Use of potentially dangerous function

Name: Use of potentially dangerous function

Description: Use of a standard library function that is not thread-safe.

ID: cpp/potentially-dangerous-function

Kind: problem

Severity: warning

Precision: high

This rule finds calls to functions that are dangerous to use. Currently, it checks for calls to `gmtime`, `localtime`, `ctime` and `asctime`.

The time related functions such as `gmtime` fill data into a `tm` struct or `char` array in shared memory and then returns a pointer to that memory. If the function is called from multiple places in the same program, and especially if it is called from multiple threads in the same program, then the calls will overwrite each other's data.

Recommendation

Replace calls to `gmtime` with `gmtime_r`. With `gmtime_r`, the application code manages allocation of the `tm` struct. That way, separate calls to the function can use their own storage.

Similarly replace calls to `localtime` with `localtime_r`, calls to `ctime` with `ctime_r` and calls to `asctime` with `asctime_r`.

Example

The following example checks the local time in two ways:

```
// BAD: using gmtime
int is_morning_bad() {
    const time_t now_seconds = time(NULL);
    struct tm *now = gmtime(&now_seconds);
    return (now->tm_hour < 12);
}

// GOOD: using gmtime_r
int is_morning_good() {
    const time_t now_seconds = time(NULL);
    struct tm now;
    gmtime_r(&now_seconds, &now);
    return (now.tm_hour < 12);
}
```

The first version uses `gmtime`, so it is vulnerable to its data being overwritten by another thread. Even if this code is not used in a multi-threaded context right now, future changes may make the program multi-threaded. The second version of the code uses `gmtime_r`. Since it allocates a new `tm` struct on every call, it is immune to other calls to `gmtime` or `gmtime_r`.

References

- SEI CERT C Coding Standard: CON33-C. Avoid race conditions when using library functions.
- Common Weakness Enumeration: CWE-676.