FB.STCAL\_INVOKE\_ON\_STATIC\_DATE\_FORMAT\_INSTANCE

Детектор определяет ошибку, в ходе которой DateFormats по своей сути небезопасны для многопоточного использования. Детектор обнаружил вызов экземпляра DateFormat, полученного через статическое поле. Это выглядит подозрительно.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE366 |
| CVE | — |
| Возможные последствия | Использование общего статического экземпляра DateFormat в многопоточной среде может привести к некорректному форматированию или разбору дат, поскольку DateFormat не является потокобезопасным. |
| Комментарий | Вместо статического DateFormat рекомендуется использовать ThreadLocal<DateFormat> для обеспечения потокобезопасности или перейти на java.time.format.DateTimeFormatter, который является потокобезопасным и предпочтителен в современных Java-приложениях. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| private static final DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  public static void incorrectTest() {  String date = dateFormat.format(new Date());  System.out.println(date);  } | private static final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");  public static void correctTest() {  String date = LocalDate.now().format(formatter);  System.out.println(date);  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Использование ThreadLocal<DateFormat>:** Вместо статического экземпляра DateFormat создайте ThreadLocal, который будет предоставлять уникальный экземпляр DateFormat для каждого потока. Это гарантирует, что каждый поток работает со своим собственным экземпляром, избегая проблем с конкурентным доступом.
2. **Переход на java.time.format.DateTimeFormatter:** В современных Java-приложениях рекомендуется использовать классы из пакета java.time, введенного в Java 8. Класс DateTimeFormatter является потокобезопасным и предоставляет более гибкие возможности для форматирования и разбора дат и времени.