taming magic-links

Aleksa Sarai
github.com/cyphar

current status

kernel

- `openat2` merged
- draft patch for magic-link hardening

userspace

- libpathrs in alpha
  - still designing API
  - core features work
  - “safe procfs” is wip

SUSE

the why

- container processes can re-open `/proc/$pid/exe` of the container manager with `O_RDWR`
    - the current protections against this are awful
- being able to restrict reopening makes sense

SUSE

the patchset

- require magic-link reopens be a subset of the magic-link's mode
- `openat(fd, "", O_EMPTYPATH)`
- `open_how->reopen` mask for restricting reopening
- expose the reopening restrictions in `/proc/$pid/fdinfo`

SUSE

the patchset

- require magic-link reopens be a subset of the magic-link's mode
  - `O_PATH` of a regular file – allow any reopen
  - `O_PATH` of a magic-link – copy the magic-link mode
  - `O_{RDWR,{RD,WR}ONLY}` – any subset of the open mode
- reopens are based on the magic-link mode
  - … which is based on the `f_mode`

SUSE

remaining issues

hardening magic-links

- do the currently proposed magic-link mode rules make sense?
- should we continue to allow mounting on top of symlinks through procfs tricks?

magic-link semantics

- what magic-link modes make sense for directories?
- should we future proof for `REOPEN_NO_EXEC`?
- path component restrictions would be consistent but tricky…

SUSE

handling directories

- magic-link mode is based on `f_mode`
- … and with hardening we want the mode to make sense
- … however, directories do not have an associated `f_mode` bit
- … so we cannot have a 777 magic-link mode for open directories

SUSE

handling directories

- should we add an f_mode bit just for this (*atm* cosmetic) feature?
- should we even be using the f_mode here?
  - we need to add `FMODE_PATH_*` anyway …
  - we need to filter `f_mode` in `/proc/$pid/fdinfo` anyway …
- `readdir` is also not restricted, it probably should be?

SUSE

the exec bit

- currently there's no restrictions on execution, meaning you cannot create a file descriptor that:
  - cannot be `fexecve`'d (files)
  - cannot be resolved through (directories)
- should we future-proof the design so we can block this?
- should we just implement it now? what about binfmt_script?

mounting on top of symlinks

- currently you can bind-mount a non-dir on top of a symlink by mounting on top of a magic-link to a O_NOFOLLOW to the symlink
- this really seems like a bug, and makes certain kinds of hardening impossible (cannot use RESOLVE_NO_XDEV for magic-links since we want to cross a mount)
- should we just block it?

SUSE

links

github.com/cyphar/linux
magiclink/open_how-reopen

libpathrs

- what should the API look like?