- 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...

Code Smell

Quoted identifiers should not be used
Code Smell

Loop start and end labels should match
Code Smell

Block start and end labels should match
Code Smell

Cipher algorithms should be robust
Vulnerability

"COMMIT" should not be used inside a loop
Bug

Individual "WHERE" clause conditions should not be unconditionally true or false
Bug

Nullable subqueries should not be used in "NOT IN" conditions
Bug

"COMMIT" and "ROLLBACK" should not be called from non-autonomous transaction triggers
Bug

Positional and named arguments should not be mixed in invocations
Bug

Functions should end with "RETURN" statements
Bug

Collections should not be iterated in "FOR" loops
Bug

Using weak hashing algorithms is security-sensitive
Security Hotspot

Dynamically executing code is security-sensitive
Security Hotspot

GOTO should not be used to jump backwards

Analyze your code

Code Smell | Blocker | pitfall

Jumping back to a previous statement using `GOTO` is a way to reimplement loops, which PL/SQL already provides in much more readable forms.

Noncompliant Code Example

```
SET SERVEROUTPUT ON

DECLARE
  result PLS_INTEGER := 0;
  counter PLS_INTEGER := 1;
BEGIN
  <<loop>>
  result := result + counter;
  counter := counter + 1;

  IF counter <= 9 THEN
    GOTO loop;                     -- Noncompliant
  END IF;

  DBMS_OUTPUT.PUT_LINE('Sum from 1 to 9 is ' || result); -- Displays 1 + 2 + ... + 8 + 9 = 45
END;
/
```

Compliant Solution

```
SET SERVEROUTPUT ON

DECLARE
  result PLS_INTEGER := 0;
BEGIN
  FOR counter IN 1 .. 9 LOOP
    result := result + counter;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Sum from 1 to 9 is ' || result); -- Displays 1 + 2 + ... + 8 + 9 = 45
END;
/
```

Available In:
sonarlint | sonarcloud | sonarqube Developer Edition