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...

GNU extensions should not be used

Analyze your code

Code Smell   Minor   lock-in obsolete gnu

Proprietary compiler extensions can be handy, but they commit you to always using that compiler. This rule raises an issue when GNU extensions are used, such as:

- Ternary operator with omitted second operand
- Case ranges in switch statements
- Expression statements, i.e. code blocks producing value
- Index range in array initializers
- A array initializer without =
- A structure member initializer with a colon
- Decimal floating points numbers `_Decimal32`, `_Decimal64`, and `_Decimal128`
- Structures and union without named data members

Noncompliant Code Example

```
struct S {
  int f;
};

struct S s[] = {
  [0] { // Noncompliant
    f : 0 // Noncompliant
  }
  [1 ... 3] = { // CHECK :8 :11 S3715:use of GNU array range
    .f = 2
  }
};

int fun(int p) {
  switch (p) {
    case 0 ... 1: // Noncompliant
      do_the_thing();
      break;
    case 2:
      //...
  }

  p = ({ // Noncompliant
    int a = 10, b = 20;
    (a * b) + 10;
  });

  return p ?: 0; // Noncompliant
}

_Decimal32 d32; // Noncomplaint

struct Empty {}; // Noncomplaint in C
```

Compliant Solution

```
struct S {
  int f;
};

struct S s[] = {
  [0] = {
```

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

```
      .f = 0
    },
    [1] = {
      .f = 2
    }
    [2] = {
      .f = 2
    },
    [3] = {
      .f = 2
    }

};

int fun(int p) {
  switch (p) {
    case 0:
    case 1:
      do_the_thing();
      break;
    case 2:
      //...
  }

  int a = 10, b = 20;
  p = (a * b) + 10;

  return p ? p: 0;
}
```

Available In:
sonarcloud | sonarqube Developer Edition