|  |  |
| --- | --- |
| 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 |
| 18-05-2023 | DAY 28 | 3 Hours |
| Activities done during the day:   1. Scanned an insecure application named “WebGoat-2023”. The scan result contains.  * Some Critical issues are found: * Code Smells * Remove this 'public' modifier.   **File Name:** Make Sure this file name for CSRFIntegrationTest.java  **Description:**  JUnit5 is more tolerant regarding the visibilities of Test classes than JUnit4, which required everything to be public.  **Explanation with Code snippets:**  src/it/java/org/owasp/webgoat/CSRFIntegrationTest.java    **Risk / Undesirable impact:**  **Weakened Access Control**: By removing the public modifier from the class, the visibility of the class is reduced, which can lead to weakened access control. Depending on the code's context and requirements, limiting the visibility of a class may restrict its accessibility and impact the overall functionality or interoperability of the codebase.  **Recommendations:**  Consider Codebase Design and Access Control: Before removing the public modifier from a class, carefully evaluate the codebase's design and access control requirements. Ensure that the class's visibility aligns with the intended use and accessibility within the codebase. If the class is intentionally designed to be accessed publicly, removing the public modifier may not be necessary.  **Solution**:  To remove the public modifier from the class declaration,  class CSRFIntegrationTest extends IntegrationTest {  // Class implementation  // ...  } Use already-defined constant 'UTF8' instead of duplicating its value here. **File Name:** Make Sure this file name for MvcConfiguration.java  **Description:**  String literals should not be duplicated  **Explanation with Code snippets:**  src/main/java/org/owasp/webgoat/container/MvcConfiguration.java    **Risk / Undesirable impact:**   1. Hardcoded Encoding: The code sets the default encoding for the pluginMessages object to "UTF-8". While "UTF-8" is a commonly used encoding, hardcoding it in the code may introduce issues if the encoding needs to be changed in the future. If the application requires a different encoding, modifying the code in multiple places can be error-prone and time-consuming. 2. Fixed Resource Location: The code sets the basenames for the message resources to "i18n/WebGoatLabels". Hardcoding the resource location limits the flexibility and reusability of the code. If the message resources need to be loaded from a different location or if multiple resource locations need to be used, modifying the code becomes necessary.   **Recommendations:**   1. Use Constant for Encoding: Instead of hardcoding the encoding value, utilize an already-defined constant for "UTF-8". In Java, the constant StandardCharsets.UTF\_8 represents the UTF-8 encoding. 2. Externalize Resource Locations: Instead of hardcoding the resource location as "i18n/WebGoatLabels", consider externalizing the resource location to a configuration file or property. This provides flexibility in configuring the resource location without modifying the code directly.   **Solution**:  PluginMessages pluginMessages = new PluginMessages(messages, language, resourcePatternResolver);  pluginMessages.setDefaultEncoding(StandardCharsets.UTF\_8.name());  pluginMessages.setBasenames("i18n/WebGoatLabels");  pluginMessages.setFallbackToSystemLocale(false);  return pluginMessages; | | |