Securing Path Resolution

[`openat2()` and libpathrs]

Aleksa Sarai [SUSE]
<cyphar@cyphar.com>

Where?

- https://github.com/cyphar/linux [branch: resolveat/master]
  - [v12] posted last week, [v13] will be posted in a week or so.
  - https://lwn.net/Articles/796868/ is a good overview.
- https://github.com/openSUSE/libpathrs

Unsafe Resolution?

- `open("/foo/bar/shadow", O_RDONLY)`
  - What if `shadow` is a symlink? (Just use `O_NOFOLLOW`.)
  - What if `bar` is a symlink? *(Okay, let's sanitise the path then.)*
  - Now what if `bar` gets replaced with a symlink during resolution?
- CVE-2017-1002101. CVE-2018-15664. CVE-2019-10152. CVE-2019-11246.
  - Many more examples, and probably countless more yet-undiscovered.
- **Solution:** Add `flags` that can restrict the entire resolution process.

File Descriptor Trickery

- `open("/proc/self/fd/...")` "re-open"s fd with *different modes*.
  - Works with `O_PATH` (and is (ab)used by both LXC and runc).
- Has caused security issues in the past (CVE-2019-5736).
- **Solution:**
  - Obey trailing magic-link modes (`f_mode` is exposed in procfs)
  - Make `O_PATH` of a magic-link inherit its mode.

O_EMPTYPATH

```
openat(fd, “”, O_EMPTYPATH)
```

- `open(“/proc/self/fd/...”)` but without procfs.
- Ignored with `O_PATH` or non-empty paths.
  - Thus backwards-compatible with garbage `flags`.

openat2

```
int openat2(int dfd, const char *path,
            const struct open_how *how, size_t size);


struct open_how {
  u32 flags;           // open(2) flags
  union {
    u16 mode;          // open(2) mode
    u16 upgrade_mask; // restrict O_PATH upgrading
  };
  u16 resolve;         // RESOLVE_* flags
  // future fields go here
};
```

openat2

- `RESOLVE_NO_XDEV`: Block `vfsmount` crossings.
- `RESOLVE_NO_MAGICLINKS`: Block `nd_jump_link()` crossings.
- `RESOLVE_NO_SYMLINKS`: Block `get_link()` crossings.
- `RESOLVE_BENEATH`: Block `nd_jump_root()` and "/..".
- `RESOLVE_IN_ROOT`: Scope all root jumps to dirfd.

libpathrs

- Using `openat2(RESOLVE_IN_ROOT)` correctly is non-trivial.
  - Lots of messing around with `O_PATH`.
  - No other syscalls support `RESOLVE_IN_ROOT`.
  - How do we deal with old kernels?
- **Solution:** Rust library that provides "nice" helpers that Do The Right Thing™.
  - … and it emulates `RESOLVE_IN_ROOT` on old kernels!
  - … but this requires we port programs to use it.

What's Next?

- Get `openat2(2)` merged.
  - Still lots of open questions left:
    - Should we refine `flags`?
    - How do we go about limiting `fexecve(2)` in the future?
- Port programs to use libpathrs.
  - Requires a bit of work, since "strings as paths" no longer applies.

Discussion.

Time to break out the pitchforks!

CVE-2019-5736

- **Goal:** Clobber the host container runtime binary.
  - Container runtimes all use `prctl(PR_SET_DUMPABLE, 0)`.
- **Attack:**
  - Get container runtime to `exec("/proc/self/exe")`.
  - Attacker opens `/proc/self/exe` with O_RDONLY.
  - Get process to exit or exec (to stop using the executable).
  - Re-open the `/proc/self/exe` fd with `O_WRONLY`.