New issue

Thread creation while a directory handle is open does a fchdir, affecting other threads (race condition) #23010

Closed

vinc17fr opened on Feb 18, 2025

When a thread is created while a directory handle is open, a `fchdir` is done (according to `strace` output under GNU Linux), affecting other threads. The issue can be reproduced on various platforms (GNU Linux, Android, macOS) with the following testcase, where `stat` randomly fails with a "No such file or directory" error (`ENOENT`):

```
#!/usr/bin/env perl

# Create a directory with files in it, for instance with
#   mkdir test && cd test && touch `seq 999` && cd ..
# then run this Perl script with the directory name in argument.
# But one can get failures even with an empty directory, because
# there are at least . and .. in a directory.

use strict;
use threads;

@ARGV == 1 || @ARGV == 2 or die "Usage: $0 <dir> [ <maxthreads> ]\n";
my ($dir,$maxthreads) = @ARGV;

-d $dir or die "$0: $dir is not a directory\n";

if (defined $maxthreads)
  {
    $maxthreads =~ /^\d+$/ && $maxthreads >= 1 && $maxthreads <= 32
      or die "$0: maxthreads must be an integer between 1 and 32\n";
  }
else
  {
    $maxthreads = 2;
  }

sub stat_test ($) {
```

```
  foreach my $i (1..100)
    {
      stat "$dir/$_[0]"
        or warn("$0: can't stat $_[0] ($!, i = $i)\n"), last;
    }
}

my $nthreads = 0;

sub join_threads () {
  my @thr;
  0 until @thr = threads->list(threads::joinable);
  foreach my $thr (@thr)
    { $thr->join(); }
  $nthreads -= @thr;
}

opendir DIR, $dir or die "$0: opendir failed ($!)\n";
while (my $file = readdir DIR)
  {
    $nthreads < $maxthreads or join_threads;
    $nthreads++ < $maxthreads or die "$0: internal error\n";
    threads->create(\&stat_test, $file);
  }
closedir DIR or die "$0: closedir failed ($!)\n";
join_threads while $nthreads;

__END__

"
Example of failure:
./dir-stat2: can't stat 2 (No such file or directory, i = 18)
./dir-stat2: can't stat 6 (No such file or directory, i = 17)
"
```

In short, for each file in the directory, a thread is created, which does 100 `stat` on the file. This script uses no more than 2 worker threads at the same time by default: the main loop waits for a worker thread to terminate before a new one is created.

To reproduce the issue more easily, create a directory with many files:

```
mkdir test && cd test && touch `seq 999` && cd ..
```

Then run this script with the directory name (e.g. `test`) in argument. Failures are much more likely to occur with `strace -f` (as usual, also use the `-o` option to redirect the output to a file).

In the `strace` output under GNU Linux, one can see for instance:

```
[...]
692379 newfstatat(AT_FDCWD, "test/275",  <unfinished ...>
692373 <... openat resumed>)            = 5
```

```
692379 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=0, ...}, 0) = 0
692373 fstat(5,  <unfinished ...>
692379 newfstatat(AT_FDCWD, "test/275",  <unfinished ...>
692373 <... fstat resumed>{st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
692379 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=0, ...}, 0) = 0
692373 fchdir(4 <unfinished ...>
692379 newfstatat(AT_FDCWD, "test/275",  <unfinished ...>
692373 <... fchdir resumed>)            = 0
692379 <... newfstatat resumed>0x5557afc648b8, 0) = -1 ENOENT (No such file or directory)
692373 openat(AT_FDCWD, ".", O_RDONLY|O_NONBLOCK|O_CLOEXEC|O_DIRECTORY) = 6
692379 write(2, "./dir-stat2: can't stat 275 (No "..., 64 <unfinished ...>
[...]
```

This excerpt shows 3 `newfstatat` on the same file `test/275`. The first two succeeded, but the third one failed with `ENOENT` (No such file or directory).

I think that this bug comes from the fix of #10387 (old rt.perl.org bug 75174), `11a11ec`, which does a `fchdir` in `Perl_dirp_dup` from `sv.c`, so Perl versions since 2010 should be affected, and this issue is still present in the repository. As the current working directory is global to the process, this affects other threads. Even though the current working directory is set back to the old value, this is a race condition, which can affect real scripts (this is how I identified this bug). Note that this is a vulnerability as the directory may be an untrusted one, so really bad things could happen (even when the directory is trusted, BTW).

I could reproduce the issue under GNU Linux with several file systems (ext4, tmpfs, NFS), and also on my Android phone (using Termux) and on a macOS machine (from the cfarm project).

Past bug reports:

- Debian bug 1098226
- Linux-nfs mailing-list (I had initially blamed NFS for this issue, due to the strangeness of the error message).


vinc17fr added **Needs Triage** on Feb 18, 2025

vinc17fr on Feb 18, 2025 Author

I had copied an incorrect version of the script. This is now the correct one.

Note that as I said in a comment in the Debian bug, if I change the script to do

```
opendir DIR, $dir or die "$0: opendir failed ($!)\n";
my @files = readdir DIR;
foreach my $file (@files)
  {
    $nthreads < $maxthreads or join_threads;
    $nthreads++ < $maxthreads or die "$0: internal error\n";
```

```
    threads->create(\&stat_test, $file);
  }
closedir DIR or die "$0: closedir failed ($!)\n";
```

then the failures still occur. But if I change it to do

```
opendir DIR, $dir or die "$0: opendir failed ($!)\n";
my @files = readdir DIR;
closedir DIR or die "$0: closedir failed ($!)\n";
foreach my $file (@files)
  {
    $nthreads < $maxthreads or join_threads;
    $nthreads++ < $maxthreads or die "$0: internal error\n";
    threads->create(\&stat_test, $file);
  }
```

(i.e. moving the `closedir` before the loop), then the failures no longer occur (and the `strace` output shows that the `fchdir` is no longer done). So the failures occur only when the directory handle is open, explaining what I wrote above.

Leont on Feb 18, 2025 Contributor •••

> I think that this bug comes from the fix of #10387 (old rt.perl.org bug 75174), `11a11ec`, which does a fchdir in Perl_dirp_dup from sv.c,

Yeah. There's a real bug there that they were fixing, but this feels like the wrong solution. It needs a new `DIR*` structure but not a new handle. Using it from two threads would be garbage but so is using it from a forked process so that's nothing new. The important thing is that it doesn't crash.

🙁 1

jkeenan removed **Needs Triage** on Feb 18, 2025

vinc17fr on Feb 19, 2025 Author •••

> As the current working directory is global to the process, this affects other threads. Even though the current working directory is set back to the old value, this is a race condition, which can affect real scripts (this is how I identified this bug).

Well, as part of this bug, there is a second bug: in `Perl_dirp_dup`, after the first `fchdir`, if the `PerlDir_open` fails for any reason (for instance, after it has been opened, the directory has been set to `d--x--x--x` permissions (111 in octal), which allows one to chdir to this directory but not to open it again), then the current working directory is not set back to the old value, so that there isn't even a race condition in such a case. This is unusual, but could be used as an attack against some scripts.

There are other issues with the current `Perl_dirp_dup` code: `PerlDir_open(".")` may fail for good reasons (the current directory needs to be executable, but not necessarily readable), and the second `fchdir` may also fail.

Leont mentioned this on Feb 20, 2025

Clone dirhandles without fchdir #23019

carnil on May 23, 2025

Crossreference to the oss-security post from **@vinc17fr**: https://www.openwall.com/lists/oss-security/2025/05/22/2

carnil on May 23, 2025

CVE-2025-40909 was assigned for this issue according to https://www.openwall.com/lists/oss-security/2025/05/23/1

github-actions mentioned this in 2 pull requests on May 27, 2025

Ci Improvements greenbone/gvm-libs#929

Change: scan progress calculation. greenbone/gvm-libs#880

ap on May 29, 2025 Contributor

As of 5.41.13, the `fchdir`-based code path no longer exists in `Perl_dirp_dup`. Thanks for bringing this issue to our attention.

ap closed this as completed on May 29, 2025

netbsd-srcmastr added a commit that references this issue on May 29, 2025

`CVE-2025-40909 fixed in perl 5.41.13` ··· `6deebe8`

Leont on May 29, 2025 Contributor •••

I have pushed a squashed version of the fix to the CVE-2025-40909 branch (currently `918bfff`) for anyone who wants to backport it.

github-actions mentioned this on Jun 3, 2025

Change: remove gvm_http_response_stream typedef from http utils greenbone/gvm-libs#930

61 remaining items

Load more

Metadata

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests