FB.DP\_DO\_INSIDE\_DO\_PRIVILEGED

Этот код вызывает метод, требующий проверки разрешений безопасности. Если этому коду будут предоставлены разрешения безопасности, но он может быть вызван кодом, у которого нет разрешений безопасности, тогда вызов должен происходить внутри блока doPrivileged.

Таблица №1 – Основные характеристики детектора FB.DP\_DO\_INSIDE\_DO\_PRIVILEGED

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Privileged Operation Scope Vulnerability |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Несанкционированное повышение привилегий; нарушение принципа минимальных привилегий; уязвимости безопасности при выполнении кода с избыточными правами; обход механизмов защиты SecurityManager. |
| Комментарий | Ошибка возникает, когда в блоке doPrivileged выполняются избыточные операции, не требующие повышенных прав или критический код выходит за границы ограниченного привилегированного контекста или смешиваются привилегированные и непривилегированные операции. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // 1. Избыточные операции в doPrivileged  AccessController.doPrivileged((PrivilegedAction<Void>) () -> {  String userDir = System.getProperty("user.dir"); // Требует прав  System.out.println("Log: " + userDir); // НЕ требует прав!  return null;  });  // 2. Смешанные привилегированные операции  AccessController.doPrivileged((PrivilegedAction<Void>) () -> {  File f = new File("/secure/data.bin"); // Проверка пути - безопасна  if (f.exists()) { // Проверка существования - требует прав!  f.delete(); // Критическая операция  }  return null;  });  } | public static void correctTest() {  // 1. Минимизация привилегированного блока  String userDir = AccessController.doPrivileged(  (PrivilegedAction<String>) () -> System.getProperty("user.dir"));  System.out.println("Log: " + userDir); // Вне привилегированного блока  // 2. Разделение проверок и операций  File f = new File("/secure/data.bin");  boolean exists = AccessController.doPrivileged(  (PrivilegedAction<Boolean>) f::exists);    if (exists) {  AccessController.doPrivileged(  (PrivilegedAction<Void>) () -> {  f.delete();  return null;  });  }  } |

Методы устранения уязвимости при срабатывании детектора.

1. Принцип минимального привилегированного кода;
2. Разделение проверок и операций;
3. Использование ограниченных привилегий;
4. Анализ стека вызовов.