FB.DMI\_COLLECTION\_OF\_URLS

Детектор определяет ошибку, в ходе которой метод или поле является или использует карту или набор URL-адресов. Поскольку оба метода URL-адреса, Equals и hashCode, выполняют разрешение доменных имен, это может привести к значительному снижению производительности.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Использование коллекций, таких как карты или наборы, содержащие URL-адреса, может привести к значительному снижению производительности, поскольку методы equals() и hashCode() для объектов URL вызывают разрешение доменных имен при каждой операции. |
| Комментарий | Рекомендуется избегать использования URL-адресов непосредственно в коллекциях, таких как HashMap или HashSet, без предварительного разрешения доменных имен. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  try {  Set<URL> urlSet = new HashSet<>();  urlSet.add(new URL("<http://example.com>"));  urlSet.add(new URL("<http://example.com>"));  // equals и hashCode для URL могут выполнять разрешение доменных имен  System.out.println(urlSet);  } catch (Exception e) {  throw new RuntimeException(e);  }  } | public static void correctTest() {  try {  Set<CustomURL> urlSet = new HashSet<>();  URL url1 = new URL("<http://example.com>");  URL url2 = new URL("<http://example.com>");  // Создание собственных объектов с кастомными equals и hashCode  urlSet.add(new CustomURL(url1));  urlSet.add(new CustomURL(url2));  System.out.println(urlSet);  } catch (Exception e) {  throw new RuntimeException(e);  }  }  // Класс, оборачивающий URL и предоставляющий собственные реализации equals и hashCode  public static class CustomURL {  private final String protocol;  private final String host;  private final int port;  public CustomURL(URL url) {  this.protocol = url.getProtocol();  this.host = url.getHost();  this.port = url.getPort();  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || getClass() != obj.getClass()) return false;  CustomURL customURL = (CustomURL) obj;  return port == customURL.port &&  protocol.equals(customURL.protocol) &&  host.equals(customURL.host);  }  @Override  public int hashCode() {  return 31 \* protocol.hashCode() + 31 \* host.hashCode() + port;  }  @Override  public String toString() {  return protocol + "://" + host + ":" + port;  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Используйте IP-адреса:** Если возможно, храните и сравнивайте IP-адреса вместо доменных имен.
2. **Предварительное разрешение DNS:** Перед добавлением URL-адресов в коллекцию, разрешите их доменные имена и храните IP-адреса или комбинацию протокола, IP-адреса, порта и пути.
3. **Создайте класс-обертку:** Создайте собственный класс, оборачивающий URL, и переопределите методы equals() и hashCode() таким образом, чтобы они сравнивали только протокол, хост (без разрешения DNS), порт и путь. Используйте экземпляры этого класса в вашей коллекции. Пример такого класса URLKey приведен в корректном коде.
4. **Используйте URI вместо URL (если применимо):** Класс java.net.URI предназначен для идентификации ресурсов и не выполняет разрешение DNS в своих методах equals() и hashCode(). Если вам не требуется функциональность, специфичная для URL (например, открытие соединения), рассмотрите возможность использования URI.
5. **Кэширование:** Если вам необходимо работать с доменными именами, рассмотрите возможность кэширования результатов разрешения DNS, чтобы избежать повторных запросов. Однако это не решает проблему при первоначальном добавлении URL в коллекцию.
6. **Анализ производительности:** Если вы подозреваете, что коллекции URL-адресов вызывают проблемы с производительностью, проведите профилирование вашего приложения, чтобы точно определить узкие места.