Blindsqli.php

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 在直接将用户输入的 id 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。

Cryptolog.php

* **总漏洞数**: 2

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 在直接将用户输入的 user 和 pass 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。
2. **漏洞类型**: 不安全的会话处理 (Improper Session Handling)
   * **CWE 标号**: CWE-613
   * **描述**: 在会话操作中使用不安全的 session\_unset() 方法，可能导致会话未正确销毁，存在安全隐患。
   * **修复建议**: 使用 session\_destroy() 方法来确保会话被安全销毁。

Example.java

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 在直接将用户输入的 username 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。

Example1.rb

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 在直接将用户输入的 name 和 id 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。

Example2.js

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 在直接将从HTTP响应中获取的 valTom 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。

Sql.js

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 在直接将用户输入的 username 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。

Sqli.cs

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 直接将用户输入的 id 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。

Sqli.php

* **总漏洞数**: 1

1. **漏洞类型**: SQL注入 (SQL Injection)
   * **CWE 标号**: CWE-89
   * **描述**: 直接将用户输入的 id 参数拼接到 SQL 查询字符串中，存在SQL注入风险。攻击者可以通过构造恶意的输入来执行任意SQL查询。
   * **修复建议**: 使用预处理语句和参数化查询，避免直接拼接用户输入的数据到SQL查询中。