NO\_BASE\_CALL.STAT

При переопределении метода не вызван метод базового класса, который должен быть вызван (либо потому, что остальные переопределения его вызывают, либо согласно документации). Но в этом детекторе используется статистический подход. Если при заранее указанной доле случаев процент указан.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | Roslyn;SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Игнорирование вызова метода базового класса может нарушить корректность работы программы, особенно если в родительском методе реализована важная логика. |
| Комментарий | Рекомендуется явно вызывать метод суперкласса, если это является принятой практикой или предусмотрено контрактом родительского класса. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Base {  void init() {  System.out.println("Parent init");  }  }  class Child extends Base {  @Override  void init() {  // нет вызова super  System.out.println("Child init");  }  }  Base a = new Child();  // базовый метод не вызван  a.init();  } | public static void correctTest() {  class Base {  void init() {  System.out.println("Parent init");  }  }  class Child extends Base {  @Override  void init() {  super.init();  System.out.println("Child init");  }  }  Base a = new Child();  a.init();  } |

Методы устранения уязвимости при срабатывании детектора:

1. Определите, является ли вызов метода базового класса (super.methodName()) необходимым или общепринятой практикой для переопределяемого метода. Это можно сделать, изучив документацию базового класса, просмотрев примеры использования или следуя соглашениям команды разработчиков.
2. Если вызов метода базового класса является стандартной практикой или рекомендуется, добавьте super.methodName(arguments) в начало или конец переопределенного метода. Порядок вызова может зависеть от логики базового и производного классов.