Objective C static code analysis

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

Object declarations should contain no more than 2 levels of pointer indirection

Analyze your code

Code Smell | Critical | based-on-misra brain-overload pitfall

While they are extraordinarily useful, pointers are not the most intuitive concept in the world. Pointers to pointers are even harder to understand and use correctly. And with each additional level of indirection, pointer variables become more difficult to use correctly. Therefore pointer declarators should be limited to no more than two levels of nesting.

Noncompliant Code Example

```
typedef int * INTPTR;
struct s {
 int ** s1;
 int *** s2; // Noncompliant
};

struct s ** ps1;
struct s *** ps2; // Noncompliant

int ** ( *pfunc1)();
int ** ( **pfunc2)();
int ** (***pfunc3)(); // Noncompliant
int *** ( **pfunc4)(); // Noncompliant

void function( int ** par1,
               int *** par2, // Noncompliant
               INTPTR * par3,
               int * par4[],
               int ** par5[]) // Noncompliant
{
  int ** ptr1;
  int *** ptr2; // Noncompliant
  INTPTR * ptr3;
  int * ptr4[ 10 ];
  int ** ptr5[ 10 ]; //Noncompliant
}
```

Compliant Solution

```
typedef int * INTPTR;
struct s {
 int ** s1;
 int ** s2;
};

struct s ** ps1;
struct s ** ps2;

int ** (*pfunc1)();
int ** (**pfunc2)();
int ** (**pfunc3)();
int ** (**pfunc4)();

void function( int ** par1,
               int ** par2,
               INTPTR * par3,
               int * par4[],
               int * par5[])
{
```

```
  int ** ptr1;
  int ** ptr2;
  INTPTR * ptr3;
  int * ptr4[ 10 ];
  int * ptr5[ 10 ];
}
```

See

- MISRA C:2004, 17.5 - The declaration of objects should contain no more than 2 levels of pointer indirection
- MISRA C++:2008, 5-0-19 - The declaration of objects shall contain no more than two levels of pointer indirection
- MISRA C:2012, 18.5 - Declarations should contain no more than two levels of pointer nesting

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