Operating Systems
(1DT020 & 1TT802)

Lecture 15

Protection and Security

May 20, 2009

Léon Mugwaneza

http://www.it.uu.se/edu/course/homepage/os/vt09
Protection vs Security

- **Protection**: one or more mechanisms for controlling the access of programs, processes, or users to resources
  - Page Table Mechanism
  - File Access Mechanism

- **Security**: use of protection mechanisms to prevent misuse of resources
  - Misuse defined with respect to policy
    - E.g.: prevent exposure of certain sensitive information
    - E.g.: prevent unauthorized modification/deletion of data
  - Requires consideration of the external environment within which the system operates
    - Most well-constructed system cannot protect information if user accidentally reveals password

A short introduction to protection and security

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne, others from Kubiatowicz - CS162 ©UCB Fall 2007 (University of California at Berkeley)
Protection

• Basic protection achieved through **dual mode operation** and **address translation**

• To Assist with Protection, **Hardware provides at least two modes**: “Kernel” mode (or “protected”) and “User” mode
  – Mode set with bits in control register only accessible in kernel-mode
  – Some instructions only available in kernel mode (Privileged instructions)

  \ Intel processor actually has four “rings” of protection:
  » PL (Privilege Level) from 0 – 3 (PL0 has full access, PL3 has least)
  » Privilege Level set in code segment descriptor (CS)
  » Typical OS kernels on Intel only use PL0 (“kernel”) and PL3 (“user”)

• **Protection goals**
  – Isolate processes and kernel from one another (private address spaces)
  – Allow flexible translation that allows easy sharing between processes
  – User cannot change mode to kernel mode or modify page table mapping
  – Limited access to memory: cannot adversely effect other processes
    » Side-effect: Limited access to memory-mapped I/O operations
  – Limited access to interrupt controller (and other devices)
How does a user program request a service in the kernel?

**System call:** Voluntary procedure call into kernel

- Needs ISA instruction
  - eg. syscall or break on MIPS, INT on INTEL, trap on Motorola
  - Hardware for controlled User → Kernel transition
  - Can any kernel routine be called? No! Only specific ones.
  - System call ID encoded into system call instruction
    » Well-defined interface with kernel (ID and parameters)

**Other sources of Synchronous Exceptions:** errors

- Divide by zero, Illegal instruction, undefined instruction, Bus error (bad address, e.g. unaligned access), Segmentation Fault (address out of range), Page Fault (for illusion of infinite-sized memory)

**Interrupts are Asynchronous Exceptions**

- Examples: timer, disk ready, network, etc....
- Interrupts can be disabled, errors cannot!

**On system call, error, or interrupt:**

- Hardware enters kernel mode with interrupts disabled
- Saves PC, then jumps to appropriate handler in kernel
- For some processors (x86), processor also saves registers, changes stack, etc.

**Actual handler typically saves registers, other CPU state, and switches to kernel stack**
System calls

• What are some system calls?
  – I/O: open, close, read, write, lseek
  – Files: delete, mkdir, rmdir, truncate, chown, chgrp, ..
  – Processes: fork, exit, wait, getpid, …
  – Network: socket create, set options, …
  – Operations on shared memory segments, semaphores, other IPCs

• Are system calls constant across operating systems?
  – Not entirely, but there are lots of commonalities
  – Also some standardization attempts (POSIX)

• System Call argument passing:
  – In registers (not very much can be passed)
  – Write into user memory, kernel copies into kernel memory
    » User addresses must be translated!
    » Kernel has different view of memory than user
  – Every argument must be explicitly checked!
Security: Preventing Misuse

• Types of Misuse:
  – Accidental:
    » If I delete shell, can’t log in to fix it!
    » Could make it more difficult by asking: “do you really want to delete the shell?”
  – Intentional:
    » Doesn’t help to ask if user wants perform action

• Three Pieces to Security
  – Authentication: who the user actually is
  – Authorization: who is allowed to do what
  – Enforcement: make sure people do only what they are supposed to do

• Loopholes in any carefully constructed system:
  – Log in as super-user and you’ve circumvented authentication
  – Log in as self and can do anything with your resources; for instance: run program that erases all of your files
  – Can you trust software to correctly enforce Authentication and Authorization?
Authentication: Identifying Users

• How to identify users to the system?
  – Passwords
    » Shared secret between two parties
    » Since only user knows password, someone types correct password ⇒ must be user typing it
    » Very common techniques to keep passwords secret
      • Encrypt passwords to help hide them
        – can we trust the encryption algorithm?
      • Force them to be longer/not amenable to dictionary attack
      • Use one-time passwords, delay checking password, …
  – Smart Cards
    » Electronics embedded in card capable of providing long passwords or satisfying challenge → response queries
    » May have display to allow reading of password
    » Or can be plugged in directly; several credit cards now in this category
  – Biometrics
    » Use of one or more intrinsic physical or behavioral traits to identify someone
    » Examples: fingerprint reader, palm reader, retinal scan
    » Becoming quite a bit more common
Authorization: Who Can Do What in the system?

- **Access-right** = \(<\text{object-name}, \text{rights-set}>\)
  - where rights-set is a subset of all valid operations that can be performed on the object

- **Domain** = set of access-rights

- **Access Control Matrix**: contains all permissions in the system
  - Resources (objects) across top
    » Files, Devices, etc…
  - Domains in columns
    » A domain might be a user, a process or a group of permissions
    » E.g. opposite: User D3 can read F2 or execute F3
  - In practice, table would be huge and sparse!

- **Important issues**:
  - When are access rights checked?
  - How (and when) to revoke authorization?
    » List of revocation attached to objects or processes?
    » Expiration dates?
Authorization: Implementation Choices

• **Access Control Lists:** store permissions with object
  – Still might be lots of users!
  – UNIX limits each file to: r,w,x for owner, group, world
  – More recent systems allow definition of groups of users and permissions for each group
  – ACLs allow easy changing of an object’s permissions
    » Example: add Users C, D, and F with rw permissions to object O

• **Capability List:** each process tracks which objects it has permission to touch
  – Popular in the past, idea out of favor today
  – Consider page table: Each process has list of pages it has access to, not each page has list of processes …
  – Capability lists allow easy changing of a domain’s permissions
    » Example: you are promoted to system administrator and should be given access to all system files

• **A combination approach:** Users have capabilities (groups or roles), Objects have ACLs
  – ACLs refer to users or groups
  – Change object permission by modifying ACL
  – Change broader user permission via change in group membership
Authorization Continued

• **Principle of least privilege:** programs, users, and systems should get only enough privileges to perform their tasks
  – Very hard to do in practice
    » How do you figure out what the minimum set of privileges is needed to run your programs?
  – People often run at higher privilege than necessary
    » Such as the “administrator” privilege under Windows
• **One solution: Signed Software**
  – Only use software from sources that you trust, thereby dealing with the problem by means of authentication
  – Fine for big, established firms such as Microsoft, since they can make their signing keys well known and people trust them
    » Actually, not always fine: recently, one of Microsoft’s signing keys was compromised, leading to malicious software that looked valid
  – What about new startups?
    » Who “validates” them?
    » How easy is it to fool them?
Involuntary Installation

• What about software loaded without your consent?
  – Macros attached to documents (such as Microsoft Word)
  – Active X controls (programs on web sites with potential access to whole machine)
  – Spyware included with normal products

• Active X controls can have access to the local machine
  – Install software/Launch programs

• Sony Spyware [Sony XCP] (October 2005)
  – About 50 CDs from Sony automatically installed software when you played them on Windows machines
    » Called XCP (Extended Copy Protection)
    » Modify operating system to prevent more than 3 copies and to prevent peer-to-peer sharing
  – Side Effects:
    » Reporting of private information to Sony
    » Hiding of generic file names of form $sys_xxx; easy for other virus writers to exploit
    » Hard to remove (crashes machine if not done carefully)
  – Vendors of virus protection software declare it spyware
    » Computer Associates, Symantec, even Microsoft
Enforcement

• **Enforcer checks passwords, ACLs, etc**
  – Makes sure the only authorized actions take place
  – Bugs in enforcer $\Rightarrow$ things for malicious users to exploit

• **In UNIX, superuser can do anything**
  – Because of coarse-grained access control, lots of stuff has to run as superuser in order to work
  – If there is a bug in any one of these programs, you lose!

• **Paradox**
  – Bullet-proof enforcer
    » Only known way is to make enforcer as small as possible
    » Easier to make correct, but simple-minded protection model
  – Fancy protection
    » Tries to adhere to principle of least privilege
    » Really hard to get right
State of the World

State of the World in Security
- Authentication: Encryption
  » But almost no one encrypts or has public key identity
- Authorization: Access Control
  » But many systems only provide very coarse-grained access
  » In UNIX, need to turn off protection to enable sharing
- Enforcement: Kernel mode
  » Hard to write a million line program without bugs
  » Any bug is a potential security loophole!

Some types of security problems
- Abuse of privilege
  » If the superuser is evil, we’re all in trouble can’t do anything
  » What if sysop in charge of instructional resources went crazy and deleted everybody’s files (and backups)?
- Imposter: Pretend to be someone else
  » Example: in unix, can set up an .rhosts file to allow logins from one machine to another without retyping password
  » Allows “rsh” command to do an operation on a remote node
  » Result: send rsh request, pretending to be from trusted user install .rhosts file granting you access
Other Security Problems

• **Virus**: A piece of code that attaches itself to a program or file so it can spread from one computer to another, leaving infections as it travels
  – Most attached to executable files, so don’t get activated until the file is actually executed
  – Once caught, can hide in boot tracks, other files, OS

• **Worm**: Similar to a virus, but capable of traveling on its own
  – Takes advantage of file or information transport features
  – Because it can replicate itself, your computer might send out hundreds or thousands of copies of the worm

• **Trojan Horse**: Named after huge wooden horse in Greek mythology given as gift to enemy; contained army inside
  – At first glance appears to be useful software but does damage once installed or run on your computer

• **Stack and Buffer-overflow**
  – input comes from network request and is not checked for size
  – Allows execution of code with same privileges as running program!
    👍 New mode bits in processors
      » Put in page table; says “don’t execute code in this page”
Summary

• Protection: Prevent unauthorized Sharing of resources
  – Address space protected using translation of addresses through Memory Management Unit (MMU)
    » Every Access translated through page table
    » Changing of page tables only available to kernel
  – Dual-Mode
    » Kernel/User distinction: User restricted
    » User→Kernel: System calls (+ error or Interrupts)
    » Inter-process communication: shared memory, or through kernel (system calls)

• Security : prevent misuse
  – User Identification
    » Passwords/Smart Cards/Biometrics
    » Encrypt password to help hid them
    » Force passwords to be longer/not amenable to dictionary attack
  – Authorization
    » Abstract table of users (or domains) vs permissions
    » Implemented either as access-control list or capability list