FB.RV\_DONT\_JUST\_NULL\_CHECK\_READLINE

Детектор определяет ошибку, в ходе которой значение, возвращаемое readLine, отбрасывается после проверки, не является ли возвращаемое значение ненулевым. Почти во всех ситуациях, если результат не равен NULL, вы захотите использовать это ненулевое значение. Повторный вызов readLine даст вам другую строку.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE440 |
| CVE | — |
| Возможные последствия | Игнорирование возвращаемого значения readLine() после проверки на null может привести к потере данных, поскольку следующий вызов readLine() прочитает другую строку, а текущая строка будет утеряна. |
| Комментарий | Если readLine() не возвращает null, его результат следует сохранить в переменную и обработать, иначе логика чтения данных может быть нарушена. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  try (BufferedReader reader = new BufferedReader(new InputStreamReader([system.in](http://system.in)))) {  if (reader.readLine() != null) { // Проверка, но строка отбрасывается  System.out.println("Прочитана строка, но мы её не используем!");  }  } catch (IOException e) {  throw new RuntimeException(e);  }  } | public static void correctTest() {  try (BufferedReader reader = new BufferedReader(new InputStreamReader([system.in](http://system.in)))) {  String line = reader.readLine(); // Читаем строку один раз  if (line != null) {  System.out.println("Прочитанная строка: " + line);  }  } catch (IOException e) {  throw new RuntimeException(e);  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Сохраните результат readLine() в переменную.** Вместо того чтобы просто проверять, не является ли результат null, присвойте возвращаемое значение строковой переменной.
2. **Используйте сохраненную строку.** После проверки на null используйте переменную, содержащую прочитанную строку, для дальнейшей обработки (вывод на экран, анализ, сохранение и т.д.).
3. **Пересмотрите логику, если повторный вызов readLine() был намеренным.** В редких случаях может потребоваться пропустить строку после ее проверки, но это должно быть явно обосновано и понятно в коде. В большинстве ситуаций прочитанная строка содержит важные данные.
4. **Убедитесь, что поток чтения корректно обрабатывается в цикле.** Если вы читаете файл построчно в цикле, сохраняйте каждую прочитанную строку для последующей обработки внутри тела цикла. Избегайте повторного вызова readLine() без использования предыдущего результата.