FB.BC\_UNCONFIRMED\_CAST

Детектор определяет ошибку, в ходе которой приведение не отмечено, и не все экземпляры типа, из которого происходит приведение, могут быть приведены к типу, к которому оно приводится. Убедитесь, что логика вашей программы гарантирует, что это приведение не завершится неудачей.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Если приведение типа выполняется без предварительной проверки, это может привести к ClassCastException во время выполнения, если объект не принадлежит к ожидаемому классу. |
| Комментарий | Перед приведением типа рекомендуется использовать оператор instanceof, чтобы убедиться, что объект действительно является экземпляром ожидаемого класса. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class MyClass {  public void processObject(Object obj) {  // Непроверенное приведение типа  // Может вызвать ClassCastException, если obj не является String  String str = (String) obj;  System.out.println(str.length());  }  }  Object obj = 42;  MyClass myClass = new MyClass();  myClass.processObject(obj); // Ошибка во время выполнения  } | public static void correctTest() {  class MyClass {  public void processObject(Object obj) {  // Проверяем, действительно ли obj является строкой  if (obj instanceof String) {  String str = (String) obj;  System.out.println(str.length());  } else {  System.out.println(false);  }  }  }  Object obj = 42;  MyClass myClass = new MyClass();  myClass.processObject(obj); // Безопасная обработка  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Используйте instanceof для проверки типа:** Перед выполнением приведения типа всегда проверяйте, является ли объект экземпляром ожидаемого класса с помощью оператора instanceof.
2. **Обрабатывайте случаи несовместимого типа:** Предусмотрите логику обработки ситуаций, когда объект не принадлежит к ожидаемому типу (например, вывод сообщения об ошибке, выполнение альтернативных действий).
3. **Пересмотрите логику программы:** Убедитесь, что логика вашей программы действительно гарантирует, что приведение всегда будет безопасным. Если это не так, необходимо добавить явную проверку типа.
4. **Используйте обобщения (Generics):** В случаях, когда это возможно, используйте обобщения для определения типа коллекции или объекта на этапе компиляции, что позволяет избежать необходимости явного приведения типа и связанных с ним рисков.