





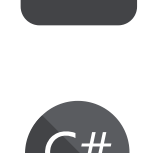







































-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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

Exceptions should not be ignored	
Variables should not be initialized with "NULL"	
Boolean checks should not be inverted	
Unused local variables should be removed	
Package names should comply with a naming convention	
Variables should comply with a naming convention	
Return of boolean expressions should not be wrapped into an "if-then-else" statement	
Boolean literals should not be redundant	
Comments should not be nested	
"DBMS_UTILITY.FORMAT_ERROR_STACK" and "FORMAT_ERROR_BACKTRACE" should be used together	
Procedures should not contain "RETURN" statements	
Track uses of "TODO" tags	

Tags 

Search by name... 

Exceptions should not be ignored

Analyze your code

-  Code Smell
-  Minor 
-  cwe error-handling owasp suspicious

When exceptions occur, it is usually a bad idea to simply ignore them. Instead, it is better to handle them properly, or at least to log them.

Noncompliant Code Example

```
SET SERVEROUTPUT ON

DECLARE
    d VARCHAR2(1);
BEGIN
    SELECT dummy INTO d FROM DUAL WHERE dummy = 'Y'; -- Will raise NO_DATA_FOUND
    DBMS_OUTPUT.PUT_LINE('d = ' || d);
EXCEPTION
    WHEN NO_DATA_FOUND THEN -- Noncompliant, did we really want to mask this exception?
        NULL;
END;
/
```

Compliant Solution

```
SET SERVEROUTPUT ON

DECLARE
    d VARCHAR2(1);
BEGIN
    SELECT dummy INTO d FROM DUAL WHERE dummy = 'Y'; -- Will raise NO_DATA_FOUND
    DBMS_OUTPUT.PUT_LINE('d = ' || d);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: No data found');
END;
/
```

See

- [OWASP Top 10 2017 Category A10](#) - Insufficient Logging & Monitoring
- [MITRE, CWE-391](#) - Unchecked Error Condition

Available In:

 |  |  Developer Edition