sonar RULES

Products ⌄

⊘	Secrets
SAP	ABAP
APEX	Apex
C	C
C	C++
⬡	CloudFormation
COBOL	COBOL
C#	C#
CSS	CSS
✗	Flex
GO	Go
HTML	HTML
☕	Java
JS	JavaScript
K	Kotlin
⎈	Kubernetes
	Objective C
php	PHP
PL/I	PL/I
PL/SQL	PL/SQL
🐍	Python
RPG	RPG
◆	Ruby
≣	Scala
≋	Swift
⌦	Terraform
🗎	Text
TS	TypeScript
⚒	T-SQL
VB	VB.NET
VB6	VB6
XML	XML

Objective C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your OBJECTIVE C code

All rules	315	🔒 Vulnerability	10	🐛 Bug	75	◈ Security Hotspot	18	◉ Code Smell	212	↻ Quick Fix	13

Tags ⌄   Search by name... 🔍

"memset" should not be used to delete sensitive data

🔒 Vulnerability

POSIX functions should not be called with arguments that trigger buffer overflows

🔒 Vulnerability

Function-like macros should not be invoked without all of their arguments

🐛 Bug

The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist

🐛 Bug

"pthread_mutex_t" should be unlocked in the reverse order they were locked

🐛 Bug

"pthread_mutex_t" should be properly initialized and destroyed

🐛 Bug

"pthread_mutex_t" should not be consecutively locked or unlocked twice

🐛 Bug

Functions with "noreturn" attribute should not return

🐛 Bug

"memcmp" should only be called with pointers to trivially copyable types with no padding

🐛 Bug

Stack allocated memory and non-owned memory should not be freed

🐛 Bug

Closed resources should not be accessed

🐛 Bug

Dynamically allocated memory should be released

🐛 Bug

Unused assignments should be removed

Analyze your code

◉ Code Smell   ◆ Major ?   🏷 cwe symbolic-execution cert unused

A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources. Therefore all calculated values should be used.

Noncompliant Code Example

```
i = a + b; // Noncompliant; calculation result not used befor
i = compute();
```

Compliant Solution

```
i = a + b;
i += compute();
```

Exceptions

This rule ignores:

- variable declarations initializers
- prefix and postfix increments and decrements `x++;`
- null pointer assignments `x = NULL;`
- self assignments (i.e. `x = x;`)

See

- MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable')
- CERT, MSC13-C. - Detect and remove unused values
- CERT, MSC56-J. - Detect and remove superfluous code and values

Available In:

sonarcloud 🌩 | sonarqube ◥ Developer Edition

Freed memory should not be used

🐞 Bug

Memory locations should not be released more than once

🐞 Bug

Memory access should be explicitly bounded to prevent buffer overflows

🐞 Bug

Printf-style format strings should not lead to unexpected behavior at runtime

🐞 Bug

Recursion should not be infinite

🐞 Bug

Resources should be closed

🐞 Bug

Hard-coded credentials are security-sensitive

🛡 Security Hotspot

"goto" should jump to labels declared later in the same function

☢ Code Smell

Only standard forms of the "defined" directive should be used

☢ Code Smell

Switch labels should not be nested inside non-switch blocks

☢ Code Smell