FB.EQ\_DOESNT\_OVERRIDE\_EQUALS

Детектор определяет ошибку, в ходе которой класс расширяет класс, который определяет метод равенства и добавляет поля, но не определяет сам метод равенства. Таким образом, равенство экземпляров этого класса будет игнорировать идентичность подкласса и добавленных полей. Убедитесь, что это именно то, что задумано, и что вам не нужно переопределять метод равенства. Даже если вам не нужно переопределять метод равенства, все равно рассмотрите возможность его переопределения, чтобы задокументировать тот факт, что метод равенства для подкласса просто возвращает результат вызова super.equals(o).

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE595 |
| CVE | — |
| Возможные последствия | Объекты подкласса могут считаться равными, даже если их уникальные поля различаются, что может привести к ошибкам в коллекциях (Set, Map) и логике сравнения. |
| Комментарий | Если подкласс вводит новые поля, стоит переопределить equals() (и hashCode()) для корректного сравнения. Если поведение родительского equals() приемлемо, лучше явно переопределить его и вызвать super.equals(o) для документирования намерений. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Parent {  private final int id;  public Parent(int id) {  [this.id](http://this.id) = id;  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || getClass() != obj.getClass()) return false;  Parent parent = (Parent) obj;  return id == [parent.id](http://parent.id);  }  }  class Child extends Parent {  // Новое поле, но equals() не переопределен  private final String name;  public Child(int id, String name) {  super(id);  this.name = name;  }  }  Child c1 = new Child(1, "Alice");  Child c2 = new Child(1, "Bob");  // сравнивает только id, игнорируя name  System.out.println(c1.equals(c2));  } | public static void correctTest() {  class Parent {  private final int id;  public Parent(int id) {  [this.id](http://this.id) = id;  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || getClass() != obj.getClass()) return false;  Parent parent = (Parent) obj;  return id == [parent.id](http://parent.id);  }  }  class Child extends Parent {  private final String name;  public Child(int id, String name) {  super(id);  this.name = name;  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (!(obj instanceof Child)) return false;  if (!super.equals(obj)) return false;  Child child = (Child) obj;  return name.equals(child.name);  }  }  Child c1 = new Child(1, "Alice");  Child c2 = new Child(1, "Bob");  System.out.println(c1.equals(c2));  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Переопределите метод equals() в подклассе:** Если новые поля, добавленные в подклассе, должны влиять на определение равенства объектов, переопределите метод equals() в подклассе. В своей реализации сначала вызовите super.equals(o) для проверки равенства унаследованных полей, а затем добавьте логику сравнения новых полей. Не забудьте также переопределить метод hashCode() в соответствии с новыми правилами равенства.
2. **Явно переопределите equals() и вызовите super.equals(o):** Если поведение метода equals() родительского класса полностью соответствует требованиям к сравнению объектов подкласса (т.е., новые поля не должны влиять на равенство), явно переопределите метод equals() в подклассе и просто верните результат вызова super.equals(o). Это документирует ваше намерение и предотвращает случайное использование метода equals() из Object, если родительский класс не переопределил его. Аналогично поступите с методом hashCode().
3. **Соблюдайте контракт equals() и hashCode():** При переопределении equals(), убедитесь, что ваша реализация следует всем правилам контракта equals() (рефлексивность, симметричность, транзитивность, консистентность, сравнение с null). Также помните, что если два объекта равны согласно equals(), их hashCode() должен возвращать одно и то же значение.