

Objective C static code analysis: Non-reentrant POSIX functions should be replaced with their reentrant versions

3 minutes

A function is called reentrant if it can be interrupted in the middle of its execution and then safely called again ("re-entered") before its previous invocations complete execution.

It is especially important that multi-threaded applications do not call the same non-reentrant function from different threads.

This rule will trigger an issue each time a function in the configurable list is invoked.

Noncompliant Code Example

Given a function that includes `localtime`:

```
#include <stdio.h>
#include <time.h>

void print_date_and_time(struct tm *time_ptr)
{
    printf(
        "Current date and time: %d/%02d/%02d %02d:%02d:%02d\n",
        time_ptr->tm_year + 1900,
        time_ptr->tm_mon,
        time_ptr->tm_mday,
        time_ptr->tm_hour,
        time_ptr->tm_min,
        time_ptr->tm_sec);
}

void print_unix_epoch_date_and_time()
{
    time_t unix_epoch_time = (time_t)0;
    struct tm *local_time_ptr = localtime(&unix_epoch_time); //
    Noncompliant, call to the non-reentrant localtime() function
    print_date_and_time(local_time_ptr);
}

int main(int argc, char* argv[])
{
    time_t current_time;
    struct tm *local_time_ptr;

    time(&current_time);

    local_time_ptr = localtime(&current_time); // Noncompliant, call to
    the non-reentrant localtime() function

    // As expected, this will print: Current date and time: 1970/00/01
    01:00:00
    print_unix_epoch_date_and_time();

    // This will actually also print Current date and time: 1970/00/01
    01:00:00
    // Indeed, localtime() is non-reentrant, and always returns the
    same pointer
    print_date_and_time(local_time_ptr);

    return 0;
}
```

Compliant Solution

```
#include <stdio.h>
#include <time.h>

void print_date_and_time(struct tm *time_ptr)
{
    printf(
        "Current date and time: %d/%02d/%02d %02d:%02d:%02d\n",
```

```
    time_ptr->tm_year + 1900,
    time_ptr->tm_mon,
    time_ptr->tm_mday,
    time_ptr->tm_hour,
    time_ptr->tm_min,
    time_ptr->tm_sec);
}

void print_unix_epoch_date_and_time()
{
    time_t unix_epoch_time = (time_t)0;
    struct tm local_time;
    localtime_r(&unix_epoch_time, &local_time); // Compliant
    print_date_and_time(&local_time);
}

int main(int argc, char* argv[])
{
    time_t current_time;
    struct tm local_time;

    time(&current_time);

    localtime_r(&current_time, &local_time); // Compliant

    // As expected, this will print: Current date and time: 1970/00/01
    01:00:00
    print_unix_epoch_date_and_time();

    // As expected, this will print the current date and time
    print_date_and_time(&local_time);

    return 0;
}
```

Available In: