Exploit Mitigation

http://outflux.net/slides/2013/arm/mitigation.pdf

ARM Summit, Edinburgh 2013

Kees Cook <keescook@google.com>
(pronounced “Case”)

Overview

- Classic Attack Structure
- Fuzzing and Static Analysis
- Page Permission Mitigations
- Reduced Target Exposure
- Address Space Layout Randomization
- Existing Infrastructure
- Existing Work

Classic Attack Structure

- Find arbitrary write bug
  - Endless stream of CVEs
- Insert malicious code into address space
  - Local userspace address? Stack? Heap? Remote packet reception?
- Redirect execution flow
  - Return from function, close a socket, send a packet, whatever
- Run malicious code
  - commit_creds(prepare_kernel_creds(NULL))
- Clean up
  - Reset locks, fix overwritten structures, etc
- Example
  - http://www.vsecurity.com/download/tools/linux-rds-exploit.c

Fuzzing and Static Analysis

- Trinity
  - http://codemonkey.org.uk/projects/trinity/
  - Could use even more smarts added for various kernel interfaces
  - Helpful to run on real hardware
- Custom hardware
  - Facedancer for USB
    - http://goodfet.sourceforge.net/hardware/facedancer21/
- smatch
- coccinelle

Page Permission Mitigations

- Do not write RO data (R, no W)
- Do not execute data (R or RW, neither X)
- Do not allow code to be written (RX, no W)
- Do not execute userspace from kernel
  - x86: SMEP, ARM: PXN. Can be emulated.
- Do not read/write userspace from kernel
  - x86: SMAP, ARM: none. Can be emulated.

Reducing Target Exposure

- Certain data of the kernel are only written at initialization time and/or very rarely
- Make these read-only (and if we must, allow for limited and well-known runtime exceptions)
  - Various function tables, e.g. x86 IDT
  - ARM vectors mapping is RW in kernel
- Largely unexplored in upstream

Address Space Layout Randomization

- Disrupts finding where to write and execute
  - Statistical defense
  - Very vulnerable to information leaks
- Well established in userspace
  - Stack, Mmap (large heap, shared objects, “PIC”), Brk (heap), Text (“PIE”)
- Kernel ASLR
  - Now: Text (x86)
  - Soon: modules, kmalloc, vmalloc
  - Need help: Text(ARM)

Existing Infrastructure

- x86, s390: set_memory_ro(), set_memory_nx()
  - CONFIG_DEBUG_RODATA
  - CONFIG_DEBUG_SET_MODULE_RONX
- equivalent of CONFIG_X86_PTDUMP
  - x86: /sys/kernel/debug/kernel_page_tables
  - ARM via /dev/mem:
    - http://grsecurity.net/~spender/kmaps-arm-v6.c
    - http://grsecurity.net/~spender/kmaps-arm-lpae.c

Existing Work

- Larry Bassel, Laura Abbott: CONFIG_STRICT_MEMORY_RWX
  - RFC for initial R, RW, RX support
  - http://lists.infradead.org/pipermail/linux-arm-kernel/2013-October/203261.html
- grsecurity/PaX
  - Full RO, RW, RX, “KERNEXEC”, and “UDEREF”
  - http://forums.grsecurity.net/viewtopic.php?f=7&t=3292

Questions?

http://outflux.net/slides/2013/arm/mitigation.pdf

keescook@{chromium.org,google.com}

kees@outflux.net