middleware connects with My SQL
* Database: The Database for the application is MSySQL. It contains table for Users to retrieve a list of users and their roles.

# System Configuration

Frontend Application Configuration:

Tools Required:

* Open-source Microsoft Visual Studio Code
* Node JS, Node package Manager

Installation Steps:

* Front End Application needs to be run locally or deployed on to the server.
* Local configuration needs Node Js to be Running on the system followed by Node Package Manger (NPM).
* Go to the provided source code folder: “UI-code” and open in Visual studio Code.
* Open Terminal and cd to folder: “UI-code” and enter the following in the terminal to install all the required dependencies:
  + Npm install
* Once that is complete, application can be started by entering following in the same terminal in VS Code
  + Npm start
* Application is started and navigate to any browser and type the following
  + <http://localhost:3000>
* Validate the application running if we see the following screen

Front end Login Page

Graphical user interface, application

Description automatically generated

Backend Application Configuration:

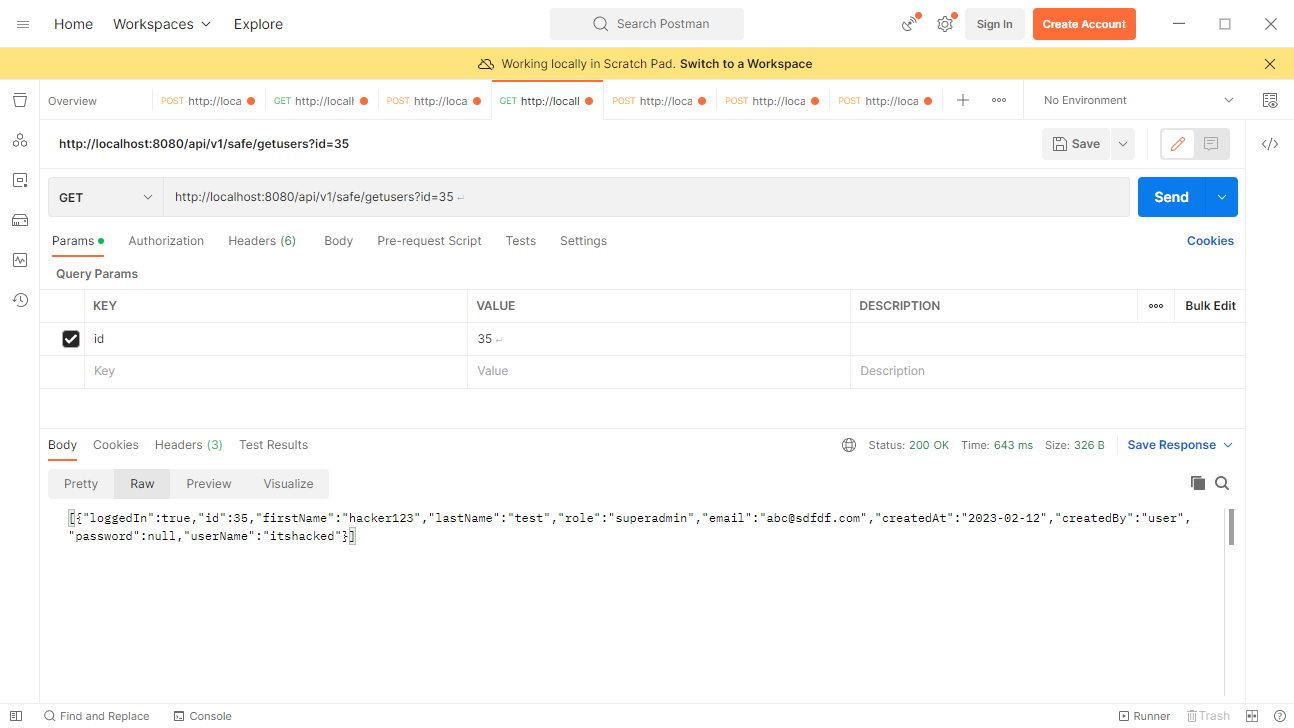
Tools Required:

Install the following Tools & Open source libraries:

* Open-source Java SDK
* IntelliJ Idea Community version
* Microsoft MySQL Installed, MySQL Workbench
* Postman

Installation Steps:

* Backend End Application needs to be run locally or deployed on to the server.
* Local configuration for MYSQL, download MySQL weork bench and install locally.
* Create a new Root user with default password.
* Start MySQL server through workbench
* Create the schema and database tables using the database script provided
* Insert few record into users table
* Extract the Source code provided into folder: “java-api”
* From IntelliJ Idea, load the java-api folder and perform mvn clean install that installs all the maven de pendencies
* Run the Java Springboot API using: **mvn spring-boot:run** in the terminal
* The rest API is running on:
  + <http://localhost:8080>
  + Validate API using postman by performing a Get request using end point: <http://localhost:8080/api/v1/safe/getusers?id=35>



# System Maintenance

As this is project for capstone, all the required code is provided that runs on stand alone local installation. No system maintenance is required. If this application is deployed on the server, regular server maintenance can be done. No explicit system maintenance is required from application perspective.

# System backup

As all the required code to run the project is provided along with capstone to deploy the application on local machine or even when deployed on server, backup is not needed. Database that is deployed on server can be backed up to prevent any data loss

# Networking, and connectivity

The web application can be deployed on local machine hence establishmen of networking/connectivity is not needed. Even when we plan to deploy the web application on any server, the connectivity needs to be opened between MYSQL instance of database instance and the Java API accessing the database and the API and Frontend application needs to be deployed on same domain.

# Access and authentication:

The access and authentication of database: My SQL is Root and default password that is being used. This can be changed to custom user access based on where the database is being deployed. For local installation, the root user/default password holds good. The Rest API or Front app doesnot need any access/authentication to access the application.

# Cabling and critical service

Not relevant in the capstone context