C static code analysis: GNU attributes should be used correctly

1-2 minutes

`__attribute__` is a GNU extension that allows to decorate functions, parameters, variables… with some attributes. It may help for compiler optimizations or for the writer of some code to better state his intent (and have the compiler check it).

If this extension is used incorrectly, it will usually not break the build, but it still means that the code may not behave as the developer expects. This rule reports such occurrences of bad use of `__attribute__`.

Noncompliant Code Example

```
int f1() __attribute__((returns_nonnull)); // Noncompliant; "returns_nonnull" only applies to return values which are pointers

void g(int *a) __attribute__((nonnull(1))){} // Noncompliant; "nonnull" position in the function definition is not allowed

void h() __attribute__((warn_unused_result)); // Noncompliant; "warn_unused_result" does not work with function without return value

void test() {
  int __declspec(empty_bases)i; // Noncompliant; "empty_bases" only applies to classes
  char c = (char __attribute__((aligned(8)))) i; // Noncompliant, attribute is ignored
```

}}