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());  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Реализуйте интерфейс Serializable:** Если объекты данного класса должны быть сериализованы, убедитесь, что класс реализует интерфейс java.io.Serializable. Обратите внимание, что все поля класса также должны быть сериализуемыми (либо реализовывать Serializable, либо быть примитивными типами). Если поле ссылается на несериализуемый объект, его необходимо пометить как transient, и при десериализации его значение будет null (или потребуется предоставить логику для его восстановления).
2. **Используйте transient для несериализуемых полей:** Если класс должен быть сериализуемым, но некоторые его поля не могут или не должны быть сериализованы, пометьте эти поля ключевым словом transient. Значение transient полей не будет сохранено при сериализации и будет инициализировано значением по умолчанию (null для объектов, 0 для чисел, false для boolean) при десериализации.
3. **Используйте другие способы передачи данных:** Если объект не предназначен для сериализации, рассмотрите альтернативные способы передачи его состояния (например, запись в текстовый файл, базу данных или использование другого формата сериализации, такого как JSON или XML, с соответствующими библиотеками).
4. **Проверьте иерархию классов:** Если класс наследуется от другого класса, убедитесь, что все нестатические и нетранзитные поля в иерархии классов являются сериализуемыми.
5. **Внимательно относитесь к сторонним библиотекам:** При использовании объектов из сторонних библиотек, которые вы пытаетесь сериализовать, убедитесь, что эти классы реализуют Serializable. Если нет, вам может потребоваться создать класс-обертку, который реализует Serializable и содержит необходимые данные из несериализуемого объекта.