FB.EQ\_UNUSUAL

Детектор определяет ошибку, в ходе которой класс не использует никаких шаблонов, которые мы распознаем для проверки совместимости типа аргумента с типом объекта this. Возможно, в этом коде нет ничего плохого, но его стоит просмотреть.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE595 |
| CVE | — |
| Возможные последствия | Метод equals() может некорректно обрабатывать сравнение объектов разных типов, что может привести к неожиданным ошибкам, например, ClassCastException или ложным совпадениям. |
| Комментарий | Обычно equals() начинается с if (!(obj instanceof ClassName)) return false;. Если код не использует этот или похожий шаблон, проверьте, действительно ли сравнение работает так, как задумано. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Person {  private final String name;  private final int age;  public Person(String name, int age) {  this.name = name;  this.age = age;  }  @Override  public boolean equals(Object obj) {  // нет стандартной проверки на совместимость типов  try {  Person p = (Person) obj;  return this.name.equals(p.name) && this.age == p.age;  } catch (ClassCastException e) {  return false;  }  }  }  Person p1 = new Person("Alice", 25);  Person p2 = new Person("Alice", 25);  Person p3 = new Person("Bob", 30);  System.out.println(p1.equals(p2)); // true (но реализация equals странная)  System.out.println(p1.equals(p3)); // false  System.out.println(p1.equals("Alice")); // выброс ClassCastException  } | public static void correctTest() {  class Person {  private final String name;  private final int age;  public Person(String name, int age) {  this.name = name;  this.age = age;  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || getClass() != obj.getClass()) return false;  Person person = (Person) obj;  return age == person.age && name.equals(person.name);  }  @Override  public int hashCode() {  return 31 \* name.hashCode() + age;  }  }  Person p1 = new Person("Alice", 25);  Person p2 = new Person("Alice", 25);  Person p3 = new Person("Bob", 30);  System.out.println(p1.equals(p2)); // true  System.out.println(p1.equals(p3)); // false  System.out.println(p1.equals("Alice")); // false (без исключений)  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Добавьте стандартную проверку типа:** В начале метода equals() добавьте проверку, является ли переданный объект obj экземпляром того же класса, используя if (obj == null || getClass() != obj.getClass()) return false; или if (!(obj instanceof ClassName)) return false;. Выберите подходящий вариант в зависимости от того, должно ли сравнение быть строгим по классу или допускать сравнение с подклассами.
2. **Проверьте существующую логику:** Если вы используете нестандартный способ проверки типа, тщательно проанализируйте, корректно ли обрабатываются все возможные типы входных объектов и не нарушается ли контракт метода equals().
3. **Избегайте try-catch для проверки типа:** Использование try-catch для обработки ClassCastException вместо явной проверки типа является плохой практикой и может скрыть другие потенциальные ошибки.
4. **Соблюдайте контракт equals():** Убедитесь, что ваша реализация метода equals() следует всем правилам контракта: рефлексивность, симметричность, транзитивность, консистентность и сравнение с null.
5. **Переопределите hashCode():** Если вы переопределяете equals(), обязательно переопределите и метод hashCode() таким образом, чтобы для двух равных объектов hashCode() возвращал одно и то же значение.