Requirements Definition Document

Prepared By:	Alyssa Indriso
Organization:	River Otters
Date:	3/09/2020
Version:	*3*

Preface

This document was created for Mr. Resch and Mr. Munila to discuss our current requirements of the Mitigation Repository project. This defines the current plan of this project which is subject to change due to the agile work environment.

Here is a summary of the requirement definition document's version history:

Version 1:

This is the base creation of the document and this is the requirements planned as of sprint 1. This discussed the implementation of both the website, database, and the web server. This defines the current database design and the plan to handle the repository with user implementation.

Version 2:

Sentences were rewritten to provide a clear understanding of the material. Minor syntax errors and spelling errors have been revised. This document now demonstrates a clear perspective to the reader.

TABLE OF CONTENTS

1 Introduction

This project will allow users to view mitigations all in one location. Users who will use this system, previously would spend countless hours of researching mitigations trying to find the correct one. The Mitigation Repository plans to eliminate these barriers by allowing the user to add their own mitigations to the database for others to view. Not only will the user be able to search for mitigations, but users will be able to queue their own mitigations that may not be known to the public eye. This project will have both user and admin control since database monitoring will be necessary to ensure information is the most accurate.

Those already working in cybersecurity will find ease using this Mitigation Repository since it eliminates the need to install any software onto work equipment. The repository will be accessible through most web browsers and mobile devices. This allows for the ability to pull the repository on the go, if necessary. Users will be able to search for certain mitigations and view mitigations that fulfil the user specified requirements. It is a user-friendly approach to solving scouring the web for finding a certain mitigation since all mitigations in the repository will be stored so users can keep coming back to the webpage. Users will also be able to view newly added mitigations to ensure the most up to date mitigations.

2 Glossary

Entity Relationship Diagram: a diagram to visualize data relationship and dependencies between entities inside of a database.

Forking: the ability to clone a mitigation so users can work on mitigations to add different solutions to the same mitigation.

Repository: a central table for all storage locations. It is used to implement version control and can store multiple versions of a data record. There are countless types of mitigations in cyber security so using centralized data, it ensures that all of the data is being pulled and stored in one location.

Risk Mitigation: a strategy to prepare or decrease the presence of threats faced by data center. Usually illustrated in steps for a user to follow to reduce possible negative effects on a system's data center.

Security Control: a way to reduce or mitigate a risk to assets (ex: physical property, information data, hardware systems). These are classified by multiple criteria which is given a category (ex: directive, preventative, detective, etc.) and a type (ex: physical, administrative, or technical)

Use Case Diagram: a diagram to visualize the different roles throughout the system and how those roles interact with the system.

3 User requirements definition

As a user, our biggest priority is to ensure that this website is as user friendly as possible. The home page will prompt the option to either find or create a mitigation. Keeping this on the same page will avoid opening copious tabs to get one simple task done. If the user chooses to find a mitigation, to add ease to searching there will be the option to search by security control category, control type, or a simple text box search.

Once the user-specified search is complete, the website will present the user with all of the mitigations related to said search. These results will be displayed in order of creation to ensure the newest issues are brought to the user's attention first, since old issues tend to be resolved or become irrelevant after some time.

If a certain mitigation sparks the user's attention, clicking on the displayed mitigation will provide the user with more detailed information about the user-chosen mitigation. There the user will be able to read more about the mitigation and if something feels like it can be elaborated on, the user will have the option to add to the mitigation. This will allow for forking and multiple solutions for a certain mitigation.

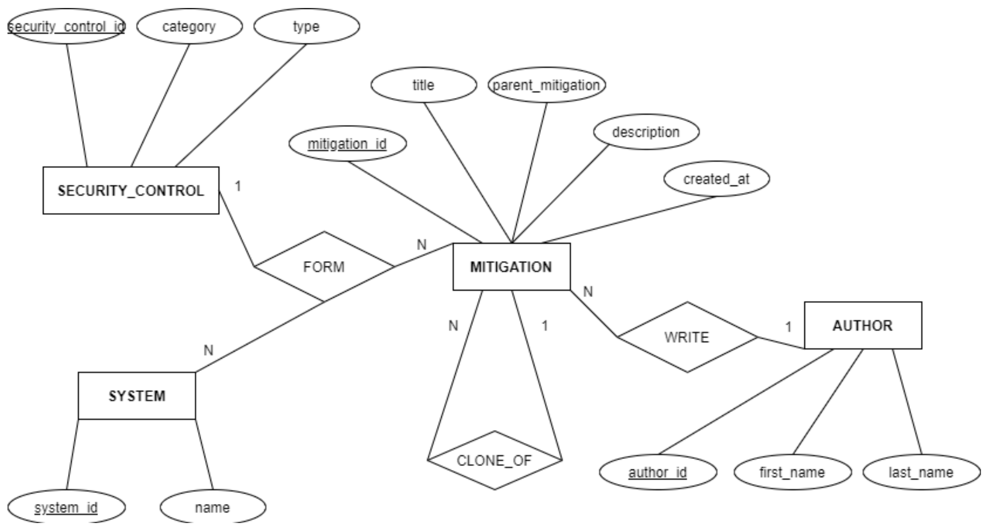
As previously stated, the user will be able to add mitigations to the repository. When adding a mitigation, it will ask the user for the name of the mitigation, the category and type that it classifies as, a brief description on how to assess the risk, and the user's name. Once the user submits the request, it will be stored for others to view.

When adding to the repository, the user will be able to branch off of previous mitigations or newly created mitigations. Even if the mitigation falls under a category that is not previously defined in the database, it will allow the user to add to the database.

4 System Architecture/Models

4.1 ER Diagram

The figure below is an Entity Relationship diagram of how the mitigation database will be implemented.



To elaborate on this diagram, each mitigation in the database will have an ID, a title, a parent mitigation, a description, and a date that it was created at. The purpose of storing the date when the mitigation was created is for when a user first enters the website. There will be a “recently added” piece of the webpage that will allow users to stay up to date with the most recently added mitigations. The purpose of having a parent mitigation tied within each mitigation is to allow forking and the ability to “clone” mitigations to provide extra detail to the user selected mitigation.

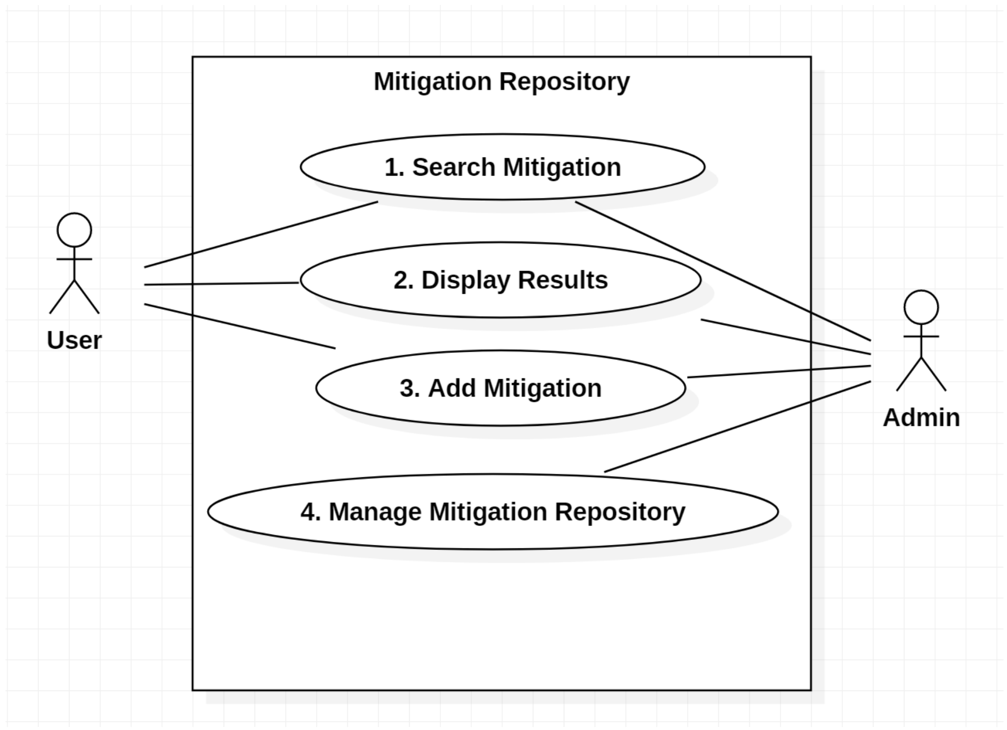
Next, there is the author who wrote the mitigation, to store exactly who is adding to the repository for admin purposes. This will store the author’s first name, last name, and a special ID to avoid any repeats of names.

Mitigations fall into different security controls and this will help keep the database clean and easy to navigate. Each security control will have an ID, category, and type. For more information about security categories and type, please reference the glossary for explanation. By defining the security controls with an ID, category, and type, this will make searching easier for the user to search mitigations based on select information.

The purpose for storing the system in the database is so the mitigations that the user found can be tracked. Software systems run into different issues, therefore, they should be treated as such in the repository.

4.2 Use Case Diagram

The figure below is a Use Case Diagram illustrating the roles involved in using this repository.



Users will be able to search mitigations, display all mitigations, and add mitigations to the repository. Since there will be administrative controls included, the admin will oversee any changes in the repository. This will be done in the backend of the project (the database) and will allow the admin to insert, select, and update all mitigations in the repository.

5 System requirements specification

This is where an advanced technical breakdown will commence to ensure the mitigation repository runs with the best possible outcome.

5.1 Mitigation Search breakdown

The user will be making search calls by category, type, or general search. With this in mind, database procedures predefined in the database will be required to implement the overall search function. This will tie in with the webpage with ease and allow for less stress on the web server.

5.2 Mitigation Display breakdown

When the user makes a selection of a certain mitigation, this will make a database procedure call, as well as carry over, all of the correct data. The web page will display the mitigation data, thereby giving the user a more detailed description.

5.3 Administrative Control

To ensure that the mitigation repository is the most accurate, the administrative controls modify the repository for possible spelling errors and remove any invalid mitigation. This allows for repository management and prevents regular users from modifying any of the data.

6 System evolution

Moving forward, this project will have to be moved to a different web server since the web server used is provided by a member of the Computer Science Department at Rowan University. A server transfer would be required, but all of the code pertaining to this project will be contained on GitHub. The link to the project on GitHub will be posted in the Appendix for later viewing. Since the database will be forever growing, the project could data capacity. The need to upgrade to a larger server to handle the extra stress and avoid server lag would be necessary.

The designers of the project have taken into consideration the growing database table and have allowed for the user to add new categories if they arise. Community driven databases can become unmonitored and allow for technical issues to occur. Because the system is an ever-growing database, it requires an extra hand to modify any issues that arise.

7 Appendices/References

Otters, R. (n.d.). Mitigations Repository. Retrieved from https://github.com/anttesoriero/MitigationsRepository