FB.DMI\_HARDCODED\_ABSOLUTE\_FILENAME

Детектор определяет ошибку, в ходе которой код создает объект File, используя жестко запрограммированный абсолютный путь.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE440 |
| CVE | — |
| Возможные последствия | Использование жестко заданного абсолютного пути может привести к проблемам с переносимостью кода, так как такой путь будет работать только на конкретной машине или в определенной среде. |
| Комментарий | Вместо этого следует использовать относительные пути или динамически определяемые пути, чтобы улучшить совместимость и адаптируемость программы на разных системах. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  // жестко прописанный абсолютный путь  File file = new File("C:/Users/Username/Documents/myfile.txt");  if (file.exists()) {  System.out.println("Файл существует");  } else {  System.out.println("Файл не найден");  }  } | public static void correctTest() {  // использование относительного пути  // Динамическое получение текущей рабочей директории  String workingDirectory = System.getProperty("user.dir");  File file = new File(workingDirectory + "/myfile.txt");  if (file.exists()) {  System.out.println("Файл существует");  } else {  System.out.println("Файл не найден");  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Используйте относительные пути:** Если файл находится относительно расположения приложения или текущей рабочей директории, используйте относительный путь.
2. **Используйте системные свойства:** Для доступа к стандартным каталогам (домашняя директория пользователя, временная директория и т.д.) используйте системные свойства, такие как user.home, user.dir, java.io.tmpdir.
3. **Используйте переменные окружения:** В некоторых случаях местоположение файлов может быть определено через переменные окружения. Получайте значения переменных окружения с помощью System.getenv().
4. **Используйте конфигурационные файлы:** Для определения путей к файлам и каталогам используйте конфигурационные файлы (например, .properties, XML, YAML), которые могут быть настроены для разных окружений.
5. **Передавайте пути как параметры конфигурации или аргументы командной строки:** Позвольте пользователю или процессу развертывания указывать пути к необходимым файлам и каталогам.
6. **Используйте ресурсы приложения:** Если файл является частью приложения (например, конфигурационный файл по умолчанию), рассмотрите возможность его включения в ресурсы приложения и доступа к нему через ClassLoader.getResource() или getResourceAsStream().
7. **Абстрагируйте работу с файловой системой:** Рассмотрите возможность использования абстракций (например, интерфейсов), которые позволяют работать с файлами и каталогами независимо от конкретной файловой системы и путей.