

Objective C static code analysis: "memset" should not be used to delete sensitive data

2-3 minutes

The compiler is generally allowed to remove code that does not have any effect, according to the abstract machine of the C language. This means that if you have a buffer that contains sensitive data (for instance passwords), calling memset on the buffer before releasing the memory will probably be optimized away.

The function [memset_s](#) behaves similarly to memset, but the main difference is that it cannot be optimized away, the memory will be overwritten in all cases. You should always use this function to scrub security-sensitive data.

This rule raises an issue when a call to memset is followed by the destruction of the buffer.

Note that memset_s is defined in annex K of C11, so to have access to it, you need a standard library that supports it (this can be tested with the macro `__STDC_LIB_EXT1__`), and you need to enable it by defining the macro `__STDC_WANT_LIB_EXT1__` before including `<string.h>`. Other platform specific functions can perform the same operation, for instance [SecureZeroMemory](#) (Windows) or [explicit_bzero](#) (FreeBSD)

Noncompliant Code Example

```
void f(char *password, size_t bufferSize) {
    char localToken[256];
    init(localToken, password);
    memset(password, ' ', strlen(password)); // Noncompliant,
password is about to be freed
    memset(localToken, ' ', strlen(localToken)); // Noncompliant,
localToken is about to go out of scope
    free(password);
}
```

Compliant Solution

```
void f(char *password, size_t bufferSize) {
    char localToken[256];
    init(localToken, password);
    memset_s(password, bufferSize, ' ', strlen(password));
    memset_s(localToken, sizeof(localToken), ' ', strlen(localToken));
    free(password);
}
```

See

- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-14](#) - Compiler Removal of Code to Clear Buffers

Available In: