FB.UR\_UNINIT\_READ\_CALLED\_FROM\_SUPER\_CONSTRUCTOR

Детектор определяет ошибку, в ходе которой метод вызывается в конструкторе суперкласса. На данный момент поля класса еще не инициализированы.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Вызов метода из конструктора суперкласса до инициализации полей подкласса может привести к работе с неинициализированными значениями, что может вызвать NullPointerException, некорректное поведение или нарушение инвариантов объекта. |
| Комментарий | Метод вызывается в конструкторе суперкласса, когда поля подкласса ещё не инициализированы. Следует избегать вызова методов, зависящих от состояния подкласса, в процессе его конструирования. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Parent {  public Parent() {  // вызов метода, который  // использует неинициализированные поля  init();  }  public void init() {  //  }  }  class Child extends Parent {  private int value = 10;  @Override  public void init() {  System.out.println(value);  }  }  } | public static void correctTest() {  class Parent {  public Parent() {  // Убираем вызов init() из конструктора  }  public void init() {  // ...  }  }  class Child extends Parent {  private int value = 10;  public Child() {  super();  init();  }  public void init() {  System.out.println(value);  }  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Переместите вызов метода, зависящего от состояния подкласса, в конструктор подкласса.** Вызывайте метод после того, как будут инициализированы поля подкласса.
2. **Используйте шаблон проектирования "шаблонный метод" (Template Method) с осторожностью.** Если суперкласс вызывает абстрактный метод, реализованный в подклассе, убедитесь, что подкласс корректно обрабатывает состояние до инициализации своих полей. Лучше избегать такой зависимости, если это возможно.
3. **Перепроектируйте иерархию классов, чтобы избежать вызова методов подкласса из конструктора суперкласса.** Рассмотрите возможность переноса логики инициализации в подкласс.
4. **Если метод в суперклассе должен выполнять какую-то общую инициализацию, сделайте его независимым от состояния подкласса.** Передавайте необходимые параметры вместо того, чтобы полагаться на поля подкласса.
5. **Используйте фабричные методы или статические конструкторы вместо публичных конструкторов, если требуется более сложная логика инициализации, зависящая от типа объекта.** Это может дать больше контроля над процессом создания объекта.