**FB.NM\_LCASE\_HASHCODE**

Этот класс определяет метод hashcode(). Этот метод не переопределяет метод hashCode() в java.lang.Object, что, вероятно, и было задумано.

Таблица №1 – Основные характеристики детектора **FB.NM\_LCASE\_HASHCODE**

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Naming Convention |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Некорректная работа хэш-коллекций |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| static class BadHashCode {  private int id;  private String name;    // Неправильное имя метода  public int hashcode() { // FB.NM\_LCASE\_HASHCODE  return id;  }  } | static class GoodHashCode {  private int id;  private String name;    // Правильное имя метода  @Override  public int hashCode() {  return Objects.hash(id, name);  }    @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  GoodHashCode that = (GoodHashCode) o;  return id == that.id && Objects.equals(name, that.name);  }  } |

Методы устранения уязвимости при срабатывании детектора

1. Всегда переопределяйте hashCode() вместе с equals().
2. Используйте одинаковые поля в equals() и hashCode().
3. Генерируйте методы автоматически с помощью IDE (Alt+Insert в IntelliJ IDEA).
4. Следуйте контракту hashCode(): Если x.equals(y) == true, то x.hashCode() == y.hashCode(); Обратное не обязательно должно быть верным; Многократные вызовы должны возвращать одинаковое значение (если объект не изменялся)
5. Для неизменяемых объектов кэшируйте значение хэш-кода.