**Data Structures Workspace Security Document**

* Future developers and users interested in the vulnerability/security side of this project can use this document to understand specific areas in the project or development environment which are prone to intrusion. We use this document to specify what can go wrong with regards to security, how we have fixed these issues, and how future development should take consideration into these concerns. We will systematically lay out each security issue, elaborate the groups effected, and the potential risk this could create. After this introduction section we will lay out our attempts to resolve this issue, and we will offer a final section on what future issues/vulnerabilities should be considered.

Contents

[Vulnerabilities/Issues: 1](#_Toc100938341)

[GitHub Pull Requests & Actions 1](#_Toc100938342)

[Introduction 1](#_Toc100938343)

[Resolution 1](#_Toc100938344)

[Eclipse Secure Storage Passwords 1](#_Toc100938345)

[Introduction 1](#_Toc100938346)

[Resolution 1](#_Toc100938347)

[Input Data Manipulation 2](#_Toc100938348)

[Introduction 2](#_Toc100938349)

[Resolution 2](#_Toc100938350)

# Vulnerabilities/Issues:

## GitHub Pull Requests & Actions

### Introduction

* With the visibility our project being set to public or for a specified demographic of users (students, faculty, and individuals eager to learn more about data structures) the possibility for malicious pull requests and changes to our code is prevalent. Ensuring the security of all future users contains keeping safe guard of privileges which can allow malicious authors to push code with the intent of causing a vulnerability.

### Resolution

* We make use of [CodeQL](https://codeql.github.com/), an automated security scanning tool developed by GitHub to scan for vulnerabilities across the target repository. With such a tool at our disposal we can automate the scanning process of pull requests or actions by ensuring that all actions/ pull requests made by developers and contributors pass our series of security tests.

## Eclipse Secure Storage Passwords

### Introduction

* Development in the eclipse workspace links with GitHub, a remote git version control system which enables collaboration among all the various developers in your team. Eclipse has a built in eGit extension which is installed by default for any new version of the IDE. Configuration of eGit requires you to provide your GitHub account username and password which can be saved as a hashed key-value pair in Eclipse’s Secure Storage Vault. These passwords are saved in the workspace .metadata file, and scenarios can occur where this file is accidentally pushed to the repository. This may not be of concern when the repository is private and shared among collaborators who are trusted, but since the use case audience will be primarily public individuals this presents a vulnerability. These hashed passwords can be decrypted and [instructions](https://stackoverflow.com/questions/13100747/eclipse-password-recovery) have been lain out by other developers in hopes of decrypting this information.

### Resolution

* Explicitly make use of authorization tokens created in GitHub to alleviate the number of opportunities where users need to enter in their credentials.
* Ensure the .metadata folder in the workspace directory does not contain any hashed passwords before pushing changes to the GitHub repository.
* Do not track version control changes for the metadata folder. This option is the one adopted as the final resolution for this issue and eliminates any possibility for such a vulnerability.

## Input Data Manipulation

### Introduction

* As many of our methods handle the input of various data types the possibility for attackers to gain unintended information through control the format, and structure of input data. An example stems from a method that take an integer input such as the index or value of a container/data structure. If there are not checks in place for values outside of the container/data structure we open the possibility for manipulation of the system’s memory.

### Resolution

* We introduce the methodology of tightly checking for all possible edge/boundary conditions in all our methods, and as such there is little possibility for such a vulnerability to be present. Even still, if the attacker was able to successfully achieve this exploit they wouldn’t be receiving any sensitive information since the workspace must be cloned as a local copy to the user’s system.