Starred by 2 users

Owner:	jannh@google.com
CC:	proje...@google.com
Status:	Fixed *(Closed)*
Components:	----
Modified:	Dec 7, 2020

Deadline-90
Vendor-Linux
CCProjectZeroMembers
Severity-High
Finder-jannh
Product-Linux
Methodology-source-review
Methodology-Luck
Reported-2020-Sep-12
Fixed-2020-Nov-1
CVE-2020-29534

Issue 2089: Linux: io_uring: ->mm and ->files access across suid boundaries

Reported by jannh@google.com on Fri, Sep 11, 2020, 11:50 PM EDT Project Member

This bug report describes two separate issues. The first one can be reproduced on 5.8.8 (latest stable release); the second one only works on Linus' git master (commit 729e3d091984) and therefore luckily isn't a real security bug as long as it gets fixed before 5.9 (assuming that it really doesn't affect <=5.8).

=== First part: ->files access across suid boundaries ===
io_uring takes a non-refcounted reference to the files_struct of the process that submitted a request (relying on ->flush() for being notified before the files_struct can go away). Unfortunately, unshare_fd(), which is used by bprm_execve() via unshare_files(), doesn't know about that, and assumes that if the files_struct's refcount is 1, it is okay to keep using the old files_struct. Therefore, an attacker can cause io_uring to fiddle around in the file descriptor table of a privileged process, and in particular steal file descriptors using IORING_OP_FILES_UPDATE.

I guess this is partly my fault for having suggested weak references to files_struct to Jens a while back...

The attached reproducer "repro_588" reproduces this issue on 5.8.8 (but NOT on master), demonstrating that we can steal a file descriptor to /etc/shadow from a suid binary that has opened that file.

I'm not sure what the best fix for this would be. We could do add a boolean ->weak_refs to files_struct that means "the files_struct must not be recycled even if ->count is 1", but that obviously has the downside of being pretty ugly.

The alternative I see would be to give io_uring some extra special notification hook in unshare_fd() that lets it drop its references - that'd be less code, but it'd be a bigger breakage of abstraction layers. But maybe that's less bad?

=== Second part: ->mm access across suid boundaries ===
The preceding PoC does not work on git master - when the IORING_OP_FILES_UPDATE op is executed, it fails because it tries to read from the ->mm of the suid binary. I haven't debugged this properly, but what I think is happening is that the IORING_OP_FILES_UPDATE work is scheduled from task_work that executes in the context of the suid binary, and it grabs the ->mm pointer from task work context.

The PoC works again if you let the suid binary write the fd number to a fixed address and then use that address instead of &free_fd. The attached

"repro_master" demonstrates this.

This bug is subject to a 90 day disclosure deadline. After 90 days elapse, the bug report will become visible to the public. The scheduled disclosure date is 2020-12-11. Disclosure at an earlier date is possible if the bug has been fixed in Linux stable releases (per agreement with security@kernel.org folks).

repro_588.txt
3.6 KB View Download

repro_master.txt
3.7 KB View Download

Comment 1 by jannh@google.com on Wed, Dec 2, 2020, 11:43 PM EST Project Member

Status: Fixed (was: New)
Labels: -Restrict-View-Commit Fixed-2020-Nov-1

Whoops, I lost track of this one...

Fix commit: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=0f2122045b946241a9e549c2a76cea54fa58a7ff
Fixed in 5.10 and 5.9.3.

Comment 2 by jannh@google.com on Mon, Dec 7, 2020, 2:34 PM EST Project Member

Labels: CVE-2020-29534