

**ADDIS ABABA UNIVERSITY**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY**

**CENTER OF INFORMATION TECHNOLOGY AND SCIENTIFIC COMPUTING**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**Vulnerability Analyzer**

Project Proposal

PREPARED BY:

Arisema Mezgebe Mihretu ATR/7645/07

Kbrom Abadi Nrea ATR/9041/07

Tabor Nekatibeb Shiferaw ATR/9889/07

ADVISOR:

Abebe Asefa

Date: December, 2018

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| December 16,2018 | Version 1.0 | AKT | First Draft |
| December 19, 2018 | Version 2.0 | AKT | Final Version |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Abebe Asefa | Advisor | December 19, 2018 |

**Table of Contents**

[Document Approval 1](#_Toc533029052)

[List of Tables 4](#_Toc533029053)

[List of figures 4](#_Toc533029054)

[Definitions, Acronyms, and Abbreviations 5](#_Toc533029055)

[DECLARATION 6](#_Toc533029056)

[1. Introduction 7](#_Toc533029057)

[1.1 Purpose 7](#_Toc533029058)

[1.2 Scope 7](#_Toc533029059)

[1.3 Overview 7](#_Toc533029060)

[2. General Description 8](#_Toc533029061)

[2.1 Product Perspective 8](#_Toc533029062)

[2.2 Product Functions 8](#_Toc533029063)

[2.3 User Characteristics 8](#_Toc533029064)

[2.4 General Constraints 9](#_Toc533029065)

[3. Specific Requirements 9](#_Toc533029066)

[3.1 External Interface Requirements 9](#_Toc533029067)

[3.1.1 User Interfaces 9](#_Toc533029068)

[3.1.2 Hardware Interfaces 11](#_Toc533029069)

[3.1.3 Software Interfaces 11](#_Toc533029070)

[3.1.4 Communications Interfaces 11](#_Toc533029071)

[3.2 Functional Requirements 11](#_Toc533029072)

[3.2.1 FR. 1. Accepts only PHP based project 11](#_Toc533029073)

[3.2.2 FR. 2. Generate detailed report 12](#_Toc533029074)

[3.2.3 FR. 3. Rate the severity of vulnerabilities found 12](#_Toc533029075)

[3.2.4 FR. 4. Patch guide 13](#_Toc533029076)

[3.2.5 FR. 5. Identify Operation affected 13](#_Toc533029077)

[3.2.6 FR. 6. Provide API 14](#_Toc533029078)

[3.2.7 FR.7. Detect Misconfigurations 14](#_Toc533029079)

[3.3 Use Cases 15](#_Toc533029080)

[3.3.1 UC-01: Generate detailed report 15](#_Toc533029081)

[3.3.1 UC-02: Rate the severity of vulnerabilities 16](#_Toc533029082)

[3.3.1 UC-03: Patch guide 16](#_Toc533029083)

[3.3.1 UC-02: Identify Operation affected 17](#_Toc533029084)

[3.3.1 UC-02: Consume API 17](#_Toc533029085)

[3.4 Non-Functional Requirements 18](#_Toc533029086)

[3.4.1 Performance 18](#_Toc533029087)

[3.4.2 Reliability 18](#_Toc533029088)

[3.4.3 Availability 18](#_Toc533029089)

[3.4.4 Security 18](#_Toc533029090)

[3.4.5 Maintainability 18](#_Toc533029091)

[3.4.6 Portability 18](#_Toc533029092)

[3.5 Inverse Requirements 18](#_Toc533029093)

[3.6 Design Constraints 19](#_Toc533029094)

[3.7 Logical Database Requirements 19](#_Toc533029095)

[3.8 Other Requirements 19](#_Toc533029096)

[4. Change Management Process 20](#_Toc533029097)

[Bibliography 21](#_Toc533029098)

[A. Appendices 22](#_Toc533029099)

[A.1 Flow Chart 22](#_Toc533029100)

# List of Tables

[Table 1: FR. 1. Accepts only PHP based project 11](#_Toc533028765)

[Table 2: FR. 2. Generate detailed report 12](#_Toc533028766)

[Table 3: FR. 3. Rate the severity of vulnerabilities found 12](#_Toc533028767)

[Table 4: FR. 4. Patch guide 13](#_Toc533028768)

[Table 5: FR. 5. Identify Operation affected 13](#_Toc533028769)

[Table 6: FR. 6. Provide API 14](#_Toc533028770)

[Table 7: FR.7. Detect Misconfigurations 14](#_Toc533028771)

[Table 8: Logical Database 19](#_Toc533028772)

# List of figures

[Figure 1: Analysis Tab View 9](#_Toc533028773)

[Figure 2: Recommendation Tab View 10](#_Toc533028774)

[Figure 3: API Documentation Page 10](#_Toc533028775)

[Figure 4: Use Case Diagram 15](#_Toc533028776)

[Figure 5: ER Diagram 19](#_Toc533028777)

[Figure 6: System UI Flow Diagram 22](#_Toc533028778)

# Definitions, Acronyms, and Abbreviations

HTTP: Hyper Text Transfer Protocol

HTML: Hyper-Text Markup Language

# DECLARATION

We declare that this written submission represents our ideas in our own words and where others’ ideas or words have been included. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**Arisema Mezgebe Mihretu**

**Kbrom Abadi Nrea**

**Tabor Nekatibeb Shiferaw**

**Date: December 16, 2018**

# 1. Introduction

This part of the document includes the purpose, scope summary and overview of the details noted in this Software Requirement Specification document for the Static Source Code Analyser system.

## 1.1 Purpose

This document is written to describe and detail the domain area, product description and function, infrastructure requirements and some application examples of the system. To the development team that reads this document, the specifications of the Static Code Analyzer are to be explained and demonstrated well.

## 1.2 Scope

**Boundaries**

This project and the production of this system is intended for programmers and developers who use the PHP programming language. In order to reap the full functionalities provided, an intermediate understanding of the language as well as the testing and debugging of web programs is required.

**Features**

1. The Vulnerability Analyzer will analyze supplied source code in the php source code and discover vulnerabilities
2. The system will provide the vulnerabilities of the code and relay it to the user and show it in a way that is intended to help improve those vulnerabilities. It will also provide methods that can be used to solve the security hole.
3. The Vulnerability Analyzer is also provided in an API format to help incorporate the system into existing tools and software products.
4. Plugins are provided to ease the process of providing the source code to the system.

## 1.3 Overview

As stated in the purpose following three chapters of this document are written with the intention of providing a more thorough detailing of the Vulnerability Analyzer.

In the next chapter are reflected the descriptions of the product in terms of its perspective and the functions it provides. The second chapter also includes the intended user base, the constraints expected in the development of this system, as well as the dependencies of the Vulnerability Analyzer and the assumptions that make the basis for its production.

The third chapter includes the product requirement details and description of the systems functionalities. This contents of this chapter are planned to give a technical description of the system to its developers.

Lastly, the fourth chapter covers the issues of the change management process. It describes tactics to put in action, and the activities to undertake in order to accommodate and integrate new or changing requirements.

# 2. General Description

## 2.1 Product Perspective

The Vulnerability Analyzer is developed with the hopes of providing PHP programmers with the discovery and analysis tools of their source code. This system is expected to help them provide a more solid application in terms of security. This will in turn help to solve the issues that arise from using and providing vulnerable systems.

## 2.2 Product Functions

At the production phase of the Vulnerability Analyzer, the system is expected to supply users that have supplied their source code, its vulnerability analysis.

The system will provide the proper methods that are to be performed in order to mediate the issues that have been discovered.

Its plugin application will be built with the aim of supplying potential users with the choice of integration to certain code editors, hence encouraging a more thorough source code development.

The Vulnerability Analyzer can also be accessed in API format in order to facilitate works that could be performed using its provided functionalities.

## 2.3 User Characteristics

The general user base for this application is programmers that use PHP programming language for web development. The system can also be accessed by API users that may require to incorporate certain functionalities from the Vulnerability Analyzer into their system. Based on expertise required to perform the functionalities of the system, the two types of users are elaborated as follows:

**Code Analyzer User:**

Frequency of use: As Needed

Subset of product function used: the code analyzer and the plugin in the scenario that they use a code editor that has plugin available for it.

Minimal technical expertise: understanding of using browsers and websites.

Minimum education level: Website use knowledge.

**API User:**

Frequency of use: As Needed

Subset of product function used: the code analyzer API.

Minimal technical expertise: basic understanding of using and integrating APIs.

Minimum education level: API programming skills.

## 2.4 General Constraints

One of the major constraints is the availability of the system. The servers that this software will be deployed on should provide a consistent service to its users.

Portability The system will be built in a way that provides access to its functionalities through browsers and API.

Another constraint that is the matter of the Vulnerability Analyzer being reliable. This could be achieved in letting users have accurate analysis of their program as well as an simplified API.

**2.5 Assumptions and Dependencies**

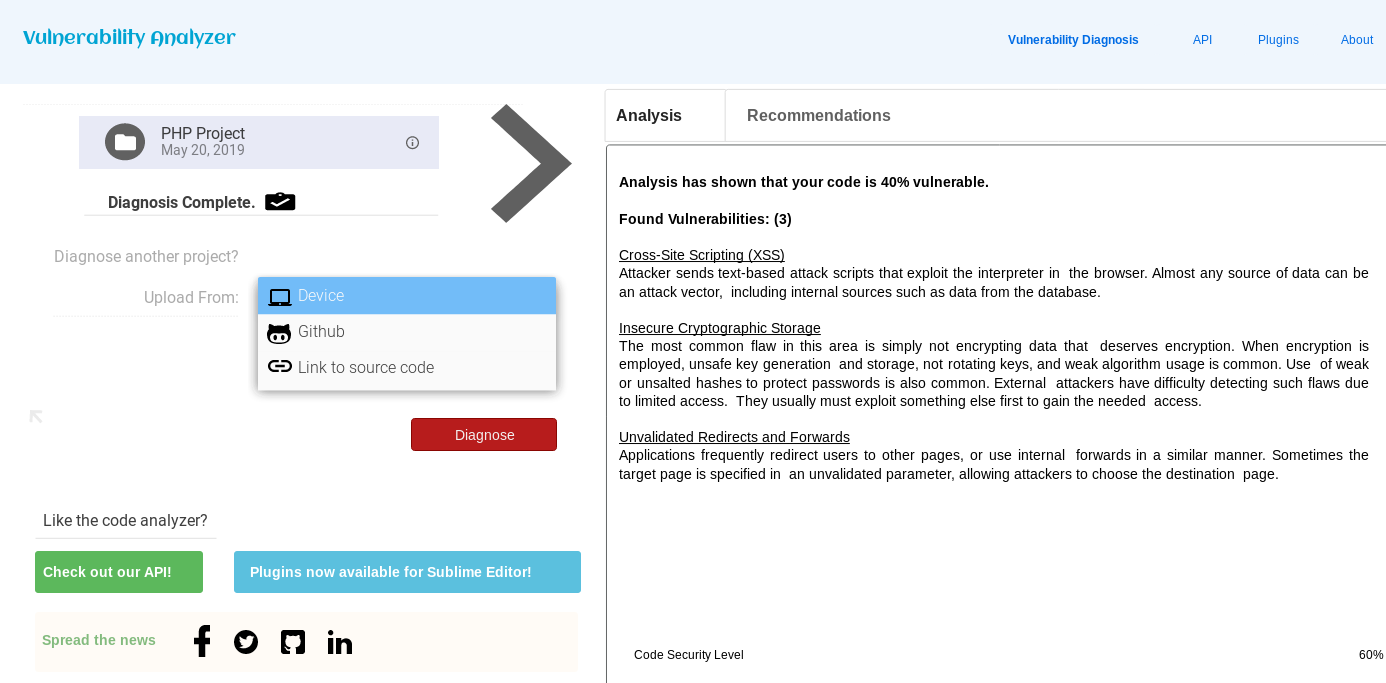
The main assumption that is to be taken into account while developing this system is that its users are knowledgeable of the occurrence and variability of vulnerabilities especially due to the use of PHP programming language in building web services and sites.

# 3. Specific Requirements

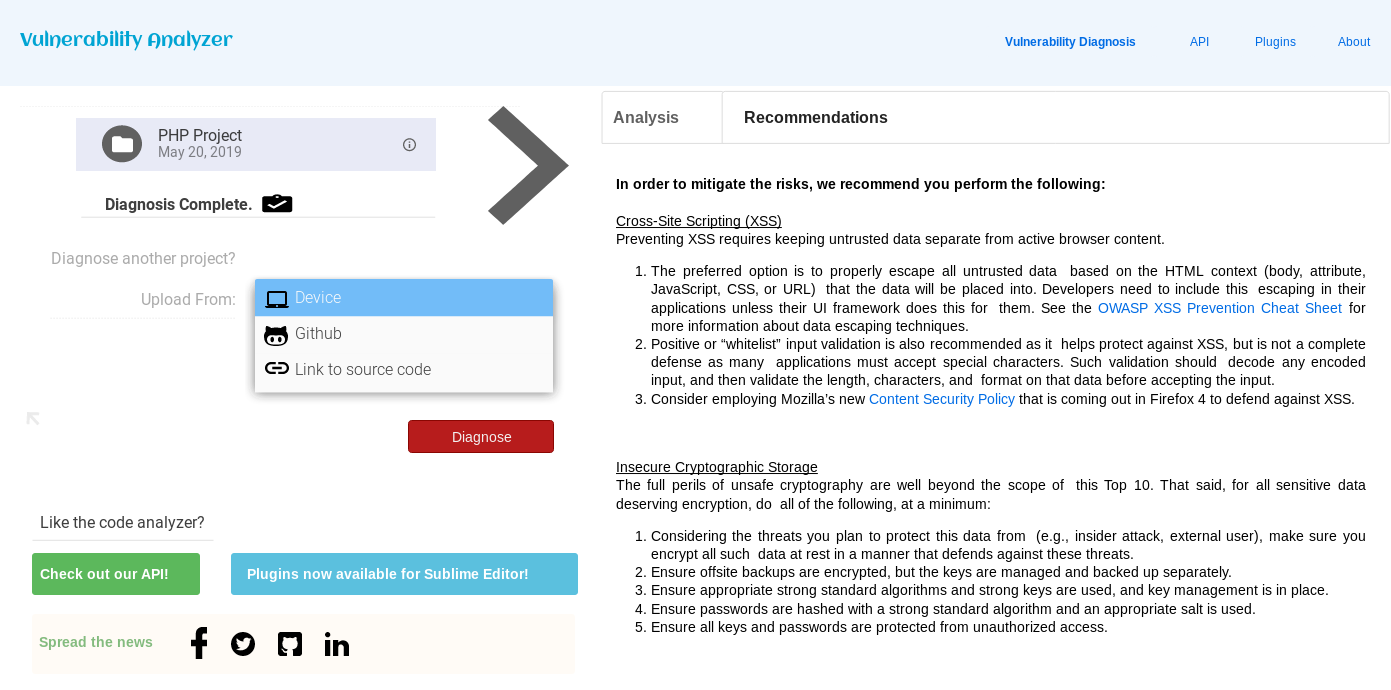
## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

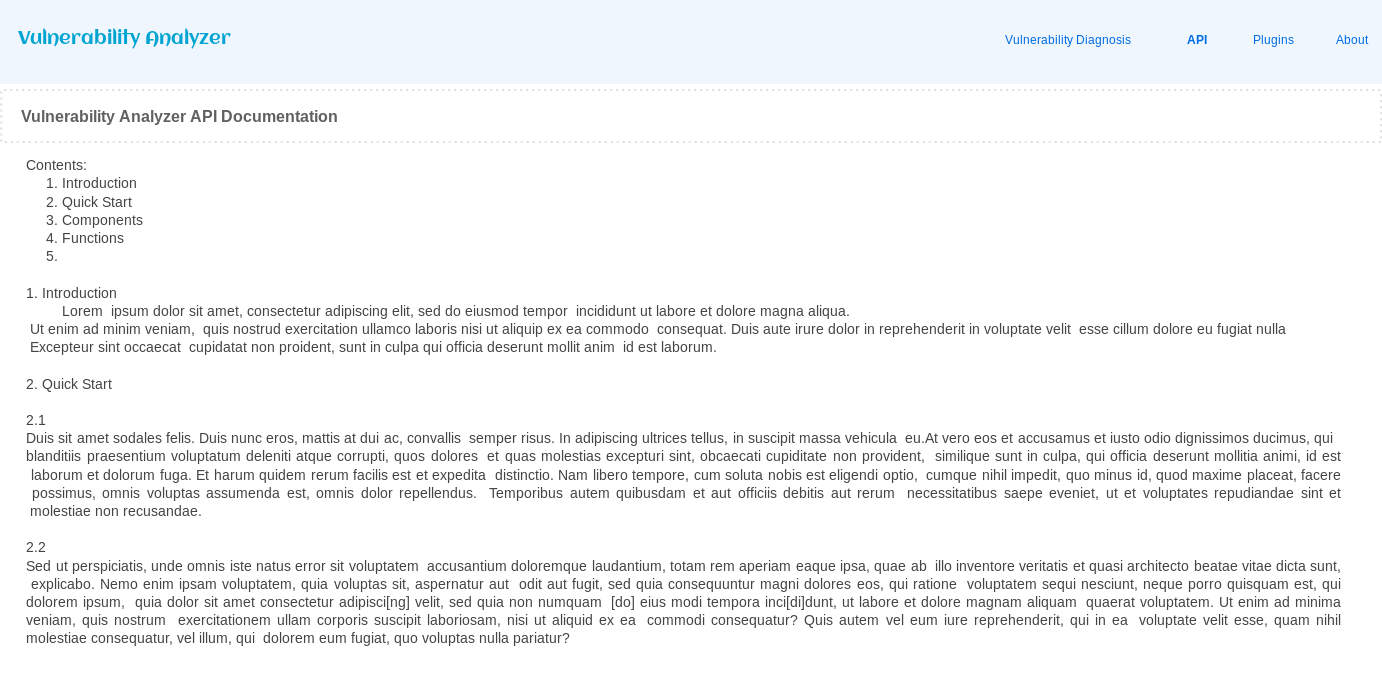
The users interfaces that allow the users to interact with the system are mainly in web format. The main site for the system includes a simple interface for the static source code analysis purpose, descriptions of the website and plugins, as well as documentation for the API. These different contents will be delivered in a single-page application format.



**Figure 1: Analysis Tab View**



**Figure 2: Recommendation Tab View**



**Figure 3: API Documentation Page**

### 3.1.2 Hardware Interfaces

The Vulnerability Analyzer is not dependent on the hardware specifics of the users’ machines.

### 3.1.3 Software Interfaces

The software interface required to interact with the web based application of the system is a browser that has supports HTML5 and JavaScript.

The plugins can be incorporated to their corresponding code editors using the code editor’s interface. Hence, the actual code editing software is required to allow the integration of plugins.

### 3.1.4 Communications Interfaces

Owing to the web based format of the system’s functionalities, HTTP is used to access them over the internet.

## 3.2 Functional Requirements

### 3.2.1 FR. 1. Accepts only PHP based project

Table 1: FR. 1. Accepts only PHP based project

|  |  |
| --- | --- |
| ID | FR. 01 |
| Name | Accepts only PHP based project |
| Summary | The system shall accept only PHP projects, and if it finds out a non-PHP based project, it shall give error feedback. |
| Input | Software project |
| Process | The system shall go through the source code and identify if it is a PHP project. |
| Output | Error message saying either that only PHP based projects are accepted or the source code shall proceed to further processing. |

### 3.2.2 FR. 2. Generate detailed report

Table 2: FR. 2. Generate detailed report

|  |  |
| --- | --- |
| ID | FR. 02 |
| Name | Generate detailed report |
| Summary | Generate report detailing all the vulnerabilities found in the source code. |
| Input | Software project |
| Process | The system shall go through the source code and identify vulnerabilities and generate report regarding the vulnerability identified. |
| Output | PDF document. |

### 3.2.3 FR. 3. Rate the severity of vulnerabilities found

Table 3: FR. 3. Rate the severity of vulnerabilities found

|  |  |
| --- | --- |
| ID | FR. 03 |
| Name | Rate the severity of vulnerabilities found |
| Summary | The system shall rate the vulnerabilities found as severe, medium or low risk |
| Input | Software project |
| Process | The system shall go through the source code and identify vulnerabilities and rate the vulnerabilities found based on their potential risk. |
| Output | Information identifying the risk level |

### 3.2.4 FR. 4. Patch guide

Table 4: FR. 4. Patch guide

|  |  |
| --- | --- |
| ID | FR. 04 |
| Name | Patch guide |
| Summary | The system shall suggest possible fix for vulnerabilities found. |
| Input | Software project |
| Process | Once all the vulnerabilities are identified the system shall be able to suggest potential mitigation techniques. |
| Output | Possible Mitigation methods |

### 3.2.5 FR. 5. Identify Operation affected

Table 5: FR. 5. Identify Operation affected

|  |  |
| --- | --- |
| ID | FR. 05 |
| Name | Identify Operation affected |
| Summary | The system shall be able to specifically identify the part of the code that result in the vulnerability. |
| Input | Software project |
| Process | The system parses every part of the the software project and identifies the part of the code that caused the problem. |
| Output | Colorizes affected operations |

### 3.2.6 FR. 6. Provide API

Table 6: FR. 6. Provide API

|  |  |
| --- | --- |
| ID | FR. 06 |
| Name | Consume API |
| Summary | The core system shall provide API for third party application to consume |
| Input | API request |
| Process | The system shall be able to accept several requests and provide response accordingly. |
| Output | JSON response |

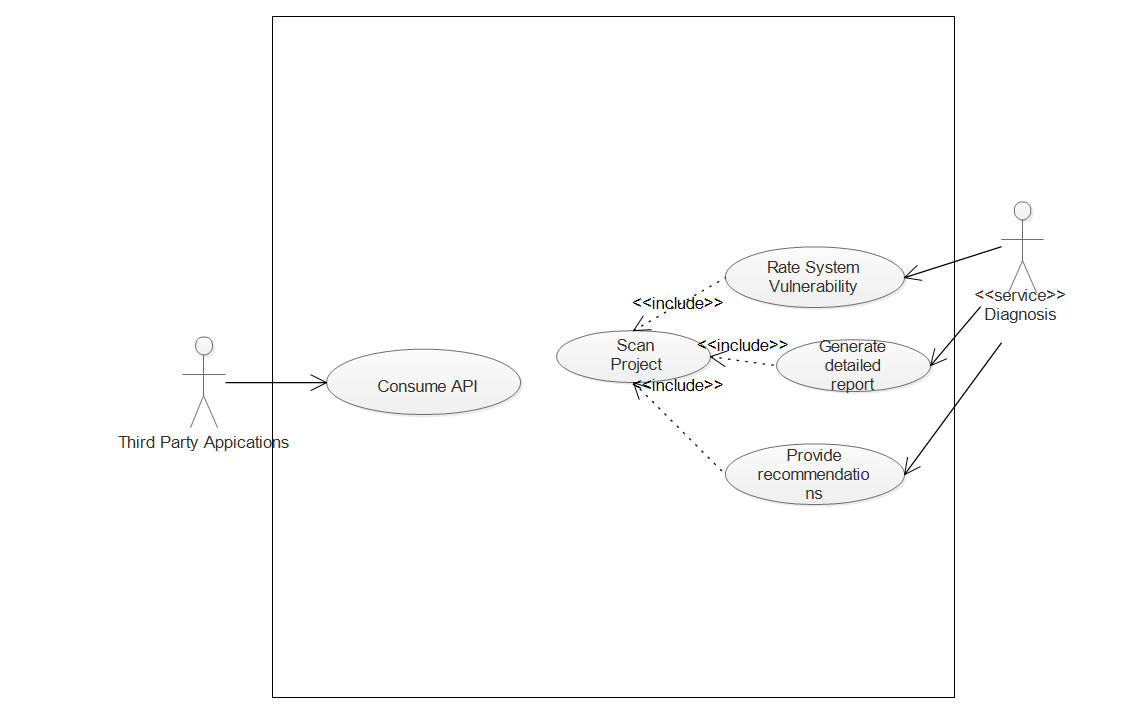
### 3.2.7 FR.7. Detect Misconfigurations

Table 7: FR.7. Detect Misconfigurations

|  |  |
| --- | --- |
| ID | FR. 07 |
| Name | Detect Misconfigurations |
| Summary | The system shall identify configurations that possibly open up the software for potential security attacks. |
| Input | Software Project |
| Process | Scanning the code, the system shall look up for misconfigured parameters |
| Output | List of misconfigured parameters |

## 

## 3.3 Use Cases

****

**Figure 4: Use Case Diagram**

### 3.3.1 UC-01: Generate detailed report

Use case name: UC-06: Generate detailed report

Primary Actors: Diagnosis Service

Summary: Generate detailed report

Preconditions: PHP project has to be provided

Trigger: Request for report

Main Success Scenario

1. The user uploads a software project

2. Clicks on “Diagnose” or “Scan” button

3. The system process the input

4. The system outputs summary vulnerability report

5. The user clicks on report

6. Downloading of the generated report starts

Alternative Path

4a. No vulnerability found

4b. The system prints message saying “The provided project is not PHP based”

### 3.3.1 UC-02: Rate the severity of vulnerabilities

Use case name: Rate the severity of vulnerabilities

Primary Actors: Diagnosis Service

Summary: Rate the severity of vulnerabilities found

Preconditions: PHP project has to be provided

Trigger: Scanning the project is completed successfully

Main Success Scenario

1. The user uploads a software project

2. Clicks on “Diagnose” or “Scan” button

3. The system process the input

4. The system outputs summary vulnerability report.

5. The user clicks on “Analysis” or “Issue breakdown” tab.

6. The system provides a table of all issues found with their corresponding ratings.

Alternative Path

4a. No vulnerability found

4b. The system prints message saying “The provided project is not PHP based”

### 3.3.1 UC-03: Patch guide

Use case name: Patch guide

Primary Actors: Diagnosis Service

Summary: Possible mitigation techniques

Preconditions: PHP project has to be provided

Trigger: User requests for it

Main Success Scenario

1. The user uploads a software project

2. Clicks on “Diagnose” or “Scan” button

3. The system process the input

4. The system outputs summary vulnerability report

5. The user clicks on “Analysis” or “Issue breakdown” tab.

6. The system provides a table containing list of all issues found

7. The user clicks on the issue of their choice

8. The user be taken to a page containing all details about the vulnerability found and suggestion to mitigate it.

Alternative Path

4a. No vulnerability found

4b. The system prints message saying “The provided project is not PHP based”

### 3.3.1 UC-02: Identify Operation affected

Use case name: Identify the operation affected

Primary Actors: Diagnosis Service

Summary: Identify the part of the code that result in the vulnerability

Preconditions: A vulnerability has to be found

Trigger: user requested for details of a vulnerability found

Main Success Scenario

1. The user uploads a software project

2. Clicks on “Diagnose” or “Scan” button

3. The system process the input

4. The system outputs summary vulnerability report.

5. The user clicks on “Analysis” tab.

6. The system provides a table of all issues found.

7. The user clicks on the vulnerability of their choice

8. From the page to which the user is redirected, the first section contains the vulnerable code snippet and the affected operation is color coded.

Alternative Path

4a. No vulnerability found

4b. The system prints message saying “The provided project is not PHP based”

### 3.3.1 UC-02: Consume API

Use case name: Consume API

Primary Actors: Third party applications

Summary: Third party applications consume API

Preconditions: The third party application has to be registered to the system

Trigger: Third party app sends API request

Main Success Scenario

1. A registered app sends

2. Clicks on “Diagnose” or “Scan” button

3. The system process the input

4. The system outputs summary vulnerability report.

5. The user clicks on “Analysis” tab.

6. The system provides a table of all issues found.

7. The user clicks on the vulnerability of their choice

8. From the page to which the user is redirected, the first section contains the vulnerable code snippet and the affected operation is color coded.

Alternative Path

4a. No vulnerability found

4b. The system prints message saying “The provided project is not PHP based”

## 3.4 Non-Functional Requirements

### 3.4.1 Performance

Due to the behavior of the system, web, plugin and API based service, its performance solely depends on the dedicated server. Considering the current number of developers it requires a powerful server machine, perhaps a distributed computing mechanism. The system must be fast and the parsing and analysis time of a given input should be prioritized. While using the system users interfaces should be updated quickly. The system involves time consuming computations for large and multiple files. Hence, a noticeable delay between request and response is assumed to be normal.

### 3.4.2 Reliability

The system shall be tested separated into domain, component and subcomponent wise so as to provide a reliable service. Servers shall be replicated in order to tolerate faulty components of the system.

### 3.4.3 Availability

The average system availability should be above 99% of the time. Network failure is not considered as part of the system. Redundant servers are used while system updates and maintenance are configured, resulting in no interruption.

### 3.4.4 Security

Users shall have Login credentials, to prevent an unauthorized access of other users data. The system shall protect user’s data from unauthorized access. Any user data uploaded for analysis should be protected against theft and intervention.

### 3.4.5 Maintainability

The system shall provide flexibility for the user when changes and modifications are needed. It is expected that more vulnerabilities are to be uncovered through time, so, a modified and updated system should be always available as soon as possible.

### 3.4.6 Portability

The system shall reside in a remote server, away from users, providing an interface tunnel through web browsers and API. Plugin is assumed to be an entity acting as a user, because it sends requests through the API. But, there will still be an executable that will reside in the users code editor.

## 3.5 Inverse Requirements

The system does not correct the code errors.

The system does not actually correct the vulnerability issues.

## 3.6 Design Constraints

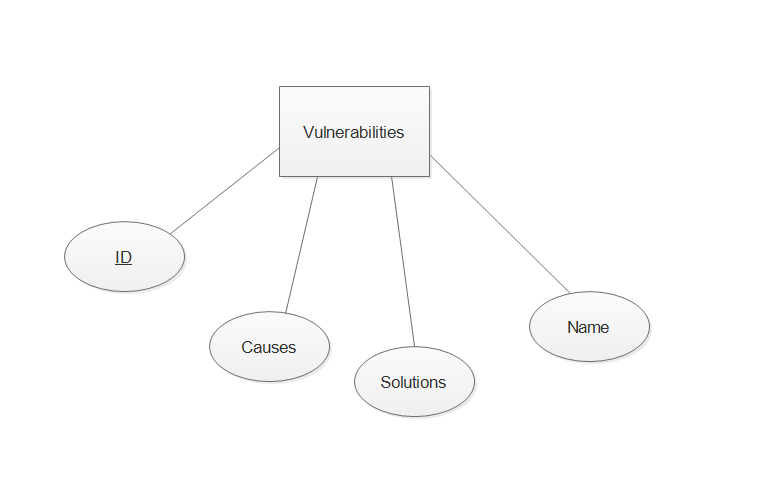
Accuracy and reliability are major constraints that will be addressed in the making of the Vulnerability Analyzer. Owing to the security testing nature of the system, it is mandatory to deliver a product that diagnoses and provides users with accurate information about the detected issues and the methods of mitigation them.

## 3.7 Logical Database Requirements

A data storage system will be maintained for the Vulnerability Analyzer system to keep records of the vulnerabilities, their causes and solutions.

Table 8: Logical Database

|  |  |  |
| --- | --- | --- |
| Data | Attribute | Use |
| Common Vulnerabilities Data | * ID * Vulnerability Name * Vulnerability Causes * Vulnerability Solutions | Used when providing the user with the solutions to the vulnerabilities that have been discovered. |



**Figure 5: ER Diagram**

## 3.8 Other Requirements

**Training-related Requirements**

In the API product of the Vulnerability Analyzer, accurate and thorough documentation is required in order for users to get the maximum possible use of the API functionality.

**Packaging Requirements**

No packaging requirements are necessary to access and make use of the system’s functionalities.

**Legal Requirements**

At the time of this SRS document, the system has not been patented, hence does not require legal compliance from users.

# 4. Change Management Process

During the development phase of the Vulnerability Analyzer, every change to the scope or requirements shall be addressed and assessed by the development team. The suggestions shall be addressed using any of the communication interfaces declared in the proposal. After analysis of the value and effect of each change, a decision to accommodate and integrate it into the system shall be decided by majority vote of the developers and, if it requires it, other stakeholders.

The use of agile development process in the construction of this system is expected to allow the process of mitigating changes feasible and permit the implementation of the accepted requirements.

# Bibliography

Arisema Mezgebe, Eman Shemsu, Hawi Tesfaye, Hayat Delil, Iman Abdulselam, Mahder Haileslasse, Meba Feyissa (April 2016), SRS, Ayate Home Remedies.

**Websites**

<https://www.owasp.org/index.php/Top_10_2010-A2-Cross-Site_Scripting_(XSS)> December 17, 2018 20:43

<https://www.owasp.org/index.php/Top_10_2010-A7-Insecure_Cryptographic_Storage> December 17, 2018 20:45

<https://www.owasp.org/index.php/Top_10_2010-A10-Unvalidated_Redirects_and_Forwards> December 17, 2018 20:46

# A. Appendices

## A.1 Flow Chart

System User Interface Flow Chart

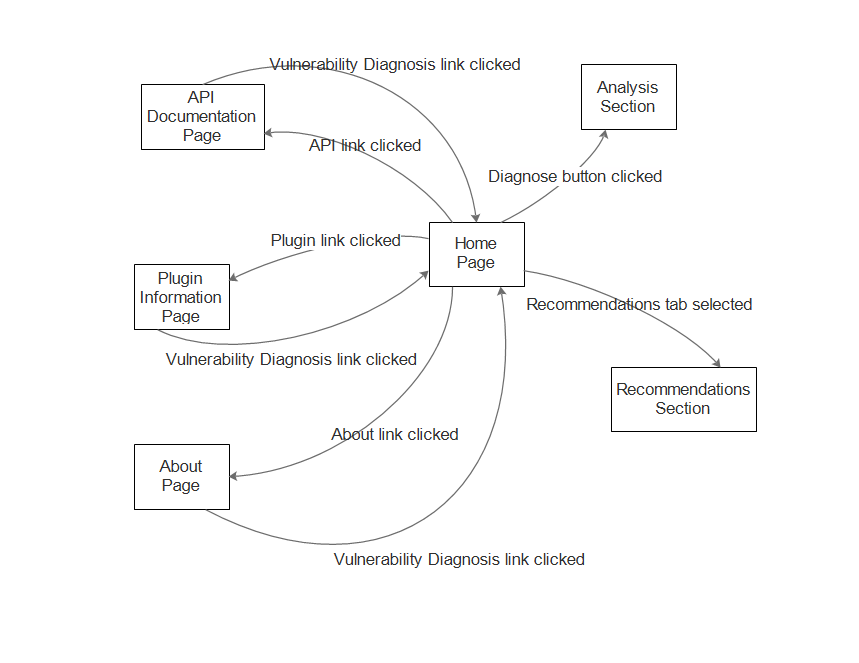


Figure 6: System UI Flow Diagram