NAME

safe_open_wrapper, safe_open_wrapper_follow, safe_open_no_create, safe_open_no_create_follow, safe_create_fail_if_exists, safe_create_replace_if_exists, safe_create_keep_if_exists, safe_create_keep_if_exists_follow − safely open or create a file

SYNOPSIS

#include "safe_open.h"

int safe_open_wrapper(const char * *fn*, **int** *flags*, **mode_t** *mode*);
int safe_open_wrapper_follow(const char * *fn*, **int** *flags*, **mode_t** *mode*);

int safe_open_no_create(const char * *fn*, **int** *flags*);
int safe_open_no_create_follow(const char * *fn*, **int** *flags*);

int safe_create_fail_if_exists(const char * *fn*, **int** *flags*, **mode_t** *mode*);

int safe_create_keep_if_exists(const char * *fn*, **int** *flags*, **mode_t** *mode*);
int safe_create_keep_if_exists_follow(const char * *fn*, **int** *flags*, **mode_t** *mode*);

int safe_create_replace_if_exists(const char * *fn*, **int** *flags*, **mode_t** *mode*);

DESCRIPTION

The functions in the **safe_open** and **safe_create** family are replacement functions for **open**(2). The functions open or create files whose path name is *fn* and return a file descriptor to 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, *mode*, a required parameter, so an appropriate value is likely to be used.

safe_open_wrapper and **safe_open_wrapper_follow** are meant to easy replacement functions for existing used of **open**(2). The knowledgeable developers can use the remaining functions to tailor the specific behavior desired for each use of **open**(2).

All the functions differ from **open**(2), 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 **O_CREAT** with **O_EXCL** (this behavior is always enforced as these two flags are always implicitly set internally in any of the functions that can create a file). **safe_open_wrapper**, **safe_open_no_create**, **safe_create_keep_if_exists** also fail when opening an existing file if the the last component is a symbolic link.

The functions also differ from **open**(2), in that in any of the functions that could possibly create a file the *mode* 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.

flags A set of flags that controls the behavior of the open. These flags are bitwise-**or**'d together as specified in **open**(2). All the flags described in **open**(2) are valid and have the same behavior with the exception of the **O_CREAT** flag. The **O_CREAT** flag is invalid if the function does not create a file, and is implicitly added to the set of flags in the functions that can create a file.

mode 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. In C++, this value has a default value of read and write for the owner (0600), and may be omitted.

Direct open(2) Replacement Functions

safe_open_wrapper

This function is meant to be a direct replacement for **open**(2). This function differs from **open**(2) if the last component of the file name is a symbolic link, the function fails with *errno* set to **EEXIST**. The behavior of **safe_open_wrapper** is based on the value of *flags* at run-time. If *flags* do not contain **O_CREAT**, then **safe_open_no_create** is called. Otherwise if flags contain **O_EXCL**, **safe_create_fail_if_exists** is called, otherwise **safe_create_keep_if_exists** is called.

safe_open_wrapper_follow

This function is the same as **safe_open_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 open(2) Replacement Functions

safe_open_no_create

This function opens an existing file. If *flags* contains **OCREAT**, the function returns -1 with *errno* set to **EINVAL**. If the the last component of the file name is a symbolic link, it also returns -1 with *errno* set to **EEXIST**.

safe_open_no_create_follow

This function is the same as **safe_open_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_create_fail_if_exists

This function is equivalent to calling **open**(2) with *flags* set to **O_CREAT** and **O_EXCL** added to *flags*, i.e. it atomically fails if the file exists and creates it otherwise.

safe_create_keep_if_exists

This function mimics a call to **open**(2) with a flag of **O_CREAT** without **O_EXCL**, i.e. create the file if it does not exist and open the file if it does exist. This function differs from **open**(2) in that it fails if the last component is a symbolic link.

safe_create_keep_if_exists_follow

This function is the same as **safe_create_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_create_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. This function is useful if the process wants to guarantee that the process created the file.

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 -1 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 -1 to the caller.

RETURN VALUE

When successful, these functions return the file descriptor of the opened file, and *errno* is unchanged. If an error occurs, -1 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**, **lstat**, **ftruncate**, **fstat**, and **unlink** may use if they fail. Other than the errors that **open**(2), can produce, the following errors may occur:

EINVAL

fn is **NULL**, or **O_CREAT** is inluded in the *flags* to the functions **safe_open_no_create** and **safe_open_no_create_follow**.

EEXIST

The last component of *fn* is a symbolic link when the file needs to be created in **safe_open_wrapper**, **safe_open_wrapper_follow**, **safe_create_fail_if_exists**, **safe_create_keep_if_exists** and **safe_create_keep_if_exists_follow**. It is also returned in the case of opening an existing where the last component of the file name, *fn*, is a symbolic link in the functions **safe_open_wrapper**, **safe_open_no_create** and **safe_create_keep_if_exists**.

EPERM

If the function **safe_create_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_create_fail_if_exists**. The application may retry the function and should eventually succeed.

AUTHOR

James A. Kupsch

SEE ALSO

open(2) **safe_open_register_path_warning_callback**(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