

In-IDE



IDE extension that lets you fix coding issues before they exist!

Discover SonarLint →

SaaS



Setup is effortless and analysis is automatic for most languages

































Discover SonarCloud →

Self-Hosted






Fast, accurate analysis; enterprise scalability

Discover SonarQube →

 Go	redundant  Code Smell
 HTML	
 Java	Default export names and file names should match  Code Smell
 JavaScript	
 Kotlin	The global "this" object should not be used  Code Smell
 Objective C	
 PHP	"catch" clauses should do more than rethrow  Code Smell
 PL/I	
 PL/SQL	Boolean checks should not be inverted  Code Smell
 Python	
 RPG	Deprecated APIs should not be used  Code Smell
 Ruby	
 Scala	Wrapper objects should not be used for primitive types  Code Smell
 Swift	
 Terraform	Multiline string literals should not be used  Code Smell
 Text	
 TypeScript	Local variables should not be declared and then immediately returned or thrown  Code Smell
 T-SQL	
 VB.NET	Function call arguments should not start on new lines  Code Smell
 VB6	
 XML	"switch" statements should have at least 3 "case" clauses  Code Smell
	A "while" loop should be used instead of a "for" loop

Using shell interpreter when executing OS commands is security-sensitive

Analyze your code

 Security Hotspot  Major  cwe owasp sans-top25

Arbitrary OS command injection vulnerabilities are more likely when a shell is spawned rather than a new process, indeed shell meta-chars can be used (when parameters are user-controlled for instance) to inject OS commands.

Ask Yourself Whether

- OS command name or parameters are user-controlled.

There is a risk if you answered yes to this question.

Recommended Secure Coding Practices

Use functions that don't spawn a shell.

Sensitive Code Example

```
const cp = require('child_process');

// A shell will be spawn in these following cases:
cp.exec(cmd); // Sensitive
cp.execSync(cmd); // Sensitive

cp.spawn(cmd, { shell: true }); // Sensitive
cp.spawnSync(cmd, { shell: true }); // Sensitive
cp.execFile(cmd, { shell: true }); // Sensitive
cp.execFileSync(cmd, { shell: true }); // Sensitive
```

Compliant Solution



```
const cp = require('child_process');





cp.spawnSync("/usr/bin/file.exe", { shell: false }); // Comp
```

See

- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Top 10 2017 Category A1](#) - Injection
- [MITRE, CWE-78](#) - Improper Neutralization of Special Elements used in an OS Command
- [SANS Top 25](#) - Insecure Interaction Between Components

Available In:

 Code Smell
Unnecessary imports should be removed  Code Smell
Boolean literals should not be used in comparisons  Code Smell
Extra semicolons should be removed  Code Smell
Class names should comply with a naming convention

© 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](#)