The State of Kernel Self Protection

Linux Security Summit NA
August 27, 2018
Vancouver, Canada

Kees (“Case”) Cook
keescook@chromium.org
@kees_cook

https://outflux.net/slides/2018/lss/kspp.pdf

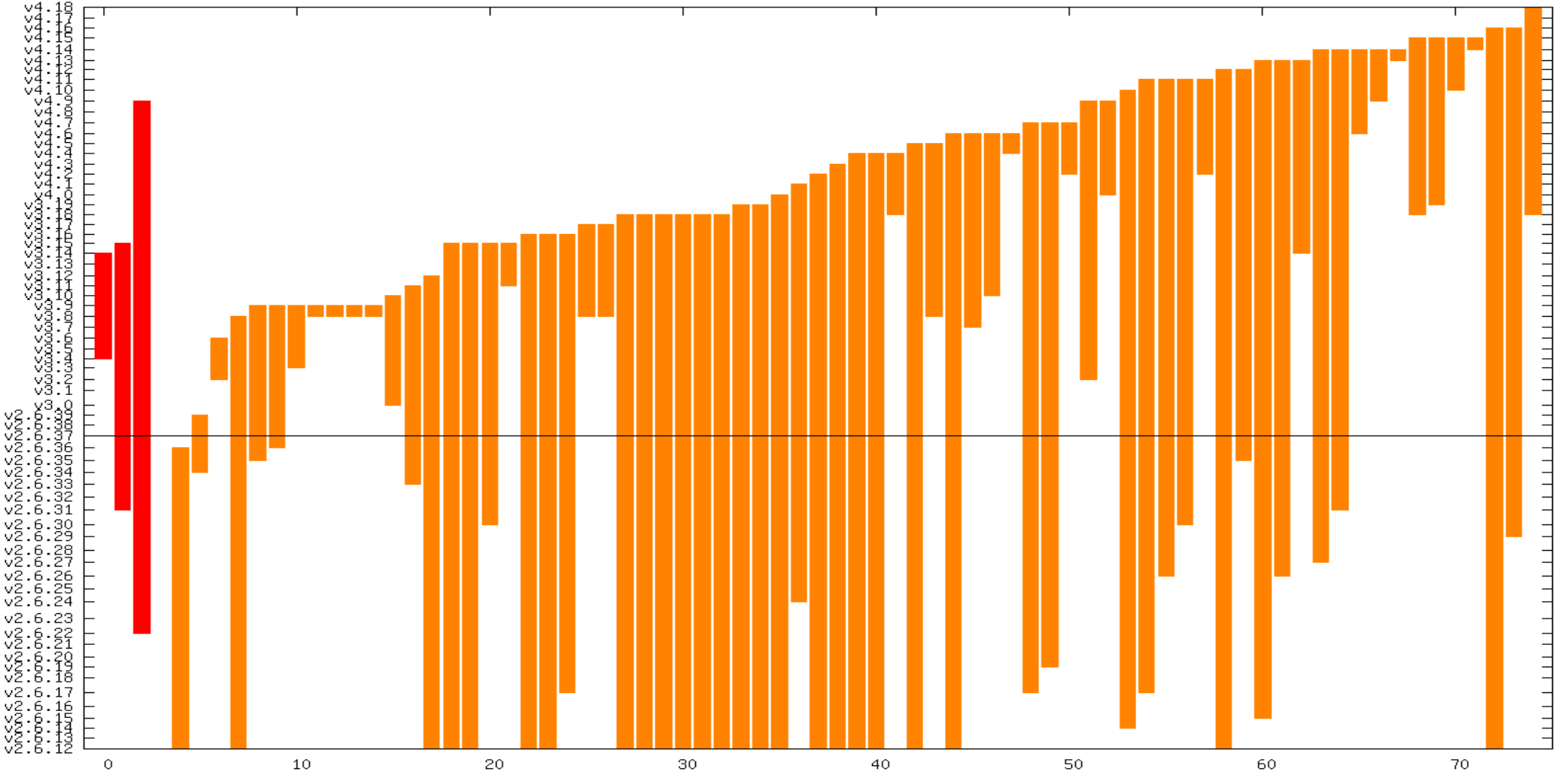
Kernel Self Protection Project

- https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project
- KSPP focuses on the kernel protecting the *kernel* from attack (e.g. refcount overflow) rather than the kernel protecting *userspace* from attack (e.g. brute force detection) but any area of related development is welcome
- Currently ~12 organizations and ~10 individuals working on about ~20 technologies
- Slow and steady

Upstream Bug Lifetime

- In 2010 Jon Corbet researched security flaws, and found that the average time between introduction and fix was about 5 years.
- My analysis of Ubuntu CVE tracker for the kernel from 2011 through 2018 has now creeped up to 6 years:
  - Critical: 3 @ 5.3 years
  - High: 71 @ 5.9 years
  - Medium: 662 @ 5.9 years
  - Low: 313 @ 5.9 years

critical & high CVE lifetimes



A year's worth of kernel releases ...

v4.14

- 3 `refcount_t` conversions (bikeshedding stall)
- randstruct plugin (automatic mode)
- SLUB freelist pointer obfuscation
- structleak plugin (by-reference mode)
- `VMAP_STACK`, arm64
- `set_fs()` removal progress
- `set_fs()` balance detection, x86, arm64, arm

v4.15

- 35 `refcount_t` conversions (32 remaining...)
- PTI, x86
- retpoline
- `struct timer_list .data` field removal
- fast refcount overflow protection, x86 (also in v4.14 -stable)
- `%p` hashing

v4.16

- 12 `refcount_t` conversions (20 more?)
- PTI, arm64
- hardened usercopy whitelisting
- `CONFIG_CC_STACKPROTECTOR_AUTO`

v4.17

- 51 VLAs removed (80 remaining...)
- Clear stack on fork
- More fixes to stack RLIMIT on exec
- `MAP_FIXED_NOREPLACE`
- Unused register clearing on syscall entry, x86
- Speculative Store Bypass Disable, x86

v4.18

- 38 VLAs removed (42 remaining...)
- Arithmetic overflow detection helpers
- Allocation overflow detection refactoring
- Speculative Store Bypass Disable, arm64

Expected for v4.19

- 33 VLAs removed (9 remaining: all in crypto API)
- Shift overflow helpers
- L1TF defenses
- Restrict `O_CREAT` for existing files and pipes in `/tmp`
- Unused register clearing on syscall entry, arm64
- Speculative Store Bypass Disable, arm64

Hopefully in v4.20

- VLAs removed completely, `-Wvla` added
- stackleak gcc plugin (x86 and arm64)

Various soon and not-so-soon features

- Link-Time Optimization
- eXclusive Page Frame Owner
- switch fallthrough marking
- SMAP emulation, x86
- brute force detection
- write-rarely memory
- memory tagging
- KASLR, arm
- Control Flow Integrity
- integer overflow detection
- per-task stack canary, non-x86
- per-CPU page tables
- read-only page tables
- {str,mem}cpy alloc size checks
- hardened slab allocator
- hypervisor magic :)

Challenges

Cultural: Conservatism, Responsibility, Sacrifice, Patience

Technical: Complexity, Innovation, Collaboration

Resources: Dedicated Developers, Reviewers, Testers, Backporters

Thoughts?

Kees (“Case”) Cook
keescook@chromium.org
keescook@google.com
kees@outflux.net

https://outflux.net/slides/2018/lss/kspp.pdf

http://www.openwall.com/lists/kernel-hardening/
http://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project