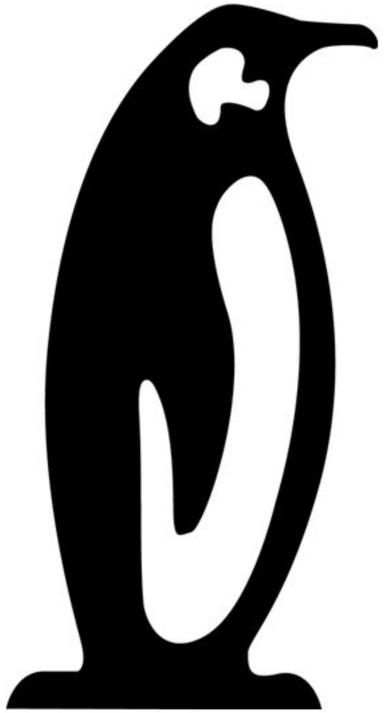

Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

seccomp × pointers

Aleksa Sarai
cyphar@cyphar.com

LINUX PLUMBERS CONFERENCE | Vienna, Austria
Sept. 18-20, 2024

seccomp limitations

```
struct open_how {
   u64 flags;
   u64 mode;
   u64 resolve;
}

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

- seccomp cannot filter pointer contents.
  - `openat2(2)` and similar syscalls are very useful to security-conscious programs but seccomp filter authors want to be able to restrict them.
- Solving this problem in general with cBPF is probably intractable.
  - However, we can try to solve this *just* for extensible struct syscalls (`clone3`, `openat2`, `mount_setattr`, …).
- This is a slightly more fleshed out version of Kees' proposal.

basic idea *hand-wavy*

syscalls

- This must be opt-in per-syscall.
- On syscall entry, if a seccomp filter is enabled, cache the structure contents so the filter and syscall see the same structure.
  - Probably stashed in `struct task` during the syscall, keyed by the structure pointer value.

filters

- `struct seccomp_data` has the cached structure contents appended.
  - Because of cBPF limitations, the information needs to be represented in a cBPF-friendly way.
- Parts of the seccomp notification API and the validator will probably need to be reworked to handle variable-sized `struct seccomp_data`.
  - Main issue is that `seccomp_data`'s size is now based on the syscall (and possibly per-call!).

syscall and `seccomp_data` changes **very hand-wavy**

syscall definitions

- Each opted-in syscall has an entry in a section (a-la `__syscalls_metadata`) which contains:
  - Which argument(s) are the pointer and size.
  - The current kernel size for that structure type.
  - (Optional) A list of the historical struct sizes if we want to make sure the “effective size” is not completely arbitrary.

`seccomp_data` extensions

- `struct seccomp_data` has following data appended:
  - (Optional) The pointer of the structure (for eBPF?).
  - A flag if there was trailing data past `ksize`.
  - The “effective size” of the structure.
  - The contents of the structure as a flat buffer.
- The “effective size” could be smaller than ksize *and* usize!
  - Smallest valid size that contains all non-zero bits.
  - Provides compatibility with old filters.
  - Not sure how the verifier could deal with this…

filters **very* hand-wavy*

```
struct seccomp_data {
  int nr;
  u32 arch;
  u64 instruction_pointer;
  u64 args[6];
  struct pointer_data {
    u64 ptr;
    u64 size;       /* <= ksize */
    bool trailing;  /* check_zeroed_user */
    u8 data[.size];
  } pointer;
};
```

- We could represent multiple pointers, but cBPF's two measly registers would result in huge filters.
  - Linus has NACKed nested pointer filtering anyway.

filters **very* hand-wavy*

```
if (data.nr == SYS_foo) {
   /* ... check arguments as normal ... */
   if (data.pointer.trailing ||
       data.pointer.size > SIZE_VER1) {
     /* filter cannot handle this version */
     return SECCOMP_RET_ERRNO(E2BIG);
   }
   if (data.pointer.size >= SIZE_VER0) {
     /* ... check v0 fields ... */
   }
   if (data.pointer.size >= SIZE_VER1) {
     /* ... check v1 fields ... */
   }
   return SECCOMP_RET_ALLOW;
}
```

- Should we implement the `trailing` logic in seccomp itself (as part of fetching the structure)?
  - Ultimately, the syscall will return `-E2BIG` anyway.
  - Tracer or notifiers might want to fake support?
- The list of struct sizes lets filters only need to handle a fixed number of known versions.
- How should we deal with the verifier…

possible extensions

- For fixed-size structure syscalls, the same design can be re-used without modifications.
- There is a planned extension for extensible-struct syscalls to do feature checking.
  - Kernel fills the structure with all valid bit patterns.
  - This needs to be done in a way that seccomp filters can unconditionally allow it.
  - RFC patchset uses the highest bit in the size.
  - What should the seccomp cache contain? Does it make sense to copy the struct in this case?

Discussion, questions, … pitchforks?

Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024