NAME

safe_fopen_wrapper, safe_fopen_wrapper_follow, safe_fopen_no_fcreate, safe_fopen_no_fcreate_follow, safe_fcreate_fail_if_exists, safe_fcreate_replace_if_exists, safe_fcreate_keep_if_exists, safe_fcreate_keep_if_exists_follow – safely open or create a stream

SYNOPSIS

#include "safe_fopen.h"

FILE *safe_fopen_wrapper(const char * *fn*, **const char *** *mode*, **mode_t** *perms***);**
FILE *safe_fopen_wrapper_follow(const char * *fn*, **const char *** *mode*, **mode_t** *perms***);**

FILE *safe_fopen_no_create(const char * *fn*, **const char *** *mode***);**
FILE *safe_fopen_no_create_follow(const char * *fn*, **const char *** *mode***);**

FILE *safe_fcreate_fail_if_exists(const char * *fn*, **const char *** *mode*, **mode_t** *perms***);**

FILE *safe_fcreate_keep_if_exists(const char * *fn*, **const char *** *mode*, **mode_t** *perms***);**
FILE *safe_fcreate_keep_if_exists_follow(const char * *fn*, **const char *** *mode*, **mode_t** *perms***);**

FILE *safe_fcreate_replace_if_exists(const char * *fn*, **const char *** *mode*, **mode_t** *perms***);**

DESCRIPTION

The functions in the **safe_fopen** and **safe_fcreate** family are replacement functions for **fopen**(3). The functions open or create a file whose path name is *fn* and return a stream associated with the file. The functions are designed to prevent common types of symbolic link attacks where the last component of the file name is a symbolic link when creating a file and optionally when opening an existing file. It also makes the initial permissions for a newly created file, *perms*, a required parameter, so an appropriate value is likely to be used.

safe_fopen_wrapper and **safe_fopen_wrapper_follow** are meant to easy replacement functions for existing used of **fopen**(3). The knowledgeable developers can use the remaining functions to tailor the specific behavior desired for each use of **fopen**(3).

All the functions differ from **fopen**(3), in their treatment of the last component of the file name being a symbolic link. They all fail if an attempt is made to create the file when the last component is a symbolic link (just like the behavior of **open**(2) using *flags* of **O_CREAT** with **O_EXCL). safe_fopen_wrapper**, **safe_fopen_no_create**, **safe_fcreate_keep_if_exists** also fail when opening an existing file if the the last component is a symbolic link.

The functions also differ from **fopen**(3), in that any of the functions that could possibly create a file the *perms* parameter is required.

Other differences and the behavior of each function is explained below in the **Functions** section.

Parameters

fn The file name to open or create.

mode This is a mode string as specified in **fopen**(3). It consists of multiple characters specifying the behavior of the function.

perms The initial permissions for a newly created file (the actual permission assigned is further restricted by the process's umask). This is a set a bitwise-**or**'d permissions as specified in **open**(2). This parameter is required by all functions that create files, and is not implicitly derived from the process's umask as with **fopen**(3). In C++, this value has a default value of read and write for the owner (0600), and may be omitted.

Direct fopen(3) Replacement Functions

safe_fopen_wrapper

This function is meant to be a direct replacement for **fopen**(3). The behavior of **safe_fopen_wrapper** is based on the value of *mode* at run-time. If *mode* is **"r"** (open for reading), **safe_fopen_no_create** is called. Otherwise if *mode* is **"w"** (open for writing) or **"a"** (open for appending), **safe_fcreate_keep_if_exists** is called. This function differs from **fopen**(3) if the last component of the file name is a symbolic link, the functions fails by returning **NULL** with *errno* set to **EEXIST**.

safe_fopen_wrapper_follow
This function is the same as **safe_fopen_wrapper** except it attempts to follow a symbolic link if it is a the last component of the file name and the file exists.

Advanced fopen(3) Replacement Functions

safe_fopen_no_create
This function opens an existing file, and never creates a file if it does not exist, even with a *mode* of *"a"* or *"w"*. It also fails if the the last component of the file name is a symbolic link.

safe_fopen_no_create_follow
This function is the same as **safe_fopen_no_create** except it attempts to follow a symbolic link if it is a the last component of the file name and the file exists.

safe_fcreate_fail_if_exists
This function attempts to create the file. It fails if the file already exists or the last component of the file name is a symbolic link by returning **NULL** with *errno* set to **EEXISTS**.

safe_fcreate_keep_if_exists
This function opens a file if it exists or creates it if it does not exist. This function differs from **fopen**(3) in that it fails if the last component is a symbolic link.

safe_fcreate_keep_if_exists_follow
This function is the same as **safe_fcreate_keep_if_exists** except it attempts to follow a symbolic link if it is a the last component of the file name and the file exists.

safe_fcreate_replace_if_exists
This function always creates a new file, first removing the directory entry that *fn* refers. The function fails if it cannot perform this task.

Path manipulation detection and warning facility

A path manipulation of *fn* is detected when the file system object referred to by *fn* changes to a different file system object during the operation of these functions.

If a path manipulation is detected, the functions retry their operations a set number of times until a path manipulation is not detected. After a set number of times, they return **NULL** with *errno* set to **EAGAIN**.

All of the open replacement functions also support registering a call back function with **safe_open_register_path_warning_callback**(3). Each time a path manipulation is detected, the callback function is called. If the callback returns a non-zero result the function immediately fails by returning **NULL** to the caller.

RETURN VALUE

When successful, these functions return the stream associated with the file at the path name, *fn*, and *errno* is unchanged. If an error occurs, **NULL** is returned and *errno* is set to an appropriate value.

ERRORS

Depending on the function *errno* may be set any of the possible values that **open**, **fdopen**, **lstat**, **ftruncate**, **fstat**, **fstat**, and **unlink** may use if they fail. Other than the standard behavior of **fopen**(3), the following errors may occur that would not in **fopen**(3).

EINVAL
fn is **NULL** or *mode* is invalid.

EEXIST
The last component of *fn* is a symbolic link when the file needs to be created in **safe_fopen_wrapper**, **safe_fopen_wrapper_follow**, **safe_fcreate_fail_if_exists**, **safe_fcreate_keep_if_exists** and **safe_fcreate_keep_if_exists_follow**. It is also returned in the case of opening an existing file where the last component of the file name, *fn*, is a symbolic link in the functions **safe_fopen_wrapper**, **safe_fopen_no_create** and **safe_fcreate_keep_if_exists**.

EPERM

If the function **safe_fcreate_replace_if_exists** cannot remove *fn*.

EAGAIN

An active attack is occurring on the filename and the function was unable to safely perform the requested action after repeated attempts. All functions can return this except **safe_fcreate_fail_if_exists**. The application may retry the function and should eventually succeed.

AUTHOR

James A. Kupsch

SEE ALSO

safe_open_wrapper(3), **fopen**(3), **open**(2), **fdopen**(3)

"How to Open a File and Not Get Hacked", *2008 Third International Conference on Availability, Reliability and* Security (ARES)," Barcelona, Spain, March 2008, http://research.cs.wisc.edu/mist/safefile