

Re: access() is a security hole?

Source: <http://www.derkeiler.com/Mailing-Lists/FreeBSD-Security/2002-10/11827.html>

From: Don Lewis (dl-freebsd@catspoiler.org)

Date: 10/12/02

Date: Fri, 11 Oct 2002 19:02:00 -0700 (PDT)
From: Don Lewis <dl-freebsd@catspoiler.org>
To: peter.jeremy@optushome.com.au

On 12 Oct, Peter Jeremy wrote:

> On Fri, Oct 11, 2002 at 10:01:30AM -0400, Chris BeHanna wrote:
>> Perhaps the way to avoid the race is to open the file, lock it,
>>and *then* call access(), then close the file or proceed based upon
>>the result.
>>
>> Yes, I know--there's a possible race between open() and fcntl().
> ...
>> Once a process has the lock, it can call fstat() to determine if
>>the real [ug]id (which it already knows) has permission to access the
>>file in the desired manner.
>
> I can see two more critical problems:
>
> Firstly, open() options like O_TRUNC and O_CREAT will have already
> modified the file based on the e[gu]id before the later checks
> discover that the r[gu]id shouldn't have access to the file. You
> can delay the O_TRUNC by calling ftruncate() later, but there's no
> way to postpone O_CREAT.
>
> Secondly (and more seriously), open() only returns a file descriptor
> referring to the leaf file in the pathname – and file locks only
> affect that specific file. Using fstat() on the returned descriptor
> can only tell you the access permissions on that particular file, it
> can't tell you that access should have been blocked due to permissions
> on intervening directories – and the file lock won't prevent an
> attacker changing the intervening directory structure.
>
> The sequence open(),fstat(),access(),stat() and verifying that the
> inode returned by the fstat() and stat() are the same makes the race
> harder to win, but there's still a race: Someone could manage to
> swap the file back to the original between the access() and stat().
>
> It's not at all clear how to solve this in userland. In the absence
> of symlinks, you can parse the pathname, using open(),fstat(),fchdir()
> to securely get to the final pathname component. Unfortunately,

FreeBSD-Security: Re: access() is a security hole?

- > *there's no way to securely do this and handle symlinks (because you*
- > *have to use lstat() to detect a symlink and there is a gap between*
- > *the lstat() and subsequent open().*

It's worse than that because you can run into the same problems with trying to verify the directory permissions before doing mkdir(), rmdir(), link(), symlink(), etc.

- > *The only solution I can see is a new "open_as_real_user()" system call*
- > *which is identical to open(2) except that it performs all the access*
- > *checks using the processes real credentials instead of the effective*
- > *credentials. (This could potentially be done using a new O_REALUSER*
- > *flag on the existing open()).*

and all the other foo_as_real_user() syscalls. I suppose an alternative to the latter would be to open_as_real_user() the directory and create fmkdir(), frmdir(), flink(), fsymlink(), etc. syscalls that act a lot like fchown() but take the fd of the directory.

Another problem with doing a stat() and looking at the ownership and permission bits is that it is an attempt to duplicate the same checks that are in the kernel, and it fails to do the right thing when new kernel features, such as access control lists, are implemented.

To Unsubscribe: send mail to majordomo@FreeBSD.org with "unsubscribe freebsd-security" in the body of the message