


















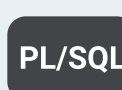














-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

PL/SQL

PL/SQL static code analysis

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

All rules188


 Vulnerability4

 Bug45


 Security Hotspot2















 Code Smell137

Tags



Search by name...



<div><div>Dynamically executing code is security-sensitive</div><div> Security Hotspot</div></div>
<div><div>SQL "JOIN" conditions should involve all joined tables</div><div> Code Smell</div></div>
<div><div>"SELECT" statements used as argument of "EXISTS" statements should be selective</div><div> Code Smell</div></div>
<div><div>"LIKE" clauses should not be used without wildcards</div><div> Code Smell</div></div>
<div><div>Weak "REF CURSOR" types should not be used</div><div> Code Smell</div></div>
<div><div>Whitespace and control characters in string literals should be explicit</div><div> Code Smell</div></div>
<div><div>Blocks containing "EXECUTE IMMEDIATE" should trap all exceptions</div><div> Code Smell</div></div>
<div><div>"EXCEPTION_INIT -20,NNN" calls should be centralized</div><div> Code Smell</div></div>
<div><div>"CREATE OR REPLACE" should be used instead of "CREATE"</div><div> Code Smell</div></div>
<div><div>Block labels should appear on the same lines as "END"</div><div> Code Smell</div></div>
<div><div>"LOOP ... END LOOP;" constructs should be avoided</div><div> Code Smell</div></div>
<div><div>"IF" statements should not be nested too deeply</div><div> Code Smell</div></div>
<div><div>"CASE" expressions should end with "ELSE" clauses</div><div> Code Smell</div></div>
<div><div>String literals should not be duplicated</div><div> Code Smell</div></div>

Dynamically executing code is security-sensitive

Analyze your code

 Security Hotspot

 Critical



 cwe owasp

Executing code dynamically is security sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2017-9807](#)
- [CVE-2017-9802](#)

Any code which is dynamically evaluated in your process will have the same permissions as the rest of your code. Thus it is very dangerous to do so with code coming from an untrusted source. [Injected Code](#) can either run on the server or in the client (exemple: XSS attack).

EXECUTE IMMEDIATE executes as a dynamic SQL statement or anonymous PL/SQL block the string passed as an argument. It's safe only if the argument is composed of constant character string expressions. But if the command string is dynamically built using external parameters, then it is considered very dangerous because executing a random string allows the execution of arbitrary code.

This rule marks for review each occurrence of dynamic code execution.

Ask Yourself Whether

- the executed code may come from an untrusted source and hasn't been sanitized.
- you really need to run code dynamically.

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

Recommended Secure Coding Practices

The best solution is to not run code provided by an untrusted source. If you really need to build a command string using external parameters, you should use EXECUTE IMMEDIATE with bind variables instead.

Do not try to create a blacklist of dangerous code. It is impossible to cover all attacks that way.

Sensitive Code Example

```
CREATE OR REPLACE PROCEDURE ckpwd (p_user IN VARCHAR2, p_pass IN VARCHAR2)
IS
  v_query   VARCHAR2(100);
  v_output  NUMBER;
BEGIN
  v_query :=      q'{SELECT COUNT(*) FROM user_pwd }'
                || q'{WHERE username = '}'
                || p_user
                || q'{ ' AND password = '}'
                || p_pass
                || q'{'}}';
  EXECUTE IMMEDIATE v_query
    INTO v_output;
END;
```

Compliant Solution

```
CREATE OR REPLACE PROCEDURE ckpwd_bind (p_user IN VARCHAR2, p_pass IN VARCHAR2)
IS
  v_query   VARCHAR2(100);
  v_output  NUMBER;
BEGIN
  v_query :=
    q'{SELECT COUNT(*) FROM user_pwd WHERE username = :1 AND password = :2}';
  EXECUTE IMMEDIATE v_query
    INTO v_output
    USING p_user, p_pass;
END;
```

See

- [OWASP Top 10 2017 Category A1](#) - Injection
- [MITRE CWE-95](#) - Improper Neutralization of Directives in Dynamically Evaluated Code (Eval Injection)

Available In:

sonarcloud



sonarqube

 Developer Edition