SAST & SCA Report

Repository: pnkjshahare/AICodeReview
Pull Request: #40
Generated: 2025-12-09 11:15 UTC

□ SCA - Dependency Vulnerabilities

No dependency files found. SCA scan skipped.

□ SAST - Code Vulnerabilities

snyk

December 9th 2025, 11:15:20 am (UTC+00:00)
Source: C:\Users\panka\AppData\Local\Temp\sast_69be72a1-228c-4be1-bbc2-2a0a62efc0c7

Snyk Code Report

H **0** high issues M **0** medium issues L **8** low issues

SCAN COVERAGE

.cs files: **1** .java files: **1** .js files: **2** .java files: **1**

L Use of Hardcoded Credentials

Fix Analysis

SNYK-CODE | CWE-798 | NoHardcodedCredentials/test

> Do not hardcode credentials in code. Found hardcoded credential used in password variable declaration.
>
> Found in: **SecurityTest.cs (line : 12)**

Data Flow

```
12:37
private string dbPassword = "password123";
SOURCE 0

12:24
private string dbPassword = "password123";
SINK 1
```

L Use of Hardcoded Credentials

Fix Analysis

SNYK-CODE | CWE-798 | NoHardcodedCredentials/test

> Do not hardcode credentials in code. Found hardcoded credential used in secret key variable declaration.
>
> Found in: **SecurityTest.cs (line : 13)**

Data Flow

13:33
```
private string apiKey = "AKIA123456SECRETKEY";
```
SOURCE 0

13:24
```
private string apiKey = "AKIA123456SECRETKEY";
```
SINK 1

L Hardcoded Non-Cryptographic Secret

Fix Analysis

SNYK-CODE | CWE-547 | HardcodedNonCryptoSecret/test

> Avoid hardcoding values that are meant to be secret. Found a hardcoded string used in here.
>
> Found in: **security-test.js (line : 46)**

Data Flow

security-test.js

46:17
```
const API KEY = "sk test 1234567890abcdef"; // □ Hardcoded secret
```
SOURCE 0

46:7
```
const API KEY = "sk test 1234567890abcdef"; // □ Hardcoded secret
```
SINK 1

L Hardcoded Secret

Fix Analysis

SNYK-CODE | CWE-547 | HardcodedSecret/test

> Hardcoded value is used as a cipher key (in crypto.createCipher). Generate the value with a

cryptographically strong random number generator and do not hardcode it in source code.

Found in: **security-test.js (line : 36)**

Data Flow

36:53
```
const cipher = crypto.createCipher("aes-128-ecb", "weakkey") ; // □ ECB mode insecure
```
SOURCE 0

36:18
```
const cipher = crypto.createCipher( "aes-128-ecb", "weakkey"); // □ ECB mode insecure
```
SINK 1

L Use of Insecure ECB Mode Encryption

Fix Analysis

SNYK-CODE | CWE-327 | InsecureECB/test

Electronic Code Book (aes-128-ecb) mode should not be used (in crypto.createCipher), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Galois/Counter mode).

Found in: **security-test.js (line : 36)**

Data Flow

36:38
```
const cipher = crypto.createCipher( "aes-128-ecb", "weakkey"); // □ ECB mode insecure
```
SOURCE 0

36:18
```
const cipher = crypto.createCipher( "aes-128-ecb", "weakkey"); // □ ECB mode insecure
```
SINK 1

L Use of Password Hash With Insufficient Computational Effort

Fix Analysis

SNYK-CODE | CWE-916 | InsecureHash/test

crypto.createHash hash (used in crypto.createHash) is insecure. Consider changing it to a secure hashing algorithm.

Found in: **security-test.js (line : 31)**

Data Flow

31:10

```
return crypto.createHash( "md5").update(pwd).digest("hex"); // □ Weak hash
```

SOURCE 0

31:10

```
return crypto.createHash( "md5").update(pwd).digest("hex"); // □ Weak hash
```

SINK 1

L Use of Hardcoded Passwords

Fix Analysis

SNYK-CODE | CWE-798,CWE-259 | NoHardcodedPasswords/test

> Do not hardcode passwords in code. Found hardcoded password used in PASSWORD.
>
> Found in: **security-test.js (line : 3)**

Data Flow

3:7

```
const PASSWORD = "P@ssw0rd123"; // □ Hardcoded password
```

SOURCE SINK 0

L Use of Password Hash With Insufficient Computational Effort

Fix Analysis

SNYK-CODE | CWE-916 | InsecureHash/test

> The MD5 hash (used by global::System.Security.Cryptography.MD5.Create) is insecure. Consider changing it to a secure hash algorithm.
>
> Found in: **SecurityTest.cs (line : 24)**

Data Flow

SecurityTest.cs

24:29

```
using var md5 = MD5.Create();
```

SOURCE SINK 0