HANDLE\_LEAK.CLOSEABLE.EXCEPTION

Выявляет проблемы утечек ресурсов с интерфейсом Closeable и некорректной обработки исключений, связанных с закрытием ресурсов.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Неправильное закрытие ресурсов может привести к утечке файловых дескрипторов, сокетов и других системных ресурсов, что в свою очередь ухудшает стабильность и производительность приложения. |
| Комментарий | Рекомендуется использовать конструкции try-with-resources, чтобы гарантировать корректное закрытие Closeable-объектов независимо от наличия исключений. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  BufferedReader reader = null;  try {  reader = new BufferedReader(new FileReader("test.txt"));  String line = reader.readLine(); // Возможное исключение  System.out.println(line);  } catch (IOException e) {  System.err.println("Ошибка при чтении: " + e.getMessage());  } finally {  // закрытие без учёта исключения в close()  try {  if (reader != null) {  reader.close(); // Может выбросить исключение, которое не обрабатывается  }  } catch (IOException e) {  // Исключение подавлено или не логируется — ресурс может утечь незаметно  System.err.println("Ошибка при закрытии ресурса: " + e.getMessage());  }  }  } | public static void correctTest() {  // try-with-resources обрабатывает исключения при закрытии  try (BufferedReader reader = new BufferedReader(new FileReader("test.txt"))) {  String line = reader.readLine();  System.out.println(line);  } catch (IOException e) {  System.err.println("Ошибка при работе с файлом: " + e.getMessage());  }  } |