Time-of-check time-of-use filesystem race condition

Name: Time-of-check time-of-use filesystem race condition

Description: Separately checking the state of a file before operating on it may allow an attacker to modify the file between the two operations.

ID: cpp/toctou-race-condition

Kind: problem

Severity: warning

Precision: medium

Often it is necessary to check the state of a file before using it. These checks usually take a file name to be checked, and if the check returns positively, then the file is opened or otherwise operated upon.

However, in the time between the check and the operation, the underlying file referenced by the file name could be changed by an attacker, causing unexpected behavior.

Recommendation

Wherever possible, use functions that operate on file descriptors rather than file names (for example, `fchmod` rather than `chmod`).

For access checks, you can temporarily change the UID and GID to that of the user whose permissions are being checked, and then perform the operation. This has the effect of "atomically" combining a permissions check with the operation.

If file-system locking tools are available on your platform, then locking the file before the check can prevent an unexpected update. However, note that on some platforms (for example, Unix) file-system locks are typically *advisory*, and so can be ignored by an attacker.

Example

The following example shows a case where a file is opened and then, if the opening was successful, its permissions are changed with `chmod`. However, an attacker might change the target of the file name between the initial opening and the permissions change, potentially changing the permissions of a different file.

```
char *file_name;
FILE *f_ptr;

/* Initialize file_name */

f_ptr = fopen(file_name, "w");
if (f_ptr == NULL)  {
  /* Handle error */
}

/* ... */

if (chmod(file_name, S_IRUSR) == -1) {
  /* Handle error */
}
```

This can be avoided by using `fchmod` with the file descriptor that was received from opening the file. This ensures that the permissions change is applied to the very same file that was opened.

```c
char *file_name;
int fd;

/* Initialize file_name */

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

/* ... */

if (fchmod(fd, S_IRUSR) == -1) {
  /* Handle error */
}
```

References

- The CERT Oracle Secure Coding Standard for C: FIO01-C. Be careful using functions that use file names for identification .
- Common Weakness Enumeration: CWE-367.