FB.LG\_LOST\_LOGGER\_DUE\_TO\_WEAK\_REFERENCE

OpenJDK представляет потенциальную несовместимость с java.util.logging.Logger. В частности, изменилось поведение java.util.logging.Logger. Вместо использования сильных ссылок теперь внутри системы используются слабые ссылки. Это разумное изменение, но, к сожалению, некоторый код использует старое поведение — при изменении конфигурации регистратора он просто удаляет ссылку на регистратор. Это означает, что сборщик мусора может освободить эту память, что в свою очередь значит, что конфигурация регистратора потеряна.

Таблица №1 – Основные характеристики детектора FB.LG\_LOST\_LOGGER\_DUE\_TO\_WEAK\_REFERENCE

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Logger Loss Due to Weak Reference |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Потеря лог-сообщений критически важных для диагностики; незарегистрированные события в системах аудита и мониторинга; усложнение отладки из-за неожиданного прекращения логирования; нарушение требований compliance в регулируемых отраслях |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public void testUnsafeLogging() {  Logger logger = unsafeLoggerRef.get();  if (logger != null) { // Логгер может быть собран GC  logger.info("Unsafe log message");  }  // Принудительная очистка для демонстрации  System.gc();    if (unsafeLoggerRef.get() == null) {  System.out.println("Logger was garbage collected!");  }  } | public void testSafeLogging() {  SAFE\_LOGGER.info("Safe log message");    // Гарантированно доступен  assert SAFE\_LOGGER != null : "Logger is null!";  } |