FB.IIL\_PREPARE\_STATEMENT\_IN\_LOOP

Метод вызывает Connection.prepareStatement внутри цикла, передавая постоянные аргументы. Если подготовленный оператор должен выполняться несколько раз, нет смысла создавать его заново для каждой итерации цикла. Следует переместить этот вызов за пределы цикла.

Таблица №1 – Основные характеристики детектора FB.IIL\_PREPARE\_STATEMENT\_IN\_LOOP

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Performance Issue |
| CWE | --- |
| CVE | --- |
| Возможные последствия | * Многократное создание PreparedStatement * Снижение производительности БД * Увеличение нагрузки на сервер |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  String[] names = {"Alice", "Bob", "Charlie"};    try (Connection conn = DriverManager.getConnection("jdbc:mock:test")) {  // Некорректно: подготовка statement в цикле  for (String name : names) {  PreparedStatement stmt = conn.prepareStatement(  "INSERT INTO users(name) VALUES(?)");  stmt.setString(1, name);  stmt.executeUpdate();  stmt.close();  }  } catch (SQLException e) {  e.printStackTrace();  }  } | public static void correctTest() {  String[] names = {"Alice", "Bob", "Charlie"};    try (Connection conn = DriverManager.getConnection("jdbc:mock:test");  PreparedStatement stmt = conn.prepareStatement(  "INSERT INTO users(name) VALUES(?)")) {    // Корректно: подготовка statement перед циклом  for (String name : names) {  stmt.setString(1, name);  stmt.executeUpdate();  }  } catch (SQLException e) {  e.printStackTrace();  }  } |