Ftw and Nftw

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 8344 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• Indeterminate File/Path • TOCTOU - Time of Check, Time of Use
Software Context	• File Management
Location	• ftw.h
Description	Users of ftw() or nftw() should be careful to verify file status before performing any potentially sensitive file operations. The ftw() and nftw() functions traverse a directory tree, invoking a user-supplied function on "stat" information for each file. nftw() is like ftw() but provides additional options for controlling the traversal. These functions present some of the same vulnerabilities that exists with stat(). Often, one would perform some operation on some or all of the files visited via ftw() or nftw(). If the appropriateness of performing an operation is dependent on the file status, then this information should be independently verified to protect against TOCTOU attacks.
APIs	**Function Name** / **Comments** ftw nftw
Method of Attack	The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between

1. daisy:35-BSI (Barnum, Sean)

	the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results. An attacker could potentially change the attributes of a file or replace the file by a symbolic link in the time between ftw() or nftw() obtaining file status information, and the time when an operation is performed on that file. This could result in one operating on a file other than what was intended, which may result in a state that the attacker can exploit in some way.
Exception Criteria	
Solutions	<table><tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr><tr><td>Whenever ftw() or nftw() is used to perform a potentially sensitive operation.</td><td>To verify the validity of status information, save the information provided by ftw() or nftw(), open the file, get status information using fstat() on the open file descriptor, compare the statuses to be sure they are equivalent, then perform the file operation using the file descriptor. The preceding approach will not work if the operation cannot be performed via the file descriptor, as is the case with some</td><td>Often effective, but there may be use cases for which no perfect direct solution exists and a higher level redesign of the software approach may be needed to achieve a high level of security.</td></tr></table>

	common file operations. In this case, if one has control over where the file tree is located, files should be located on a secure filesystem that is not vulnerable to manipulation.	
Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
Generally applicable.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applicable.	Minimize the time between check and use, and perform any other sorts	Does not eliminate the underlying vulnerability but can help

		of verification that may make sense for the particular use case.	make it more difficult to exploit.
	Generally applicable.	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.
	Generally applicable.	Processes should avoid operating with greater file access privileges than necessary.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.

Signature Details	int ftw(const char *dir, int (*fn)(const char *file, const struct stat *sb, int flag), int depth);
	
	int nftw(const char *dir, int (*fn)(const char *file, const struct stat *sb, int flag, struct FTW *s), int depth, int flags);

Examples of Incorrect Code	`int fileOp(const char *file, const struct stat *sb, int flag)`
	`{`
	`if (flag & FTW_F) {`
	`// Note: attacker could change file before open() occurs`
	`int fd = open(fileName, O_APPEND);`
	`// write to file`
	`}`
	`}`
	
	`[...]`
	
	`ftw("/a/b/c", fileOp, 10);`

Examples of Corrected Code	`int fileOp(const char *file, const struct stat *sb, int flag)`
	`{`
	`if (flag & FTW_F) {`
	`int fd = open(fileName, O_APPEND);`

```
struct stat currentStat;
fstat(fd, &currentStat);
if (! statusesAreEquivalent(sb,
&currentStat)) return 1; // error
return
// write to file
}
}
[...]
ftw("/a/b/c", fileOp, 10);
```

Source References

- ITS4 Source Code Vulnerability Scanning Tool [2]
- Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, chapter 9.
- http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/ucd-ecs-95-09.pdf[3]
- http://www.cs.berkeley.edu/~bschwarz/paper.ps or http://www.google.com/search?q=cache:g9Osr93sIOEJ:www.cs.berkeley.edu/~bschwarz/paper.ps+nftw+vulnerability&hl=en&client=firefox-a

Recommended Resources

- Linux man page for ftw()[6]
- Linux man page for nftw()[7]

Discriminant Set

Operating System	• UNIX (All)
Languages	• C • C++

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.