rules.sonarsource.com

C++ static code analysis: Accessing files should not introduce TOCTOU vulnerabilities

3-4 minutes

"Time Of Check to Time Of Use" (TOCTOU) vulnerabilities occur when an application:

- First, checks permissions or attributes of a file: for instance, is a file a symbolic link?
- Next, performs some operations such as writing data to this file.

The application cannot assume the state of the file is unchanged between these two steps, there is a race condition (ie: two different processes can access and modify the same shared object/file at the same time, which can lead to privilege escalation, denial of service and other unexpected results).

For instance, attackers can benefit from this situation by creating a symbolic link to a sensitive file directly after the first step (eg in Unix: `/etc/passwd`) and try to elevate their privileges (eg: if the written data has the correct `/etc/passwd` file format).

To avoid TOCTOU vulnerabilities, one possible solution is to do a single atomic operation for the "check" and "use" actions, therefore removing the race condition window. Another possibility is to use file descriptors. This way the binding of the file descriptor to the file cannot be changed by a concurrent process.

Noncompliant Code Example

A "check function" (for instance `access`, `stat` … in this case `access` to verify the existence of a file) is used, followed by a "use function" (`open`, `fopen` …) to write data inside a non existing file. These two consecutive calls create a TOCTOU race condition:

```
#include <stdio.h>

void fopen_with_toctou(const char *file) {
  if (access(file, F_OK) == -1 && errno == ENOENT) {
    // the file doesn't exist
    // it is now created in order to write some data
inside
    FILE *f = fopen(file, "w"); // Noncompliant: a race
condition window exist from access() call to fopen()
call calls
    if (NULL == f) {
      /* Handle error */
    }

    if (fclose(f) == EOF) {
      /* Handle error */
    }
  }
}
```

Compliant Solution

If the file already exists on the disk, `fopen` with x mode will fail:

```
#include <stdio.h>

void open_without_toctou(const char *file) {
  FILE *f = fopen(file, "wx"); // Compliant
  if (NULL == f) {
    /* Handle error */
  }
  /* Write to file */
  if (fclose(f) == EOF) {
    /* Handle error */
  }
}
```

A more generic solution is to use "file descriptors":

```
void open_without_toctou(const char *file) {
  int fd = open(file, O_CREAT | O_EXCL |
O_WRONLY);
  if (-1 != fd) {
    FILE *f = fdopen(fd, "w");  // Compliant
  }
}
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-367 - Time-of-check Time-of-use (TOCTOU) Race Condition
- CERT, FIO45-C. - Avoid TOCTOU race conditions while accessing files