FIO24-C. Do not open a file that is already open

Opening a file that is already open has implementation-defined behavior, according to the C Standard, 7.21.3, paragraph 8 [ISO/IEC 9899:2011]:

> *Functions that open additional (nontemporary) les require a file name, which is a string. The rules for composing valid le names are implementation-dened. Whether the same file can be simultaneously open multiple times is also implementation-dened.*

Some implementations do not allow multiple copies of the same file to be open at the same time. Consequently, portable code cannot depend on what will happen if this rule is violated. Even on implementations that do not outright fail to open an already-opened file, a TOCTOU (time-of-check, time-of-use) race condition exists in which the second open could operate on a different file from the first due to the file being moved or deleted (see FIO45-C. Avoid TOCTOU race conditions while accessing files for more details on TOCTOU race conditions).

Noncompliant Code Example

This noncompliant code example logs the program's state at runtime:

```
#include <stdio.h>

void do_stuff(void) {
  FILE *logfile = fopen("log", "a");
  if (logfile == NULL) {
    /* Handle error */
  }

  /* Write logs pertaining to do_stuff() */
  fprintf(logfile, "do_stuff\n");
}

int main(void) {
  FILE *logfile = fopen("log", "a");
  if (logfile == NULL) {
    /* Handle error */
  }

  /* Write logs pertaining to main() */
  fprintf(logfile, "main\n");

  do_stuff();

  if (fclose(logfile) == EOF) {
    /* Handle error */
  }
  return 0;
}
```

Because the file `log` is opened twice (once in `main()` and again in `do_stuff()`), this program has implementation-defined behavior.

Compliant Solution

In this compliant solution, a reference to the file pointer is passed as an argument to functions that need to perform operations on that file. This reference eliminates the need to open the same file multiple times.

```
#include <stdio.h>

void do_stuff(FILE *logfile) {
  /* Write logs pertaining to do_stuff() */
  fprintf(logfile, "do_stuff\n");
}

int main(void) {
  FILE *logfile = fopen("log", "a");
  if (logfile == NULL) {
    /* Handle error */
  }

  /* Write logs pertaining to main() */
  fprintf(logfile, "main\n");

  do_stuff(logfile);

  if (fclose(logfile) == EOF) {
    /* Handle error */
  }
  return 0;
}
```

Risk Assessment

Simultaneously opening a file multiple times can result in unexpected errors and nonportable behavior.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO31-C	Medium	Probable	High	**P4**	**L3**

Automated Detection

Tool	Version	Checker	Description
CodeSonar	5.1p0	**IO.RACE (customization)**	File system race condition Users can implement a custom check that triggers a warning if a file-opening function is called on a file that is already open
LDRA tool suite	9.7.1	**75 D**	Partially implemented
Parasoft C/C++test	10.4.2	**CERT_C-FIO24-a**	Avoid race conditions while accessing files
Polyspace Bug Finder	R2019b	CERT C: Rec. FIO24-C	Checks for situations where previously opened resources are reopened (rec. fully covered)

Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

SEI CERT C Coding Standard	FIO45-C. Avoid TOCTOU race conditions while accessing files
SEI CERT C++ Coding Standard	VOID FIO21-CPP. Do not simultaneously open the same file multiple times
MITRE CWE	CWE-362, Concurrent Execution Using Shared Resource with Improper Synchronization ("Race Condition") CWE-675, Duplicate Operations on Resource

Bibliography

[ISO/IEC 9899:2011]	Subclause 7.21.3, "Files"