CSD 326 Group 17

Suchit Reddi 2010110507

Snehasri Ravishankar 2010110638

Devanssh Agarwal 2010110220

Cyber Sentinel

An intentionally "vulnerable" web application

SHIV NADAR

INSTITUTION OF EMINENCE DEEMED TO BE UNIVERSITY

DELHI NCR

Index

Problem Statement

Malware is malicious software including but not limited to spyware, ransomware, viruses, and worms. Once inside a system, malware can:

- Block access to critical components of a network and render systems inoperable

- Install additional harmful software

- Discretely derive information by transmitting data from the hard drive (spyware)

Cyberattacks are unwelcome attempts to steal, expose, alter, hijack, or destroy data through unauthorized access to computer systems. These days, most website or application developers are unaware of different vulnerabilities in their programs.

The main aim of our project is to **create awareness** of common yet dangerous vulnerabilities by creating a vulnerable web application that **simulates different types of cyberattacks and then shows how to mitigate them**. We focus primarily on web application vulnerabilities as they are less complex but more common, increasing their damaging capacity.

After demonstrating how these attacks are carried out, we show how to patch a website against the demonstrated attack. We will implement **tutorials** to better understand these attack processes, vulnerable code, and its step-by-step patching.

We will have **code segments illustrating the security flaws** and how to patch them, using languages and frameworks including but not limited to PHP, HTML, and CSS. We will secure the vulnerable web application by **containerizing it with docker**, which is like a virtual machine.

Example of an attack: Cross-Site Scripting (XSS) attacks are done by injecting scripts into trusted websites. Malicious code leading to stolen cookies/session tokens can be sent to different end users in the form of browser side scripts using this vulnerability. This vulnerability can be patched by not allowing the insertion of <script> tag, generating Anti-CSRF tokens, or fixing the input that can be given.

Software Requirement Specification (SRS)

1.0 PROBLEM DEFINITION

Cyber Sentinel is a vulnerable web application that provides insights and information on the most common yet catastrophic software vulnerabilities and how to mitigate them.

1.1 The user can simulate cyber-attacks on the vulnerable web application by following the detailed steps.

1.2 They can learn where the vulnerabilities in the code are, and how to patch them.

2.0 SOFTWARE REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION

2.1.1 Purpose

2.1.1.1 The purpose of this SRS document is to describe the requirements involved in developing a vulnerable web application for testing and learning purposes.

2.1.1.2 The intended audience is anyone who develops software and web applications with basic knowledge of interacting with web applications.

2.1.2 Scope

2.1.2.1 The product is titled Cyber Sentinel.

2.1.2.2 The product will perform the following tasks:

2.1.2.2.1 Select a vulnerability from a drop-down menu.

2.1.2.2.2 Demonstrates the exploitation of vulnerabilities in the selected attack.

2.1.2.2.3 Shows how to patch the code by fixing known vulnerabilities.

2.1.2.3 The product is adaptable to change, as developers can keep on adding new vulnerability modules even after the product's final delivery phase.

2.1.3 Definitions, Acronyms, and Abbreviations

2.1.3.1 VM - Virtual Machine

2.1.3.2 OS - Operating System

2.1.3.3 XSS - Cross-site Scripting

2.1.3.4 DOM - Document Object Model

2.1.4 References

This project has references from websites like OWASP, Port Swigger, and Hacksplaining. The code for the project is from similar existing projects.

2.1.5 Overview

2.1.5.1 The requirements needed for the smooth development of this web application.

2.1.5.2 The overall description provides the requirements, such as interfaces or tools used, constraints, product functions, and user characteristics for the Cyber Sentinel software.

2.2 THE OVERALL DESCRIPTION

2.2.1 Product Perspective

2.2.1.1 Hardware Interfaces

2.2.1.1.1 It is advised to use this application in a VM to increase the security of the host OS. This application comes as a docker for this very same purpose.

2.2.1.1.2 This uses basic hardware interfaces like a keyboard and mousepad.

2.2.1.1.3 The host OS will be secure by containing the vulnerable application in Docker in the final stages of product development.

2.2.1.2 Software Interfaces

2.2.1.2.1 Front End: HTML, CSS, JavaScript, and PHP.

2.2.1.2.2 Back End: MySQL database.

2.2.1.3 Memory Constraints

2.2.1.3.1 No specific constraints on memory.

2.2.1.3.2 Memory constraints apply only when the application uses virtual machines, but these constraints are due to the VM rather than the application.

2.2.1.4 Operations

2.2.1.4.1 The application takes the user credentials to log in.

2.2.1.4.2 It provides a drop-down menu to select a vulnerability the user wants.

2.2.1.4.3 The application simulates the selected attack along with detailed steps on its execution.

2.2.1.4.4 Along with the demonstration, the patches for the vulnerability and tips on how the user can protect themselves will be displayed in a tutorial.

2.2.1.4.5 The users can select the level of difficulty they want for the application.

2.2.2 Product Functions

2.2.2.1 The software validates the user's authentication by extracting their username and password.

2.2.2.2 The application will allow the user to choose different vulnerabilities.

2.2.2.3 The application will demonstrate the attack with detailed steps on what is happening to the application during the attack process.

2.2.2.4 Once the application demonstrates the attack, it will make the user aware of how to protect themselves from these vulnerabilities by providing fixes and patches.

2.2.2.5 Different difficulty levels which have different levels of patches are available for the user to select from.

2.2.3 User Characteristics

2.2.3.1 Few attacks demonstrated on this application need a basic understanding of websites and networks.

2.2.3.2 Most of the attacks simulated by the software do not require the users to have specific knowledge of the application's internal workings.

2.2.3.3 The product does not need the user to possess extensive technical knowledge. Anyone who can use the internet can use this product successfully by following the guides.

2.2.4 Constraints

2.2.4.1 The application does not let the user select any vulnerability without creating the database first, which is necessary to ensure the smooth running of some attacks.

2.3 SPECIFIC REQUIREMENTS

2.3.1 Logical Database Requirements

2.3.1.1 The system must contain databases for the necessary information for the functioning of the application. This information includes user details and specific data entries for attacks like Stored Cross-Site Scripting.

2.3.1.2 User details refer to the username, and hashed password.

2.3.1.3 Adding new attacks may require new database tables depending on the type of attack.

2.4 FRONT - END DESCRIPTION

Cyber Sentinel is a web application that is a concoction of many web pages. Every page uses a fixed template page, sentinelPage.inc.php. PHP, JavaScript, HTML, and CSS are used.

The main sign-in page takes the username and password. Each attack will need a unique set of web pages. The demonstration page, where the user can try attacking the system. The tutorial page, where the user can learn about the vulnerability, how to use the vulnerability in this application, the vulnerable code, and finally how to patch it.

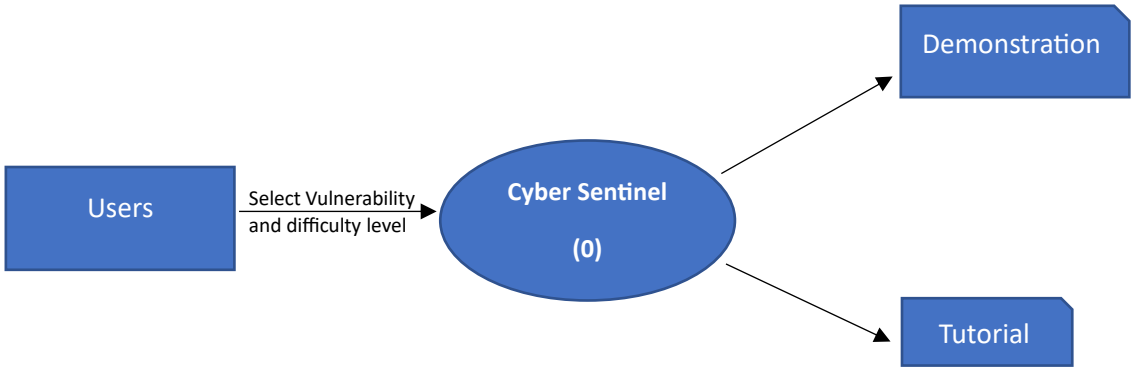
2.5 BACK - END DESCRIPTION

The database used for backend is MySQL. The login data of the application will be stored as a username, and password in a table named “users”. Few attacks will require additional fields. For example, XSS-Stored will use a table “guestbook”, to store comments and names. This part of the development process is volatile, as developers can incorporate more attack modules after completing software delivery, which might need more tables.

2.6 DATA STRUCTURES

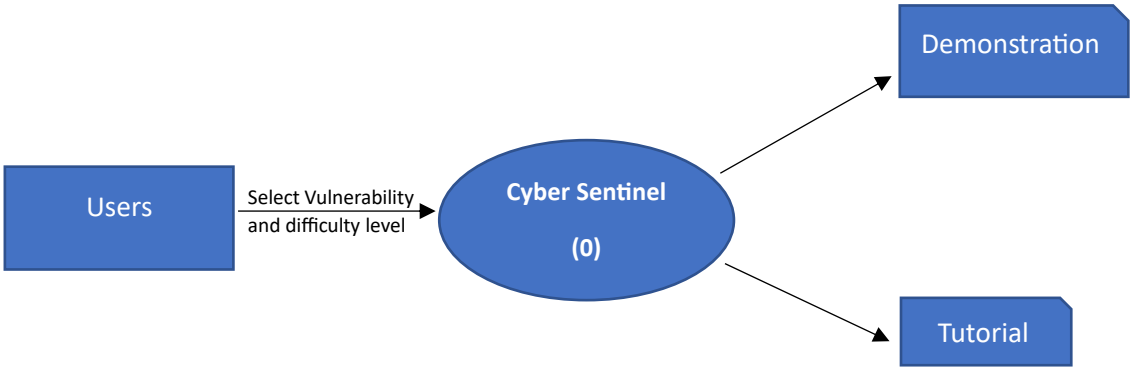
FIELD NAME	TYPE
	Table - users
user_id	text with special char
password	text with numbers and special characters (hashed)
	Table - guestbook
name	text
comment	text with numbers and special characters

2.7 DATA FLOW DIAGRAM

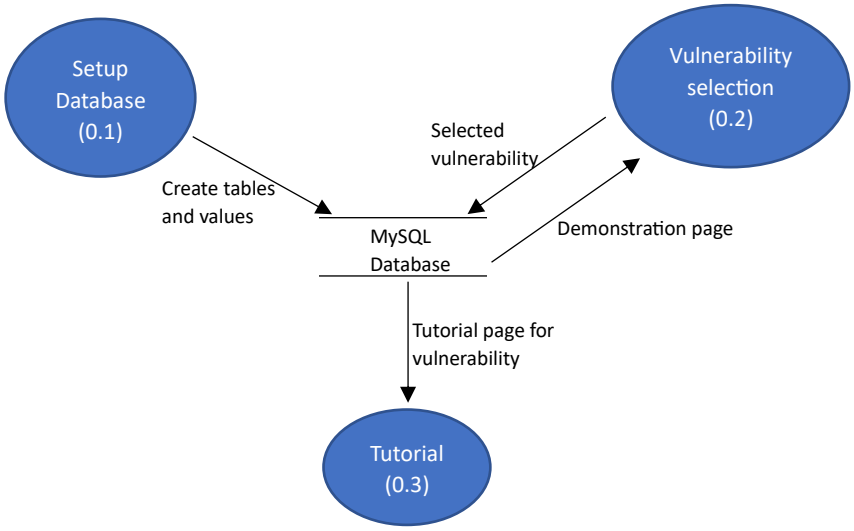


Data Flow Diagram (DFD)

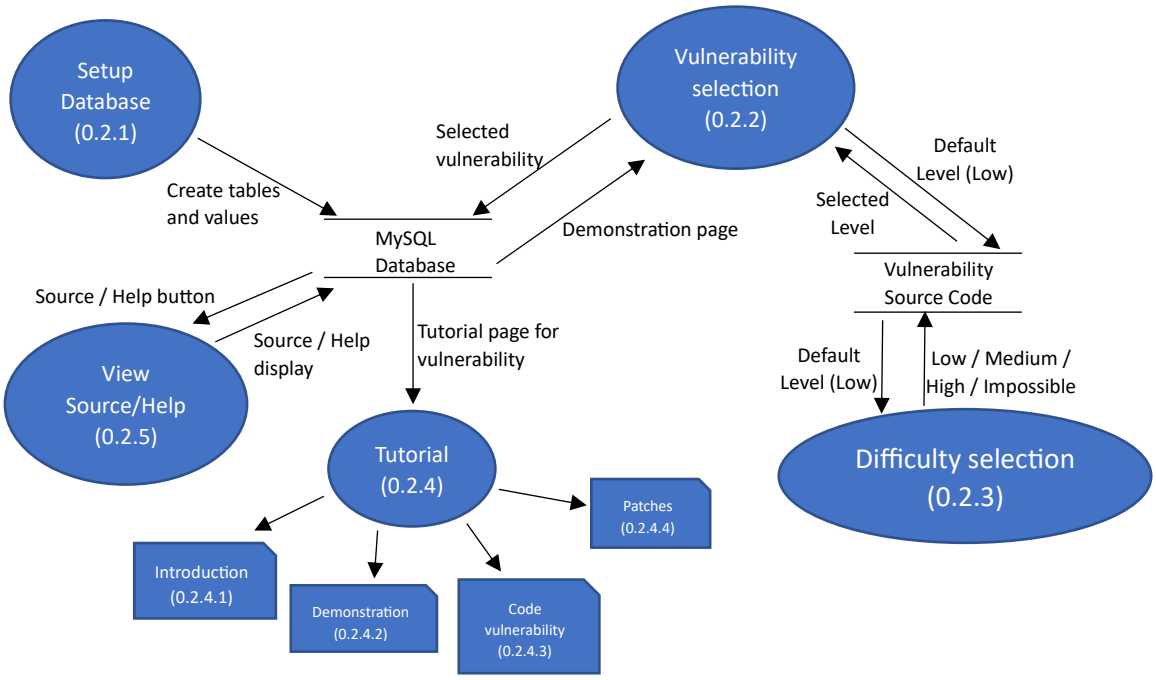
Level 0



Level 1



Level 2


Setup Database (0.2.1)
Vulnerability selection (0.2.2)
Create tables and values
Selected vulnerability
MySQL Database
Demonstration page
Default Level (Low)
Selected Level
Vulnerability Source Code
Source / Help button
Source / Help display
Tutorial page for vulnerability
View Source/Help (0.2.5)
Default Level (Low)
Low / Medium / High / Impossible
Tutorial (0.2.4)
Difficulty selection (0.2.3)
Patches (0.2.4.4)
Introduction (0.2.4.1)
Demonstration (0.2.4.2)
Code vulnerability (0.2.4.3)