FB.OBL\_UNSATISFIED\_OBLIGATION

Этот метод может не выполнить очистку (закрытие, удаление) потока, объекта базы данных или другого ресурса, требующего явной операции очистки.

Как правило, если метод открывает поток или другой ресурс, метод должен использовать блок try/finally, чтобы гарантировать очистку потока или ресурса до возврата метода.

Этот шаблон ошибок по существу аналогичен шаблонам ошибок OS\_OPEN\_STREAM и ODR\_OPEN\_DATABASE\_RESOURCE, но основан на другой (и, мы надеемся, лучшей) технике статического анализа.

Таблица №1 – Основные характеристики детектора FB.OBL\_UNSATISFIED\_OBLIGATION

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Resource Leak / Unfulfilled Obligation |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Утечка файловых дескрипторов, отказ в обслуживании при достижении лимита; потеря данных при незакрытых потоках записи; блокировка ресурсов (файлов, сокетов, соединений БД); неконтролируемый рост потребления памяти. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // 1. Ресурс никогда не закрывается  FileInputStream fis = new FileInputStream("data.txt");  // 2. Закрытие только в happy path  BufferedReader br = null;  try {  br = new BufferedReader(new FileReader("config.ini"));  String line = br.readLine();  if (line.startsWith("DEBUG")) {  return; // Утечка ресурса!  }  br.close(); // Закрытие только здесь  } catch (IOException e) {  e.printStackTrace();  }  // 3. Небезопасный try-finally  OutputStream os = null;  try {  os = new FileOutputStream("out.bin");  os.write(42);  } finally {  if (os != null) { // Правильная проверка, но...  os.close(); // Может выбросить IOException  }  }  } | public static void correctTest() {  // 1. try-with-resources (Java 7+)  try (FileInputStream fis = new FileInputStream("data.txt");  BufferedReader br = new BufferedReader(new InputStreamReader(fis))) {  System.out.println(br.readLine());  } catch (IOException e) {  e.printStackTrace();  }  // 2. Явное управление с защитой от исключений  BufferedReader br = null;  try {  br = new BufferedReader(new FileReader("config.ini"));  String line = br.readLine();  if (line.startsWith("DEBUG")) {  return;  }  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  if (br != null) {  br.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  // 3. Использование IOUtils из Apache Commons  OutputStream os = null;  try {  os = new FileOutputStream("out.bin");  os.write(42);  } catch (IOException e) {  e.printStackTrace();  } finally {  org.apache.commons.io.IOUtils.closeQuietly(os); // Безопасное закрытие  }  } |