DOUBLE\_CLOSE

Происходит дублирующий вызов системного вызова close в одном и том же ресурсе.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Average |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE415;CWE416;CWE672;CWE675 |
| CVE | — |
| Возможные последствия | Повторное закрытие ресурса может вызвать исключение, сбой программы или утечку ресурсов, если последующие операции с закрытым ресурсом ведут к ошибкам. В некоторых случаях это также может привести к повреждению данных или потере соединения. |
| Комментарий | Необходимо убедиться, что метод close() вызывается единожды для каждого ресурса, предпочтительно с использованием конструкции try-with-resources или явной проверки состояния перед закрытием. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() throws IOException {  FileInputStream fis = null;  try {  fis = new FileInputStream("test.txt");  // Читаем данные...  } finally {  if (fis != null) {  fis.close();  fis.close();  }  }  } | public static void correctTest() {  try (FileInputStream fis = new FileInputStream("test.txt")) {  // Читаем данные...  } catch (IOException e) {  throw new RuntimeException(e);  }  // Ресурс автоматически закрывается благодаря try-with-resources  } |

Методы устранения уязвимости при срабатывании детектора:

1. Используйте конструкцию try-with-resources (начиная с Java 7) для автоматического управления ресурсами, реализующими интерфейс AutoCloseable. Это гарантирует, что метод close() будет вызван ровно один раз при выходе из блока try.
2. Если try-with-resources использовать невозможно, убедитесь, что метод close() вызывается только один раз для каждого ресурса в блоке finally. Избегайте повторных вызовов close().