FB.TQ\_NEVER\_VALUE\_USED\_WHERE\_ALWAYS\_REQUIRED

Значение, указанное как не содержащее аннотацию квалификатора типа, гарантированно будет использовано в месте или местах, где требуется, чтобы значение содержало эту аннотацию.

Точнее, значение, помеченное квалификатором типа, указывающим, when=NEVER, гарантированно достигнет использования или использования, где тот же квалификатор типа указывает, when=ALWAYS.

Таблица №1 – Основные характеристики детектора FB.TQ\_NEVER\_VALUE\_USED\_WHERE\_ALWAYS\_REQUIRED

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Invalid Value Usage |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Логические ошибки в выполнении программы; некорректная обработка граничных условий; нарушение контрактов методов при передаче недопустимых значений; потенциальные уязвимости безопасности при обработке специальных значений |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // 1. Передача заведомо недопустимого значения  processStatus("INVALID"); // @NeverValue в @AlwaysRequired контексте  // 2. Использование null там, где требуется non-null  String name = null;  saveUserName(name); // @NeverValue (null) в @AlwaysRequired (non-null)  // 3. Выход за границы допустимых значений  int retries = -1; // Никогда не должно быть отрицательным  performOperationWithRetry(retries);  } | public static void correctTest() {  // 1. Проверка допустимых значений  String status = getStatus();  if (isValidStatus(status)) {  processStatus(status);  }  // 2. Защита от null  String name = getName();  if (name != null) {  saveUserName(name);  }  // 3. Валидация входных параметров  int retries = getRetryCount();  if (retries >= 0 && retries <= MAX\_RETRIES) {  performOperationWithRetry(retries);  }  } |