FIO03-C. Do not make assumptions about fopen() and file creation

The C `fopen()` function is used to open an existing file or create a new one. The C11 version of the `fopen()` and `fopen_s()` functions provides a mode flag, `x`, that provides the mechanism needed to determine if the file that is to be opened exists. Not using this mode flag can lead to a program overwriting or accessing an unintended file.

Noncompliant Code Example (`fopen()`)

In this noncompliant code example, the file referenced by `file_name` is opened for writing. This example is noncompliant if the programmer's intent was to create a new file, but the referenced file already exists.

```
char *file_name;
FILE *fp;

/* Initialize file_name */

fp = fopen(file_name, "w");
if (!fp) {
  /* Handle error */
}
```

Noncompliant Code Example (`fopen_s()`, C11 Annex K)

The C11 Annex K `fopen_s()` function is designed to improve the security of the `fopen()` function. Like the `fopen()` function, `fopen_s()` provides a mechanism to determine whether the file exists. See below for use of the exclusive mode flag.

```
char *file_name;
FILE *fp;

/* Initialize file_name */
errno_t res = fopen_s(&fp, file_name, "w");
if (res != 0) {
  /* Handle error */
}
```

Compliant Solution (`fopen_s()`, C11 Annex K)

The C Standard provides a new flag to address this problem. Subclause 7.21.5.3, paragraph 5 [ISO/IEC 9899:2011], states:

> *Opening a file with exclusive mode ('x' as the last character in the mode argument) fails if the file already exists or cannot be created. Otherwise, the file is created with exclusive (also known as non-shared) access to the extent that the underlying system supports exclusive access.*

This option is also provided by the GNU C library [Loosemore 2007].

This compliant solution uses the `x` mode character to instruct `fopen_s()` to fail rather than open an existing file:

```
char *file_name;

/* Initialize file_name */
FILE *fp;
errno_t res = fopen_s(&fp, file_name, "wx");
if (res != 0) {
  /* Handle error */
}
```

Use of this option allows for the easy remediation of legacy code. However, note that Microsoft Visual Studio 2012 and earlier do not support the `x` mode character [MSDN].

Compliant Solution (`open()`, POSIX)

The `open()` function, as defined in the *Standard for Information Technology—Portable Operating System Interface (POSIX®), Base Specifications, Issue 7* [IEEE Std 1003.1:2013], is available on many platforms and provides finer control than `fopen()`. In particular, `open()` accepts the `O_CREAT` and `O_EXCL` flags. When used together, these flags instruct the `open()` function to fail if the file specified by `file_name` already exists.

```
char *file_name;
int new_file_mode;

/* Initialize file_name and new_file_mode */

int fd = open(file_name, O_CREAT | O_EXCL | O_WRONLY, new_file_mode);
if (fd == -1) {
  /* Handle error */
}
```

Care should be taken when using `O_EXCL` with remote file systems because it does not work with NFS version 2. NFS version 3 added support for `O_EXCL` mode in `open()`. IETF RFC 1813 [Callaghan 1995] defines the `EXCLUSIVE` value to the `mode` argument of `CREATE`:

> *EXCLUSIVE specifies that the server is to follow exclusive creation semantics, using the verifier to ensure exclusive creation of the target. No attributes may be provided in this case, since the server may use the target file metadata to store the createverf3 verifier.*

For examples of how to check for the existence of a file without opening it, see recommendation FIO10-C. Take care when using the rename() function.

Compliant Solution (`fdopen()`, POSIX)

For code that operates on `FILE` pointers and not file descriptors, the POSIX `fdopen()` function can be used to associate an open stream with the file descriptor returned by `open()`, as shown in this compliant solution [IEEE Std 1003.1:2013]:

```
char *file_name;
int new_file_mode;
FILE *fp;
int fd;

/* Initialize file_name and new_file_mode */

fd = open(file_name, O_CREAT | O_EXCL | O_WRONLY, new_file_mode);
if (fd == -1) {
  /* Handle error */
}

fp = fdopen(fd, "w");
if (fp == NULL) {
  /* Handle error */
}
```

Compliant Solution (Windows)

The Win32 API `CreateFile()` allows a programmer to create or open a file depending on the flags passed in. Passing in the `CREATE_NEW` flag ensures the call fails if the file already exists. This compliant solution demonstrates how to open a file for reading and writing without sharing access to the file such that the call fails if the file already exists.

```
TCHAR *file_name;
HANDLE hFile = CreateFile(file_name, GENERIC_READ | GENERIC_WRITE, 0, 0,
                          CREATE_NEW, FILE_ATTRIBUTE_NORMAL, 0);
if (INVALID_HANDLE_VALUE == hFile) {
  DWORD err = GetLastError();
  if (ERROR_FILE_EXISTS == err) {
    /* Handle file exists error */
  } else {
    /* Handle other error */
  }
}
```

Risk Assessment

The ability to determine whether an existing file has been opened or a new file has been created provides greater assurance that a file other than the intended file is not acted upon.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
FIO03-C	Medium	Probable	High	P4	L3

Automated Detection

Tool	Version	Checker	Description
Coverity	6.5	**OPEN_ARGS**	Fully implemented
LDRA tool suite	9.7.1	**44 S**	Enhanced Enforcement
PRQA QA-C	9.5	**5012**	Partially implemented

Related Vulnerabilities

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

Related Guidelines

SEI CERT C++ Coding Standard	VOID FIO03-CPP. Do not make assumptions about fopen() and file creation
ISO/IEC TR 24731-1:2007	Section 6.5.2.1, "The `fopen_s` Function"

Bibliography

[Callaghan 1995]	IETF RFC 1813 NFS Version 3 Protocol Specification
[IEEE Std 1003.1:2013]	System Interfaces: `open`
[ISO/IEC 9899:2011]	Subclause 7.21.5.3, "The `fopen` Function" Subclause K.3.5.2.1, "The `fopen_s` Function"
[Loosemore 2007]	Section 12.3, "Opening Streams"
[Seacord 2013]	Chapter 8, "File I/O"