




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



Java static code analysis

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

All rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

Zero should not be a possible denominator

Bug

Locks should be released

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

Cipher algorithms should be robust

Analyze your code

VulnerabilityCritical🔍cwe privacy owasp sans-top25

Strong cipher algorithms are cryptographic systems resistant to cryptanalysis, they are not vulnerable to well-known attacks like brute force attacks for example.

A general recommendation is to only use cipher algorithms intensively tested and promoted by the cryptographic community.

More specifically for block cipher, it's not recommended to use algorithm with a block size inferior than 128 bits.

Noncompliant Code Example

```
import javax.crypto.Cipher;
import java.security.NoSuchAlgorithmException;
import javax.crypto.NoSuchPaddingException;

public class test {

    public static void main(String[] args) {
        try
        {
            Cipher c1 = Cipher.getInstance("DES"); // Noncompliance
            Cipher c7 = Cipher.getInstance("DESede"); // Noncompliance
            Cipher c13 = Cipher.getInstance("RC2"); // Noncompliance
            Cipher c19 = Cipher.getInstance("RC4"); // Noncompliance
            Cipher c25 = Cipher.getInstance("Blowfish"); // Noncompliance

            NullCipher nc = new NullCipher(); // Noncompliance: t
        }
        catch(NoSuchAlgorithmException|NoSuchPaddingException
        {
        }
    }
}
```

Compliant Solution





```
import javax.crypto.Cipher;
import java.security.NoSuchAlgorithmException;
import javax.crypto.NoSuchPaddingException;

public class test {

    public static void main(String[] args) {
        try
        {
            Cipher c31 = Cipher.getInstance("AES/GCM/NoPadding")
        }
        catch(NoSuchAlgorithmException|NoSuchPaddingException
        {
        }
    }
}
```

https://rules.sonarsource.com/java/RSPEC-5547

1/2

 Security Hotspot
Receiving intents is security-sensitive
 Security Hotspot
Broadcasting intents is security-sensitive
 Security Hotspot
Expanding archive files without controlling resource consumption is security-sensitive
 Security Hotspot
Configuring loggers is security-sensitive

```
    }  
}
```

See

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [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** 

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)