FB.PS\_PUBLIC\_SEMAPHORES

Этот класс использует синхронизацию вместе с функциями wait(), notify() или notifyAll() для себя (ссылка на this). Клиентские классы, использующие этот класс, могут, кроме того, использовать экземпляр этого класса в качестве объекта синхронизации. Поскольку два класса используют один и тот же объект для синхронизации, корректность многопоточности сомнительна. Не следует синхронизировать или вызывать методы семафора по общедоступной ссылке. Стоит рассмотреть возможность использования внутренней переменной-члена для управления синхронизацией.

Таблица №1 – Основные характеристики детектора FB.PS\_PUBLIC\_SEMAPHORES

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Concurrency Vulnerability |
| CWE | --- |
| CVE | --- |
| Возможные последствия | * Несанкционированный доступ к семафору * Нарушение работы синхронизации * Deadlock или race conditions |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // Некорректно: публичный семафор  class ResourcePool {  public final Semaphore available = new Semaphore(3); // Опасный публичный доступ    public void useResource() throws InterruptedException {  available.acquire();  try {  System.out.println("Resource used by " + Thread.currentThread().getName());  Thread.sleep(100); // Имитация работы  } finally {  available.release();  }  }  }    ResourcePool pool = new ResourcePool();  // Внешний код может нарушить работу семафора  pool.available.release(10); // Несанкционированное увеличение разрешений  } | public static void correctTest() {  // Корректно: инкапсулированный семафор  class SecureResourcePool {  private final Semaphore available = new Semaphore(3);    public void useResource() throws InterruptedException {  available.acquire();  try {  System.out.println("Resource used by " + Thread.currentThread().getName());  Thread.sleep(100);  } finally {  available.release();  }  }    // Контролируемый доступ к состоянию  public int availablePermits() {  return available.availablePermits();  }  }    SecureResourcePool securePool = new SecureResourcePool();  // Внешний код не может нарушить работу семафора  } |