NixOS / nix Public

Code Issues 3.4k Pull requests 575 Actions Projects Security and quality 7 Insigh

Commit 4bc5a35

3 people committed on Mar 7, 2024

```
Copy the output of fixed-output derivations before registering them

It is possible to exfiltrate a file descriptor out of the build sandbox
of FODs, and use it to modify the store path after it has been
registered.
To avoid that issue, don't register the output of the build, but a copy
of it (that will be free of any leaked file descriptor).

Co-authored-by: Theophane Hufschmitt <theophane.hufschmitt@tweag.io>
Co-authored-by: Valentin Gagarin <valentin.gagarin@tweag.io>
```

2.18-maintenance + eclairevoyant/backport-10564-to-2.18 · 2.18.9 ••• 2.18.2

1 parent f8d20e9 commit 4bc5a35

3 files changed +19 -0 lines changed

Top

Filter files…

- src
  - libstore/build
    - local-derivation-goal.cc
  - libutil
    - filesystem.cc
    - util.hh

3 files changed +19 -0 lines changed

Search within code

src/libstore/build/local-derivation-goal.cc

```
@@ -2558,6 +2558,12 @@ SingleDrvOutputs LocalDerivationGoal::registerOutputs()
2558 2558                [&](const DerivationOutput::CAFixed & dof) {
2559 2559                    auto & wanted = dof.ca.hash;
2560 2560
     2561 +                // Replace the output by a fresh copy of itself to make sure
     2562 +                // that there's no stale file descriptor pointing to it
     2563 +                Path tmpOutput = actualPath + ".tmp";
     2564 +                copyFile(actualPath, tmpOutput, true);
     2565 +                renameFile(tmpOutput, actualPath);
     2566 +
2561 2567                    auto newInfo0 = newInfoFromCA(DerivationOutput::CAFloating {
```

```
2562  2568                      .method = dof.ca.method,
2563  2569                      .hashType = wanted.type,
```

src/libutil/filesystem.cc

```
@@ -133,6 +133,12 @@ void copy(const fs::directory_entry & from, const fs::path & to, bool andDelete)
133  133      }
134  134  }
135  135
     136  +
     137  + void copyFile(const Path & oldPath, const Path & newPath, bool andDelete)
     138  + {
     139  +     return copy(fs::directory_entry(fs::path(oldPath)), fs::path(newPath), andDelete);
     140  + }
     141  +
136  142  void renameFile(const Path & oldName, const Path & newName)
137  143  {
138  144      fs::rename(oldName, newName);
```

src/libutil/util.hh

```
@@ -274,6 +274,13 @@ void renameFile(const Path & src, const Path & dst);
274  274   */
275  275  void moveFile(const Path & src, const Path & dst);
276  276
     277  + /**
     278  +  * Recursively copy the content of `oldPath` to `newPath`. If `andDelete` is
     279  +  * `true`, then also remove `oldPath` (making this equivalent to `moveFile`, but
     280  +  * with the guaranty that the destination will be “fresh”, with no stale inode
     281  +  * or file descriptor pointing to it).
     282  +  */
     283  + void copyFile(const Path & oldPath, const Path & newPath, bool andDelete);
277  284
278  285  /**
279  286   * Wrappers arount read()/write() that read/write exactly the
```

Comments 0

Please sign in to comment.