FB.BC\_BAD\_CAST\_TO\_CONCRETE\_COLLECTION

Детектор определяет ошибку, в ходе которой код преобразует абстрактную коллекцию (например, Collection, List или Set) в конкретную конкретную реализацию (например, ArrayList или HashSet). Это может быть неправильно и может сделать ваш код хрупким, поскольку в будущем будет сложнее переключиться на другие конкретные реализации. Если у вас нет для этого особой причины, просто используйте абстрактный класс коллекции.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Приведение абстрактной коллекции к конкретной реализации снижает гибкость кода и делает его более хрупким. |
| Комментарий | Рекомендуется работать с абстрактными типами коллекций (например, List, Set, Collection), чтобы сохранить гибкость кода. Приведение к конкретной реализации следует избегать, если в этом нет крайней необходимости, так как это затруднит переход на другие реализации в будущем. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class MyClass {  public void processCollection(Collection<?> collection) {  // Неправильное приведение к конкретной реализации  // Это может привести к ClassCastException, если collection не является ArrayList  ArrayList<?> arrayList = (ArrayList<?>) collection;  for (Object item : arrayList) {  System.out.println(item);  }  }  }  Collection<String> collection = new HashSet<>();  collection.add("apple");  collection.add("banana");  MyClass myClass = new MyClass();  myClass.processCollection(collection);  } | public static void correctTest() {  class MyClass {  public void processCollection(Collection<?> collection) {  // Просто перебирать коллекцию без приведения ее к абстрактному типу  // Это более безопасно, так как не зависит от типа коллекции  for (Object item : collection) {  System.out.println(item);  }  }  }  Collection<String> collection = new HashSet<>();  collection.add("apple");  collection.add("banana");  MyClass myClass = new MyClass();  myClass.processCollection(collection);  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Используйте абстрактные типы:** Объявляйте переменные, параметры методов и возвращаемые значения с использованием абстрактных интерфейсов коллекций (Collection, List, Set, Map) вместо конкретных реализаций (ArrayList, HashSet, HashMap и т.д.).
2. **Избегайте приведения к конкретным типам:** Если нет абсолютно веской причины, не приводите абстрактные коллекции к их конкретным реализациям. Работайте с общими интерфейсами.
3. **Пересмотрите требования:** Подумайте, действительно ли вам нужны специфические методы конкретной реализации. В большинстве случаев функциональности, предоставляемой абстрактными интерфейсами, достаточно.
4. **Создавайте экземпляры с абстрактными типами:** При создании новых коллекций присваивайте их переменным абстрактных типов (например, List<String> myList = new ArrayList<>();).