FB.DM\_STRING\_CTOR

Детектор определяет ошибку, в ходе которой использование конструктора java.lang.String(String) тратит впустую память, поскольку созданный таким образом объект будет функционально неотличим от строки, переданной в качестве параметра. Просто используйте аргумент String напрямую.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Использование конструктора new String(String) создаёт новый объект строки в памяти, хотя переданный аргумент уже является строкой. Это приводит к избыточному расходу памяти и дополнительной нагрузке на сборщик мусора. |
| Комментарий | Рекомендуется использовать строку напрямую без создания нового экземпляра, так как строки в Java являются неизменяемыми (immutable) и кэшируются в пуле строк, что значительно оптимизирует использование памяти. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  String original = "Hello, world!";  String duplicate = new String(original);  System.out.println(duplicate);  } | public static void correctTest() {  String original = "Hello, world!";  String duplicate = original;  System.out.println(duplicate);  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Замените new String(existingString) на прямое присваивание:** Вместо создания нового объекта String с помощью конструктора, просто присвойте существующую строку новой переменной: String duplicate = original;.
2. **Проверьте весь код:** Найдите все места, где используется конструктор new String(String) и замените их на прямое присваивание.
3. **Избегайте ненужного дублирования строк:** Понимание того, что строки в Java являются неизменяемыми, помогает избежать ненужного создания новых строковых объектов. Если вам нужна копия строки для каких-либо модификаций, используйте методы, которые явно создают новую строку с измененным содержимым (например, substring(), replace()).
4. **Обратите внимание на производительность:** Хотя создание одного лишнего объекта String может не иметь большого влияния на производительность, в больших приложениях с частым дублированием строк это может привести к заметному увеличению потребления памяти и нагрузки на GC.