Lightweight Capability Domains: Decomposing the Linux Kernel

Charles Jacobsen  Sarah Spall  Muktesh Khole  Scott Bauer  John Regehr  Anton Burtsev

Problem
- Modern kernels are vulnerable
  ◦ Over 72 Linux kernel vulnerabilities reported this year in the CVE database
  ◦ Example: a bug in Netfilter, a module for filtering IP packets, could be remotely exploited in a denial-of-service attack by crashing the entire kernel
- Attackers are one step away from taking over a machine
- Run kernel code in isolated protection domains
- Domains communicate using message passing
- Communication is controlled with capabilities
- Each subsystem has minimal authority

Capability Access Control
- Address space isolation is not enough
  ◦ Subsystems still have access to the entire kernel
  ◦ Capabilities explicitly define all resources accessible to a subsystem
  ◦ Provide a convenient abstraction of a cross-domain pointer

Decomposing Kernel Subsystems
- Kernel shares complex object hierarchies across subsystems
- Each domain has a copy of the data
- Use capability-like identifiers to refer to objects in other domains
  (See ext3/vfs example in main figure)

Fast IPC
- Synchronous IPC is slow
  ◦ But cross-core asynchronous IPC is fast, approx. 300 cycles
  ◦ Separate control and data planes
  ◦ Control plane remains synchronous
    ◦ Slow, but backward compatible
  ◦ Data path is fast
    ◦ Requires asynchronous programming

Asynchronous Language Runtime
- Want to minimize threads and avoid traditional event-based programming with stack ripping
- Runtime for AC, an asynchronous version of C developed by Barrellfish team
- Uses cactus stacks and cooperative scheduling

Related Work
- VM containers (Bromium, Qubes, Sud, Xen disaggregation)
  ◦ Work well for application isolation
- Can host subsystems with simple interfaces (block and network devices)
- No support for decomposition of complex subsystems
- Microkernels
  ◦ Build everything from scratch
  ◦ No incremental adoption
- Kernel decomposition efforts (Sawmill, Nooks)
  ◦ Giant engineering effort
- We add language support, access control, fast IPC

Status
- Built a proof-of-concept for running kernel modules in isolation, using Intel VT-x
- IPC and capability systems nearly complete
- IDL compiler, fast ipc, AC runtime in progress