CS 290
Host-based Security and Malware

Christopher Kruegel

chris@cs.ucsb.edu

Operating Systems Security

Operating Systems

- Multi-user operating systems
- Operating system functionality
  - process management
  - (virtual) memory management
  - file system management
  - I/O management
- Structure
  - operating system kernel
  - user-space programs (daemons, applications, shell)

Operating Systems

- Why do we care about operating systems (OS) security
  - protect different applications that run at the same time
  - applications may belong to different users, have different privileges
  - keep buggy/malicious apps. from crashing each other
  - keep buggy/malicious apps. from tampering with each other
  - keep buggy/malicious apps. from crashing the OS
- OS provides security services
  - isolation (between processes)
  - access control (regulates who can access which resources)

Operating Systems

- Kernel
  - provides an hardware abstraction layer for user-space programs
  - complete access to all (physical) resources
  - trusted computing base

- Dual mode operation
  - hardware (processor) support
  - when in kernel-mode, can do anything (direct hardware access)
  - when in user-mode, restricted access
  - typically, mode of operation is indicated by processor status bit(s)
  - of course, this bit can only be directly manipulated in kernel-mode

Operating Systems

Transition between different modes

– this crosses the border between two security domains
– clearly, a security relevant action

- System calls

– performs a transition from user mode to privileged (kernel) mode
– usually implemented with hardware (processor) support
    - processor interrupt (int 0x80)
    - x86 call gates (far call)
    - fast system call features (sysenter)

– ensure that only specific kernel code can be invoked
    - why not allow arbitrary calls into kernel code?

Operating Systems

- System calls
  - need to check arguments for correctness
  - sometimes involves copying data from user program to kernel
  - opens up the problem of race condition bugs

- Hardware Interrupts / Exception
  - transition from user to kernel mode
  - in response to program misbehavior
    - illegal memory access, illegal instruction, …

- Kernel -> User
  - for starting process (allocate memory, load code + data from file)
  - load registers, clear privilege bits, return to code

Operating Systems

- Memory protection
  - through virtual memory abstraction
  - every process gets its own virtual memory space
  - no direct access to physical memory
  - page tables and memory MMU perform translation

- Programs are isolated and cannot talk to each other directly

- Inter-process communication
  - in some cases, shared memory can be requested
  - pipes, messages (packets) -> input validation necessary
  - file system (which is shared state) -> race conditions

Operating Systems

- Other type of memory protection
  - physical memory can also be accessed via DMA (devices attached to bus)
  - several attacks have been published based on this
    - attack of the iPods
  - idea of I/O MMU comes to rescue

Operating Systems

- Access control
  - determine the actions that a process (subject) may perform on resources (objects)
  - requires to establish "identity" of subjects
  - implemented *as access control lists (ACL)* on objects; or *capabilities* carried by subjects

- Establishing identity
  - process of authentication
  - via something that one has, that one knows, or that one is (does)
  - should be protected by a *trusted path*

Operating Systems

- Discretionary access control
  - common model for contemporary operating systems
  - subject (owner) can change permission of objects

- Mandatory access control
  - less common, but gains popularity
  - enforced by the OS when subject cannot change permissions of objects
  - often associated with multi-level security (MLS) systems and the Bell-LaPadula model