FB.DM\_GC

Детектор определяет ошибку, в ходе которой код явно вызывает сбор мусора. За исключением конкретного использования в бенчмаркинге, это очень сомнительно.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Явный вызов сборщика мусора (System.gc(), Runtime.getRuntime().gc()) не гарантирует немедленного освобождения памяти и может привести к неожиданным задержкам в работе приложения. |
| Комментарий | Рекомендуется избегать явных вызовов сборки мусора, так как JVM самостоятельно управляет памятью наиболее эффективным способом. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  // ...  // явное принудительное освобождение памяти  System.gc();  } | public static void correctTest() {  // Единственный оправданный случай —  // бенчмаркинг перед замером потребления памяти.  System.gc(); // Освобождаем память перед замером  long beforeMemory = Runtime.getRuntime().freeMemory();  byte[] array = new byte[10\_000\_000];  long afterMemory = Runtime.getRuntime().freeMemory();  System.out.println("Memory used: " + (beforeMemory - afterMemory) + " bytes");  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Удалите явные вызовы System.gc() и Runtime.getRuntime().gc():** В большинстве случаев просто удалите строки кода, которые явно вызывают сборку мусора.
2. **Доверьте управление памятью JVM:** JVM имеет сложные алгоритмы для автоматического управления памятью. Позвольте ей самостоятельно решать, когда и как выполнять сборку мусора.
3. **Сосредоточьтесь на эффективном использовании памяти:** Вместо того чтобы пытаться принудительно вызывать сборку мусора, сосредоточьтесь на написании кода, который эффективно использует память, например, избегайте создания ненужных объектов и освобождайте ресурсы (например, закрывайте потоки и соединения) своевременно.
4. **Проанализируйте причину вызова GC:** Если вы добавили явный вызов сборки мусора, пытаясь решить проблему с памятью (например, OutOfMemoryError), то истинное решение, скорее всего, заключается в устранении утечек памяти или увеличении размера кучи JVM, а не в принудительном вызове GC.
5. **Исключение для бенчмаркинга:** В очень редких случаях, при проведении точных измерений потребления памяти в бенчмарках, явный вызов System.gc() непосредственно перед измерением может быть оправдан для минимизации влияния предыдущей активности сборщика мусора. Однако даже в этом случае следует быть осторожным и понимать последствия.