|  |  |
| --- | --- |
| 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 |
| 16-05-2023 | DAY 26 | 3 Hours |
| Activities done during the day:   1. 1. Browsed the internet to study about topics. 2. 2. Scanned an insecure application named “WebGoat-2023”. The scan result contains.  * Some Critical issues are found: * **Vulnerabilities**    + - Don't use the default "PasswordEncoder" relying on plain-text.   **File Name:** Make Sure this file name for WebSecurityConfig.java  **Description:** A user password should never be stored in clear-text, instead a hash should be produced from it using a secure algorithm:   * not vulnerable to brute force attacks. * not vulnerable to collision attacks (see rule s4790). * and a salt should be added to the password to lower the risk of rainbow table attacks (see rule s2053).   This rule raises an issue when a password is stored in clear-text or with a hash algorithm vulnerable to bruce force attacks. These algorithms, like md5 or SHA-family functions are fast to compute the hash and therefore brute force attacks are possible (it’s easier to exhaust the entire space of all possible passwords) especially with hardware like GPU, FPGA or ASIC. Modern password hashing algorithms such as bcrypt, PBKDF2 or argon2 are recommended.    **Explanation with Code snippets:**  src/main/java/org/owasp/webgoat/webwolf/WebSecurityConfig.java    **Risk / Undesirable impact:**  A cross-site request forgery (CSRF) attack occurs when a trusted user of a web application can be forced, by an attacker, to perform sensitive actions that he didn’t intend, such as updating his profile or sending a message, more generally anything that can change the state of the application.  The attacker can trick the user/victim to click on a link, corresponding to the privileged action, or to visit a malicious web site that embeds a hidden web request and as web browsers automatically include cookies, the actions can be authenticated and sensitive.  **Recommendations:**   * The web application uses cookies to authenticate users. * There exist sensitive operations in the web application that can be performed when the user is authenticated. * The state / resources of the web application can be modified by doing HTTP POST or HTTP DELETE requests for example.     **Solution**  @EnableWebSecurity  public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  @Override  protected void configure(HttpSecurity http) throws Exception {  http.csrf().disable(); // Sensitive: csrf protection is entirely disabled  // or  http.csrf().ignoringAntMatchers("/route/"); // Sensitive: csrf protection is disabled for specific routes  }  } The JWT signature (JWS) should be verified before using this token. **File Name:** JWTRefreshEndpoint.java  **Description:** Hard-coded passwords are security-sensitive  **Explanation with Code snippets:**  src/main/java/org/owasp/webgoat/lessons/jwt/JWTRefreshEndpoint.java    **Risk / Undesirable impact:**  It is easy to extract strings from an application source code or binary, passwords should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.  Passwords should be stored outside of the code in a configuration file, a database, or a password management service.  This rule flags instances of hard-coded passwords used in database and LDAP connections. It looks for hard-coded passwords in connection strings, and for variable names that match any of the patterns from the provided list.  **Recommendations:**   * The password allows access to a sensitive component like a database, a file storage, an API, or a service. * The password is used in production environments. * Application re-distribution is required before updating the password.   **Solution** String username = "steve";  String password = "blue";  Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/test?" +  "user=" + uname + "&password=" + password); // Sensitive | | |