

































-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

PL/SQL

PL/SQL static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PL/SQL code

All rules

188

 Vulnerability

4

 Bug

45

 Security Hotspot

2

 Code Smell

137

Tags

Search by name...



 Bug

Using weak hashing algorithms is security-sensitive

 Security Hotspot

Dynamically executing code is security-sensitive

 Security Hotspot

SQL "JOIN" conditions should involve all joined tables

 Code Smell

"SELECT" statements used as argument of "EXISTS" statements should be selective

 Code Smell

"LIKE" clauses should not be used without wildcards

 Code Smell

Weak "REF CURSOR" types should not be used

 Code Smell

Whitespace and control characters in string literals should be explicit

 Code Smell

Blocks containing "EXECUTE IMMEDIATE" should trap all exceptions

 Code Smell

"EXCEPTION_INIT -20,NNN" calls should be centralized

 Code Smell

"CREATE OR REPLACE" should be used instead of "CREATE"

 Code Smell

Block labels should appear on the same lines as "END"

 Code Smell

"LOOP ... END LOOP;" constructs should be avoided

 Code Smell

"IF" statements should not be nested too deeply

 Code Smell

Using weak hashing algorithms is security-sensitive

Analyze your code

 Security Hotspot

 Critical



 cwe spring owasp sans-top25

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, HMAC-MD5, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160, HMACRIPEMD160 and SHA-1 are no longer considered secure, because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...).
- To compute some message integrity.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like bcrypt, scrypt, argon2 or pbkdf2 because it slows down brute force attacks.

See

- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [MITRE, CWE-327](#) - Use of a Broken or Risky Cryptographic Algorithm
- [MITRE, CWE-916](#) - Use of Password Hash With Insufficient Computational Effort
- [SANS Top 25](#) - Porous Defenses

Available In:

sonarcloud



sonarqube



Developer Edition