PURDUE
UNIVERSITY®

CS 50011: Introduction to Systems II

Lecture 5: Low(er) Level Security

Prof. Jeff Turkstra

Lecture 06

- Virtualization
- Security-relevant features of x86_64
- Stack canaries
- Data execution prevention

Rings

- Already covered privilege levels
- “Hierarchical protection domains”
- Indicated by CPL in CS and SS segment registers
- Ring 0-only instructions

Virtualization

- User code in ring 3
  - Both for host, and guest
  - Can still set up page table entries (more later)
- Problem lies with guest kernel
  - Cannot run in ring 0 – access to entire host
- Full software emulation
  - All code run by the guest is analyzed and transformed

- QEMU has a recompiler, VirtualBox uses it (sometimes)
  - When guest code disables interrupts and VirtualBox doesn’t know when they’ll be switched back on
  - Real-mode or protected-mode code (BIOS, DOS, OS startup)
  - Certain instructions (eg, LIDT – load interrupt descriptor table)

- Paravirtualization
  - Only specially modified OSes are permitted to run
  - Pass most of the privileged execution on to the hypervisor
  - Guest OS must be modified

VirtualBox

- Guest code: ring 3, unmodified
  - “Raw mode”
- “Nasty trick” for ring-0 code
  - Reconfigures the guest so ring 0 code runs in ring 1 instead
  - Not normally used
  - Allows hypervisor to trap privileged instructions (and I/O registers)
    - “Real” ring 0 then takes over

- Downsides
  - Ring 1 causes lots of additional instruction faults
    - No privileged instructions
  - VMM must step in each time
  - Ring 1 actually has flaws
    - LGDT/SGDT, LIDT/SIDT, POPF/PUSHF
      - “Load” is privileged, “store” is not
      - CPUID too
  - Hypervisor reserves part of guest’s address space for its own use

- SYSENTER always transitions to ring 0
  - Must trap and emulate the instruction
- Etc

VirtualBox CSAM

- Code Scanning and Analysis Manager
  - Disassembles guest code
- Patch Manager (PATM)
  - Runtime replacement
- Before ring 0, CSAM scans recursively to identify problem instructions
  - PATM in-situ patching replaces the instruction with a jump to hypervisor memory
  - Arguably as advanced as a recompiler

VT-x

- VMX root mode
  - CPU operates as usual
- non-root mode
  - Virtual Machine Control Structure (VMCS) now controls CPU operation
    - Still four rings, but instruction behavior significantly different
  - Guest OSes run here

- “VM entry” root to non-root
- “VM exit” non-root to root
- Guest and host state area that is saved/restored on entry/exit
- VMCS control which operations cause VM exits

- Why?
  - Guest has its own complete address space (not shared with hypervisor)
  - Guest kernel runs in ring 0
    - SYSENTER works fine, for example
  - I/O, certain instructions still cause VM exit

AMD-V

- Slightly more complete virtualization environment
  - Doesn’t require non-root code to run with paging enabled
  - Can run protected mode and real-mode software
    - Usually only firmware and OS loaders

AES instruction set

- ASENC: One round
- AESNCLAST: Last round
- AESDEC: One round decryption
- AESDECLAST: Last round decryption
- AESKEYGENASSIST: Assist in AES round key generation
- AESIMC: AES Inverse Mix Columns
- PCLMULQDQ: Carryless multiply

- cryptsetup benchmark

RDRAND

- On-chip random number generator
- “Intel Secure Key”
- DRND – Digital random number generator
- Pairs of 256-bit raw entropy samples from “hardware entropy source” applied to AES in CBC-MAC mode
- Reduces it to a single 256-bit conditioned sample, used as the seed

- NIST SP 800-90A
- Maximum of 511 128-bit samples before seed is changed
- RDSEED instruction
  - Newer, intended as entropy source for software PRNGs
  - Thermal noise-based
- How do you know it's random enough?

Microcode updates

- Contents entirely undocumented
- Update procedure is documented
- Encrypted

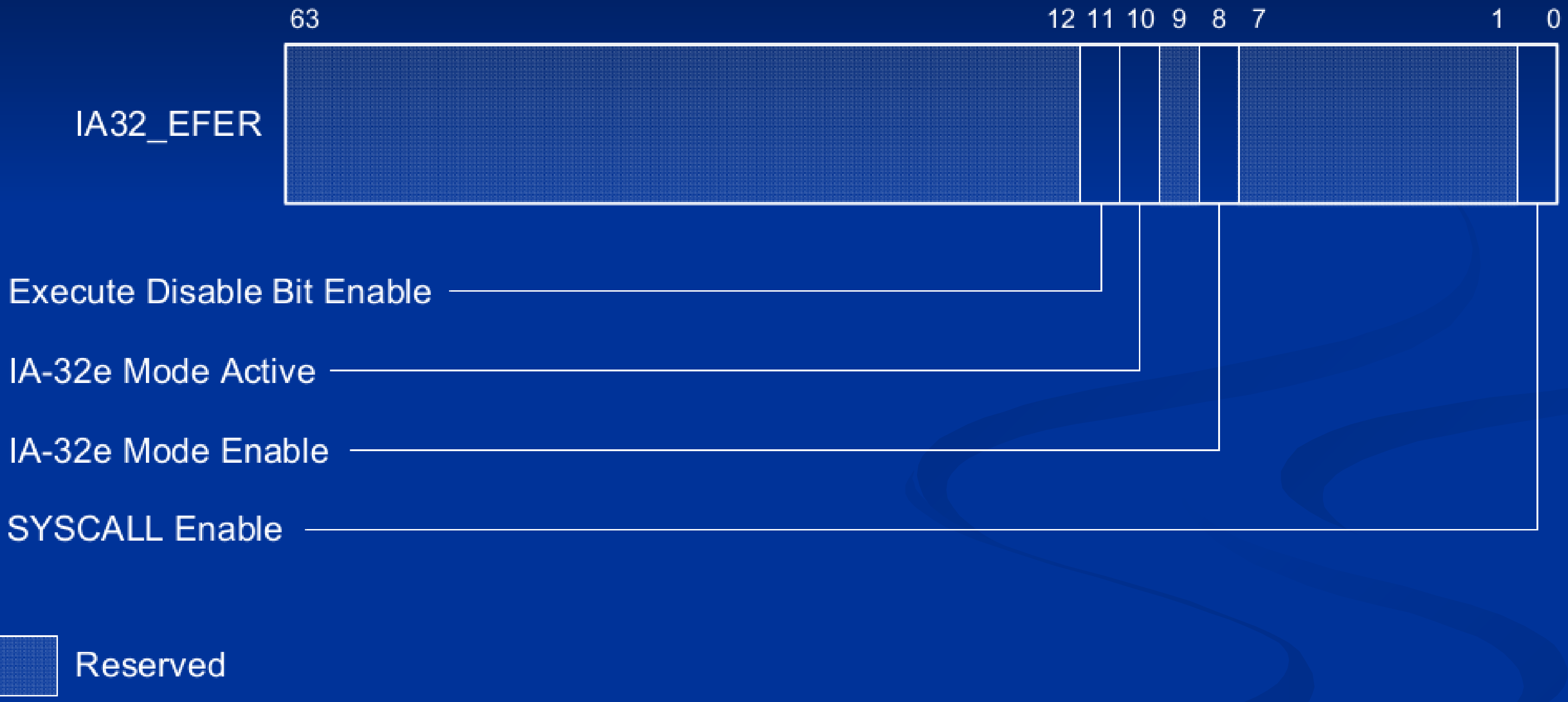
Executing where we shouldn't

- Since 80286, preventable with segmentation
  - Most OSes: we want it flat

NX bit

- No-eXecute
- Segregate code from other storage
  - Historically found on Harvard architectures
- Mark certain areas of memory as non-executable
  - Limits effectiveness of buffer overflow attacks
    - To a degree, return-to-libc

Execute disable



- Bit number 63 for 64-bit page tables
  - 1 = no execute, 0 = execute
- Not on x86’s original 32-bit page tables
  - But we can do it in software

Exec Shield

- Red Hat, Ingo Molnar
- Approximates NX emulation by tracking upper code segment limit
  - Cannot protect pages below the limit
  - mprotect() higher memory (like the stack)? Everything below it is now executable

PaX

- grsecurity
- NX emulation and a whole bunch of other stuff
- Linus refuses to merge it
    - For good reason

RSBAC

- Rule Set Based Access Control
- Same issue
- Last major update, 9/13/2016
- Similar to SELinux

AppArmor

- Kernel module
- Alternative to SELinux
- Filesystem agnostic
- Included in Ubuntu
- Owned by SUSE
  - Well, trademark at least

Stack canaries

- Example
- -fstack-protector-all

Address space layout randomization

- Mitigates against return-to-libc style attacks
  - Requires reliably locating relevant function(s)
- ASLR changes location of executable, stack, heap, libraries
- pmap example

Questions?