Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

Bug

"runFinalizersOnExit" should not be called
Bug

"ScheduledThreadPoolExecutor" should not have 0 core threads
Bug

"Random" objects should be reused
Bug

The signature of "finalize()" should match that of "Object.finalize()"
Bug

Jump statements should not occur in "finally" blocks
Bug

"super.finalize()" should be called at the end of "Object.finalize()" implementations
Bug

Using slow regular expressions is security-sensitive
Security Hotspot

Using publicly writable directories is security-sensitive
Security Hotspot

Using clear-text protocols is security-sensitive
Security Hotspot

Accessing Android external storage is security-sensitive
Security Hotspot

Receiving intents is security-sensitive
Security Hotspot

Encryption algorithms should be used with secure mode and padding scheme

Analyze your code

Vulnerability | Critical | cwe privacy owasp sans-top25

Encryption operation mode and the padding scheme should be chosen appropriately to guarantee data confidentiality, integrity and authenticity:

- For block cipher encryption algorithms (like AES):
  - The GCM (Galois Counter Mode) mode which works internally with zero/no padding scheme, is recommended, as it is designed to provide both data authenticity (integrity) and confidentiality. Other similar modes are CCM, CWC, EAX, IAPM and OCB.
  - The CBC (Cipher Block Chaining) mode by itself provides only data confidentiality, it's recommended to use it along with Message Authentication Code or similar to achieve data authenticity (integrity) too and thus to prevent padding oracle attacks.
  - The ECB (Electronic Codebook) mode doesn't provide serious message confidentiality: under a given key any given plaintext block always gets encrypted to the same ciphertext block. This mode should not be used.
- For RSA encryption algorithm, the recommended padding scheme is OAEP.

Noncompliant Code Example

```
Cipher c1 = Cipher.getInstance("AES"); // Noncompliant: by d
Cipher c2 = Cipher.getInstance("AES/ECB/NoPadding"); // Nonc

Cipher c3 = Cipher.getInstance("RSA/None/NoPadding"); // Non
```

Compliant Solution

```
// Recommended for block ciphers
Cipher c1 = Cipher.getInstance("AES/GCM/NoPadding"); // Comp

// Recommended for RSA
Cipher c3 = Cipher.getInstance("RSA/None/OAEPWITHSHA-256ANDM
// or the ECB mode can be used for RSA when "None" is not av
Cipher c3 = Cipher.getInstance("RSA/ECB/OAEPWITHSHA-256ANDMG
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography
- MITRE, CWE-327 - Use of a Broken or Risky Cryptographic Algorithm
- CERT, MSC61-J. - Do not use insecure or weak cryptographic algorithms
- SANS Top 25 - Porous Defenses

Available In:

sonarlint sonarcloud sonarqube

Broadcasting intents is security-sensitive

Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive

Security Hotspot

Configuring loggers is security-sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot