CSci 8271
Security and Privacy in Computing
Day 11: Kernel debloating and hardening

Stephen McCamant
University of Minnesota

Bloat and security risk

- Security bugs are often found in code that wasn't actually needed
  - Heartbleed, log4shell are well-known examples
- The Linux kernel has over 30MLOC, your app doesn't need it all

Static vs. dynamic reachability

- Static: what code appears reachable in the call graph
- Dynamic: what code executes under some test cases
- What to do with the large gap between them?
- This paper: apply otherwise-expensive hardening techniques

Shadow stack and CFI

- Shadow stack: save return addresses in a safe place separate from variables
  - Shadow stack location is randomized to make hard to otherwise access
- (Forward-edge) CFI: enforce legal target set for indirect jumps
  - Legal targets in a 2-level page-table-like structure

Code versions and context switching

- One version unprotected, one version hardened
- One set of dynamically reachable functions for each application and system call
  - Padding to ensure layout consistency
- Switching implemented by changing nested page tables via a hypervisor

Security and performance results

- Median 0.2%, max 4.87% of code is reachable per syscall
- 5/10 vulnerabilities and 4/5 payloads are blocked
- Syscall overhead goes from 0.43us to 3.25us
- Worst overhead is for nginx serving small files, 37%
- Redis (repeating syscalls) and SPEC CPU (fewer syscalls) are faster