FB.DMI\_NONSERIALIZABLE\_OBJECT\_WRITTEN

Детектор определяет ошибку, в ходе которой код передает несериализуемый объект методу ObjectOutput.writeObject. Если объект действительно несериализуем, возникнет ошибка.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE440 |
| CVE | — |
| Возможные последствия | Попытка сериализовать несериализуемый объект приведет к java.io.NotSerializableException. |
| Комментарий | Перед тем как передавать объект для сериализации, убедитесь, что класс объекта реализует интерфейс Serializable. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  // передача несериализуемого объекта в writeObject  class NonSerializableClass {  private int id;  public NonSerializableClass(int id) {  [this.id](http://this.id) = id;  }  public int getId() {  return id;  }  }  NonSerializableClass obj = new NonSerializableClass(1);  try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("object.dat"))) {  oos.writeObject(obj); // Попытка сериализовать несериализуемый объект  } catch (IOException e) {  System.out.println("Ошибка при сериализации: " + e.getMessage());  }  } | public static void correctTest() {  // объект должен быть сериализуемым  class SerializableClass implements Serializable {  private int id;  public SerializableClass(int id) {  [this.id](http://this.id) = id;  }  public int getId() {  return id;  }  }  SerializableClass obj = new SerializableClass(1);  try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("object.dat"))) {  oos.writeObject(obj); // Сериализация сериализуемого объекта  } catch (IOException e) {  System.out.println("Ошибка при сериализации: " + e.getMessage());  }  } |