FB.NP\_STORE\_INTO\_NONNULL\_FIELD

Значение, которое может быть нулевым, сохраняется в поле, помеченном как @Nonnull.

Таблица №1 – Основные характеристики детектора FB.NP\_STORE\_INTO\_NONNULL\_FIELD

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Null pointer dereference |
| CWE | CWE-476 |
| CVE | --- |
| Возможные последствия | Возможное возникновение NullPointerException при выполнении определенных ветвей кода |
| Комментарий (опционально) | Проверьте логику ветвления и добавьте обработку null. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public void incorrectUsage() {  String potentiallyNullValue = getPotentiallyNullValue();  // Некорректно: сохранение null в поле, которое должно быть ненулевым  this.nonNullField = potentiallyNullValue;  } | public void correctUsage() {  String potentiallyNullValue = getPotentiallyNullValue();  // Корректно: проверка на null перед сохранением значения  if (potentiallyNullValue != null) {  this.nonNullField = potentiallyNullValue;  } else {  System.out.println("Ошибка: параметр равен null.");  }  } |

Методы устранения уязвимости при срабатывании детектора

1. Проверка на null: Перед сохранением значения в поле, которое должно быть ненулевым, необходимо выполнять проверку на null. Если значение равно null, следует либо выбрасывать исключение, либо предоставлять значение по умолчанию.
2. Использование аннотаций:Используйте аннотации, такие как @NonNull или аналогичные, чтобы указать, что поле не должно принимать значение null. Это позволит статическим анализаторам обнаруживать потенциальные ошибки.
3. Инициализация значений: Убедитесь, что все ненулевые поля инициализируются корректными значениями при создании объекта или перед использованием.