Vulnerability Management Tool

Functional Specification

Student Name: Síne Doheny
Supervised by: Richard Butler
Student ID: C00226237

Table of Contents

Abstract

The purpose of this document is to outline the functionality behind how this application will operate. The document will outline the functionality that the application will provide and the actors that will be the users considered of the application. The document will discuss how the considered users will interact with the application and what the application will do in these scenarios. I will also discuss the target market of the application and the desired deliverables of the application.

Application

Introduction

Overview

The application to be developed is a vulnerability management web application. I would like to provide an organisation with a web application which manages the audit of a vulnerability from start to finish. The web application will be a tool that will organise vulnerability management with ease making the cycle of testing a system for a vulnerability, documenting and assigning the found vulnerabilities and mitigating and reporting the vulnerability an easy, organised, secure environment for handling the vulnerabilities that could put an organisation and its assets in a situation of risk.

Target Market

Information Technology vulnerabilities can affect any organisation of any size within any industry globally. On average the total number of malware events across all organisations today is roughly 170 million which means five malware events occur every second. The average cost a data breach according to IBM is currently 3.86 million dollars. In this case vulnerability management tools hold great value in an organisation. This web application is targeted towards any organisation which has an IT infrastructure. The following are the range of organisations to which this application is targeted:

Small Businesses

Any business with an internet connection that has staff who send and receive emails is susceptible to a cyber-attack. Firewalls are not designed to protect systems from vulnerabilities and a misconfigured firewall is a vulnerability in itself, antivirus software can catch known viruses but cannot catch unknown viruses. This web application will store all of these vulnerabilities in an organised manner for an organisation of small size so that they know immediately what needs to be fixed and how quick mitigation is required to occur.

Midsize Organisations

A mid-size organisation will have a more developed presence on the web compared to a small business, putting them in a position of a larger attack surface and thus a higher threat profile. IT teams often have so much going on that they do not have time to organise and consider the vulnerabilities present on their system. A vulnerability management web application where team members can see on any device what is going on in the organisations vulnerability surface in an organised manner will save time and cost when dealing with vulnerabilities.

Enterprise Organisations

Enterprise organisations have an even larger attack surface again and will always be key targets of cyber-attacks. With thousands of network nodes across all sorts of locations remote and in cities, this means there can also be thousands of vulnerabilities present too. With large databases of customer, employee and asset information it is essential to protect an enterprise organisations from the thousands of vulnerabilities that could be present.

Functionality

For my project I hope to create a web application which facilitates vulnerability management. I would like to provide an organisation with a tool that contains both test management and vulnerability management. The ideology of my tool is that it will track a test process and vulnerability process from start to finish, for example it will manage the test from setting up a test, to ensuring all legal factors and policies are complied with, then further a vulnerability test will be confirmed to go ahead and my tool will track its progress in a system from start to finish. It will provide information about the test, when it was or is to be conducted and then I will provide a vulnerability dashboard to display the vulnerabilities found in the associated asset in which the test was conducted, the application will then further show the prioritisation levels of a vulnerability or asset and it will be placed in a category of risk before patching. I will then have a dashboard where the administrator can log in and assign a vulnerability to a team or user globally, the final section of the application will display a live feed of the progress of vulnerability remediation. The owner of the vulnerability will be able to update this feed in accordance with the vulnerabilities progress. The administrator will then provide a report of the vulnerabilities progress through the vulnerability management life cycle. In doing so I am providing an organisation with a management tool for vulnerabilities on their system, essentially providing them with an area that states what needs fixing and how important and immediately the vulnerability must be mitigated. It is a vulnerability management tool which conveys the process of the audit from beginning to finish, an application which is not particularly used today and if it is used by an application it is only used in part.

Deliverables

Desirable characteristics of the web application include:

- A server which securely hosts the web application and performs all application operations efficiently.
- A database which securely stores user credentials, vulnerabilities, vulnerability test information, vulnerability information and vulnerability reports.
- Well-presented and easily navigable user interface.
- Desktop and mobile support of web application.
- Secure

Assumptions

- Time and resources will allow for the completed design and creation of all application components.

Risks

- Time may not allow for all deliverable components to be fully completed.
- The application may not work on different platforms.

Core functionality

The core functionality of the application includes authentication of users, the setting up of a test, the storage of vulnerabilities and the ability to assign a vulnerability to a user and have the vulnerability mitigated, updated as mitigated, re-tested and confirmed as mitigated.

- All users should be able to login with credentials and be validated in the application, this is important and specific users will have different forms of privileges.
- The credentials of the users will be securely stored in the database and retrieved and validated each time a user is attempting to login.
- The application must have the ability to set up a test and all test details such as legal sign off which is required etc.
- The primary functionality of the application is to store vulnerabilities safely in a quarantined database.
- The application should have the ability for an administrator to assign a vulnerability to a vulnerability management team user for mitigation.
- This user is required to mark the vulnerability as mitigated.
- The application should hold details of a vulnerability re-test after the above step to confirm mitigation is complete.

Secondary functionality

Secondary functionality of the application is as follows:

- Risk dashboard.
- Reports based on the full audit of the vulnerability.
- A notification sent to the company with confirmed vulnerability removal and the report attached.

Application Users

The application will consist of three key users, they are as follows:

Administrator:

The administrator will be the core user of the application. They will have administrative privileges to all areas of the application. They will be able to log in, manage, edit and maintain the application with duties such as setting up a test, ensuring and marking legal sign off as complete, confirming a test as ready to be assigned to a penetration tester, assigning a vulnerability found by the penetration tester to a user from the vulnerability management team, and generating reports and statistics based on a vulnerability's full cycle.

Penetration Tester:

The penetration tester will be the secondary user in this application. They will have standard privileges and access to the current test and current vulnerabilities page. They will have duties such as logging in, viewing current tests, adding a found vulnerability to the current vulnerabilities page as a result of the conducted penetration test based on the current test that was ready to be tested and assigned to the penetration tester, adding the vulnerability re-test information after a user from the vulnerability management team has marked the vulnerability as mitigated and then closing the vulnerability after re-test and confirming mitigation.

Vulnerability Management Team User:

The user from the vulnerability management team will also have standard privileges within the application. They will be a secondary user of the application. They will have duties such as logging in, viewing the vulnerability assigned to them, updating the progress of the vulnerabilities mitigation and flagging the vulnerability as mitigated.

Brief Use Cases

Use Cases for Test Setup

Login

Primary Actor:

Administrator

Preconditions:

The user has already created a valid account for the application.

Success Guarantee:

The user has successfully authenticated with the application.

Main Scenario:

1. The user will launch the web application.
2. The user will load the login page.
3. The user enters their username and password
4. These credentials are validated by the system through the database.
5. If the credentials are validated as correct, the user is granted access to the application

Alternative Flows:

Incorrect credentials were entered and the user is prompted to try enter credentials again.

Generate Test

Primary Actor:

Administrator

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can set up and manage a test.

Main Scenario:

1. The user will launch the vulnerability test page.
2. The user will click create new test.
3. The user enters the required information.
4. This information is then stored in the application database.
5. When complete the user can manage, view or edit the test information.
6. The test will be available on the current tests page.

Alternative Flows:

A test has already been created under the same name and the user will be asked if they would like to overwrite the old test.

Legal Requirements Sign Off

Primary Actor:

Administrator

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can set a flag if the legal requirements have been signed off and received.

Main Scenario:

1. The user will launch the vulnerability test page.
2. The user will click a current test they wish to edit.
3. The user sets flag to legal requirements signed off.
4. This information is then stored on the application database.
5. When complete the flag is set to yes and this information can be viewed on the test page.

Alternative Flows:

Legal requirements have not been signed off and the flag will be set to no.

Test Ready to Proceed

Primary Actor:

Administrator

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can set a flag that the test is ready to proceed and this notification will be sent to the penetration tester.

Main Scenario:

1. The user will launch the vulnerability test page.
2. The user will click a current test they wish to edit.
3. The user will set the flag as test ready.
4. This information is then stored in the application database.
5. Test is then confirmed ready to go ahead with the vulnerability test.

Alternative Flows:

Test is not ready and the flag will be set to not ready.

Use Cases for Vulnerability Testing Phase

Login

Primary Actor:

Penetration Tester

Preconditions:

The user has already created a valid account for the application.

Success Guarantee:

The user has successfully authenticated with the application.

Main Scenario:

1. The user will launch the web application.
2. The user will load the login page.
3. The user enters their username and password
4. These credentials are validated by the system through the database.
5. If the credentials are validated as correct, the user is granted access to the application

Alternative Flows:

Incorrect credentials were entered and the user is prompted to try enter credentials again.

View Current Tests

Primary Actor:

Penetration Tester.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can select a test to view.

Main Scenario:

1. The user will launch the vulnerability test page.
2. The user will click a current test they wish to view.
3. The test information will be available to the user and they can see items such as the test scope, test documents and test dates.

Alternative Flows:

Test cannot be viewed.

Add Vulnerability
Primary Actor:

Penetration Tester.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can add a found vulnerability to the current vulnerability page, this information is added after a penetration test has been performed.

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click add vulnerability.
3. The test ID will be entered so as to link the found vulnerability to a specific to test.
4. The vulnerability information will be entered
5. This information is then stored in the application database.
6. The vulnerability information can now be viewed on the current vulnerabilities page.

Alternative Flows:

Vulnerability had already been created, in this case the user will prompted if they would like to overwrite previously created vulnerability information.

Use Cases for Vulnerability Mitigation Phase

Assign Vulnerability to a User
Primary Actor:

Administrator

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can assign a created vulnerability to a user to be mitigated

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click a current vulnerability.
3. The user will click assign vulnerability.
4. The user will enter the information of the user they are assigning the vulnerability to.
5. This information is then stored in the application database.

6. The user receives a notification that they have been assigned a vulnerability to be mitigated.

Alternative Flows:

Vulnerability had already been assigned to a user.

User Login

Primary Actor:

Vulnerability management team user.

Preconditions:

The user has already created a valid account for the application.

Success Guarantee:

The user has successfully authenticated with the application.

Main Scenario:

1. The user will launch the web application.
2. The user will load the login page.
3. The user enters their username and password
4. These credentials are validated by the system through the database.
5. If the credentials are validated as correct, the user is granted access to the application

Alternative Flows:

Incorrect credentials were entered and the user is prompted to try enter credentials again.

View Assigned Vulnerability

Primary Actor:

Vulnerability management team user.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can view the vulnerability assigned to them.

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click the vulnerability they have been assigned.
3. The vulnerability information is displayed.

Alternative Flows:

Vulnerability not assigned to a user.

Update Vulnerability Progress

Primary Actor:

Vulnerability management team user.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can edit and update the vulnerability mitigation progress.

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click the vulnerability they have been assigned.
3. The vulnerability information is displayed.
4. The user can update the vulnerability mitigation progress and store information such as how the vulnerability is being mitigated.
5. This information is then stored in the database.
6. The vulnerability mitigation progress has been updated.

Alternative Flows:

Vulnerability not assigned to a user.

Vulnerability Mitigated

Primary Actor:

Vulnerability management team user.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can flag their assigned vulnerability as mitigated so as to notify the penetration tester that the vulnerability is ready to be re-tested to confirm mitigation is complete.

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click the assigned vulnerability they wish to edit.
3. The vulnerability information is displayed.
4. The user can update the vulnerability mitigation progress and flag it as mitigated.
5. This information is then stored in the database.
6. The vulnerability mitigation progress has been updated.

Alternative Flows:

Vulnerability is not mitigated and flagged as not mitigated.

Perform Vulnerability Re-test

Primary Actor:

Penetration Tester.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can add vulnerability re-test information to a current vulnerability that has been flagged as mitigated.

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click the vulnerability they wish to edit.
3. The re-test vulnerability phase information will be entered and stored separately to the already stored information.
4. This information is then stored in the application database.
5. The vulnerability re-test information can now be viewed on the current vulnerabilities page and is flagged as completely mitigated.

Alternative Flows:

Vulnerability re-test information had already been stored, in this case the user will prompted if they would like to overwrite previously created re-test vulnerability information.

Close Vulnerability

Primary Actor:

Penetration Tester.

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can close off a vulnerability.

Main Scenario:

1. The user will launch the current vulnerabilities page.
2. The user will click the vulnerability they wish to edit.
3. The vulnerability will be flagged as closed off.

4. This information is then stored in the application database.
5. The vulnerability information is now stored on the fixed vulnerabilities page.

Alternative Flows:

Vulnerability is not closed off.

Use Case for Generating Vulnerability Report

Generate Vulnerability Report

Primary Actor:

Administrator

Preconditions:

The user has logged in to the application.

Success Guarantee:

The user can generate a vulnerability report.

Main Scenario:

1. The user will launch the vulnerability report page.
2. The user will click create a new report.
3. The user enters the required information such as the vulnerability ID so that the report will be linked to the vulnerability it is based on.
4. This information is then stored on the application database.
5. When complete the user can manage, view or edit the report information.
6. The report will be available on the vulnerability report page.

Alternative Flows:

Vulnerability mitigation and re-test is not flagged as complete so a report cannot be generated.

FURPS

Functionality

This application must allow the full audit of a vulnerability from start to finish, this includes setting up a vulnerability test, the primary application purpose of storing a vulnerability and mitigating the vulnerability.

Usability

Users should remain logged in until they choose to log out of the application. The application should be easily navigable. The response time of the web application should be fast, efficient and satisfactory, however this will depend on the users internet connection.

Reliability

The application will be well tested and as reliable as possible, as organisations are relying on our application to fix their own.

Performance

The application should be high quality, quick and efficient.

Supportability

The application will be a web based application and will have the ability to run on any platform that has an internet connection.