|  |  |
| --- | --- |
| Name Of The Student | Jaganathan G |
| Internship Project Topic | TCS iON RIO-125: Application of Static Application Security Testing (SAST) Tools – Find Defects in Insecure Web-based Applications |
| Name of the Organization | TCS iON |
| Name of the Industry Mentor | Uma Devi |
| Name of the Institute | Government College of Engineering, Bodinayakkanur, Theni – 625583 |

|  |  |  |
| --- | --- | --- |
| Date | Day # | Hours Spent |
| 20-05-2023 | DAY 30 | 5 Hours |
| Activities done during the day:   1. Browsed the internet to study about topics. 2. Scanned an insecure application named “WebGoat-2023”. The scan result contains.  * Some Critical issues are found: * Code Smells * Define a constant instead of duplicating this literal "secQuestion0" 7 times.   **File Name:** Make Sure this file name for AccountVerificationHelper.java  **Description:**  Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.  On the other hand, constants can be referenced from many places, but only need to be updated in a single place.  **Explanation with Code snippets:**  src/main/java/org/owasp/webgoat/lessons/authbypass/AccountVerificationHelper.java    **Risk / Undesirable impact:**   1. Code Duplication and Maintenance: The code duplicates the string literals "secQuestion0" and "secQuestion1" multiple times. This can lead to code duplication, making it harder to maintain and update the code in the future. If there is a need to modify the string literals, you would have to make changes in multiple places, increasing the likelihood of errors and inconsistencies. 2. Lack of Readability and Reusability: Repeating the same string literals multiple times makes the code less readable and reduces its reusability. It is best practice to use constants or variables to represent repetitive or reusable values to enhance code clarity and maintainability.   **Recommendations**  Define Constants for String Literals: Instead of duplicating the string literals, define constants for them.  Use Constants in Code: Replace the duplicated string literals with the defined constants:  userSecQuestions.put(SEC\_QUESTION\_0, "Dr. Watson");  userSecQuestions.put(SEC\_QUESTION\_1, "Baker Street");  This way, you avoid code duplication, improve code maintainability, and make the code more readable and reusable.  **Solution:**  private static final String ACTION\_1 = "action1"; // Compliant  public void run() {  prepare(ACTION\_1); // Compliant  execute(ACTION\_1);  release(ACTION\_1);  } Move constants defined in this interface to another class or enum.File Name: Make Sure this file name for SolutionConstants.java **Description:**  The constant interface pattern is a poor use of interfaces.  That a class uses some constants internally is an implementation detail.  Implementing a constant interface causes this implementation detail to leak into the class’s exported API. It is of no consequence to the users of a class that the class implements a constant interface. In fact, it may even confuse them. Worse, it represents a commitment: if in a future release the class is modified so that it no longer needs to use the constants, it still must implement the interface to ensure binary compatibility. If a nonfinal class implements a constant interface,  all of its subclasses will have their namespaces polluted by the constants in the interface.  **Explanation with Code snippets:** src/main/java/org/owasp/webgoat/lessons/challenges/SolutionConstants.java   **Risk / Undesirable impact:**  **Violation of Single Responsibility Principle:** The SolutionConstants interface is responsible for defining constants, which is a different concern compared to the purpose of an interface. Interfaces are typically used to define contracts or behavior, and mixing in constant definitions can lead to a violation of the Single Responsibility Principle. This can result in code that is less maintainable and harder to understand.  **Recommendations**  Separate Constants into a Dedicated Class or Enum: Move the constants defined in the SolutionConstants interface to a dedicated class or enum that focuses solely on holding constants. This promotes better code organization and adheres to the principle of separation of concerns.  **Solution:**  Create a separate class or enum to hold the constants. Let's assume you create a class named Constants:  public final class Constants {  public static final String CONSTANT\_NAME\_1 = "value1";  public static final int CONSTANT\_NAME\_2 = 42;  // Add more constants as needed  }  Move the constants from the SolutionConstants interface to the newly created Constants class:  public final class Constants {  public static final String CONSTANT\_NAME\_1 = "value1";  public static final int CONSTANT\_NAME\_2 = 42;  // Move other constants from SolutionConstants here  }   1. Visited the digital discussion room and checked for any new announcements. 2. Finally, I wrote the Activity Report | | |