

Operating System Security

CS 239

Computer Security

January 13, 2003

CS 239, Winter 2003

Lecture 3
Page 1

Outline

- Introduction
- Memory protection
- Interprocess communications protection
- File protection

CS 239, Winter 2003

Lecture 3
Page 2

Introduction

- Operating systems provide the lowest layer of software visible to users
- Operating systems are close to the hardware
 - Often have complete hardware access
- If the operating system isn't protected, the machine isn't protected
- Flaws in the OS generally compromise all security at higher levels

CS 239, Winter 2003

Lecture 3
Page 3

Why Is OS Security So Important?

- The OS controls access to application memory
- The OS controls scheduling of the processor
- The OS ensures that users receive the resources they ask for
- If the OS isn't doing these things securely, practically anything can go wrong
- So almost all other security systems must assume a secure OS at the bottom

CS 239, Winter 2003

Lecture 3
Page 4

Single User Vs. Multiple User Machines

- The majority of today's computers usually support a single user
 - Sometimes one at a time, sometimes only one ever
- Some computers are still multi-user
 - Mainframes
 - Servers
 - Network-of-workstation machines
- Single user machines often run multiple processes, though

CS 239, Winter 2003

Lecture 3
Page 5

Server Machines Vs. General Purpose Machines

- Most server machines provide only limited services
 - Web page access
 - File access
 - DNS lookup
- Security problems are simpler for them
- Some machines still provide completely general service, though
- And many server machines can run general services . . .

CS 239, Winter 2003

Lecture 3
Page 6

Downloadable Code and Single User Machines

- Applets and other downloaded code should run in a constrained mode
- Using access control on a finer granularity than the user
- Essentially the same protection problem as multiple users

CS 239, Winter 2003

Lecture 3
Page 7

Mechanisms for Secure Operating Systems

- Most operating system security is based on separation
 - Keep the bad guys away from the good stuff
 - Since you don't know who's bad, separate most things

CS 239, Winter 2003

Lecture 3
Page 8

Separation Methods

- Physical separation
 - Different machines
- Temporal separation
 - Same machine, different times
- Logical separation
 - HW/software enforcement
- Cryptographic separation

CS 239, Winter 2003

Lecture 3
Page 9

The Problem of Sharing

- Separating stuff is actually pretty easy
- The hard problem is allowing controlled sharing
- How can the OS allow users to share exactly what they intend to share?
 - In exactly the ways they intend

CS 239, Winter 2003

Lecture 3
Page 10

Levels of Sharing Protection

- None
- Isolation
- All or nothing
- Access limitations
- Limited use of an object

CS 239, Winter 2003

Lecture 3
Page 11

Protecting Memory

- Most general purpose systems provide some memory protection
 - Logical separation of processes that run concurrently
- Usually through virtual memory methods
- Originally arose mostly for error containment, not security

CS 239, Winter 2003

Lecture 3
Page 12

Security Aspects of Paging

- Main memory is divided into page frames
- Every process has an address space divided into logical pages
- For a process to use a page, it must reside in a page frame
- If multiple processes are running, how do we protect their frames?

CS 239, Winter 2003

Lecture 3
Page 13

Protection of Pages

- Each process is given a page table
 - Translation of logical addresses into physical locations
- All addressing goes through page table
 - At unavoidable hardware level
- If the OS is careful about filling in the page tables, a process can't even name other processes' pages

CS 239, Winter 2003

Lecture 3
Page 14

Security Issues of Page Frame Reuse

- A common set of page frames is shared by all processes
- The OS switches ownership of page frames as necessary
- When a process acquires a new page frame, it used to belong to another process
 - Can the new process read the old data?

CS 239, Winter 2003

Lecture 3
Page 15

Special Interfaces to Memory

- Some systems provide a special interface to memory
- If the interface accesses physical memory,
 - And doesn't go through page table protections,
- Attackers can read the physical memory
 - Then figure out what's there and find what they're looking for

CS 239, Winter 2003

Lecture 3
Page 16

Protecting Interprocess Communications

- Operating systems provide various kinds of interprocess communications
 - Messages
 - Semaphores
 - Shared memory
 - Sockets
- How can we be sure they're used properly?

CS 239, Winter 2003

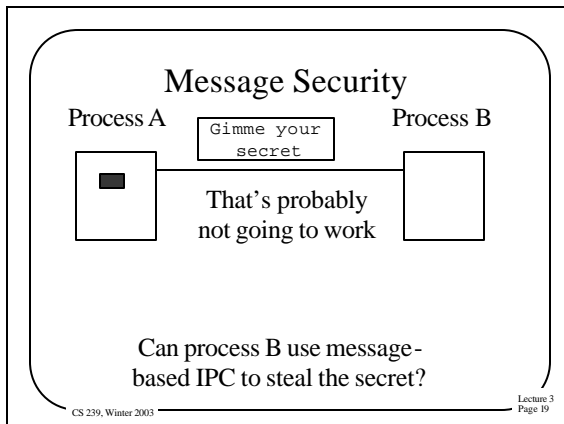
Lecture 3
Page 17

IPC Protection Issues

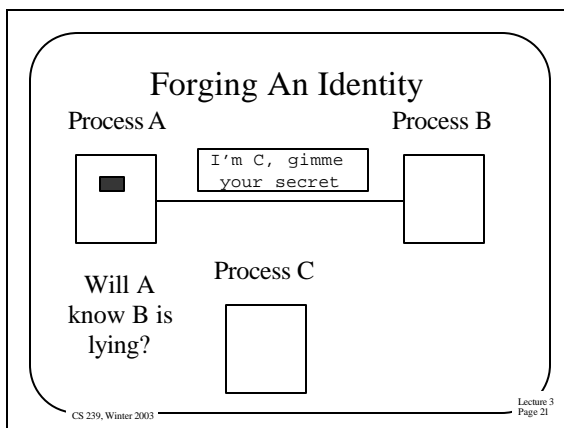
- How hard it is depends on what you're worried about
- For the moment, let's say we're worried about one process improperly using IPC to get info from another
 - Process A wants to steal information from process B
- How would process A do that?

CS 239, Winter 2003

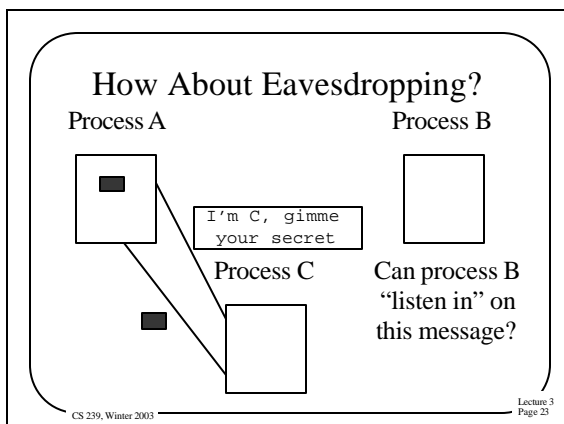
Lecture 3
Page 18



- ### How Can B Get the Secret?
- He can convince the system he's A
 - A problem for authentication
 - He can break into A's memory
 - That doesn't use message IPC
 - And is handled by page tables
 - He can forge a message from someone else to get the secret
 - He can "eavesdrop" on someone else who gets the secret
- Lecture 3
Page 20



- ### Operating System Protections
- The operating system knows who each process belongs to
 - It can tag the message with the identity of the sender
 - If the receiver cares, he can know the identity
- Lecture 3
Page 22



- ### What's Really Going on Here?
- On a single machine, what is a message send, really?
 - A message is copied from a process buffer to an OS buffer
 - Then from the OS buffer to another process' buffer
 - If attacker can't get at processes' internal buffers and can't get at OS buffers, he can't "eavesdrop"
- Lecture 3
Page 24

Other Forms of IPC

- Semaphores, sockets, shared memory, RPC
- Pretty much all the same
 - Use system calls for access
 - Which belong to some process
 - Which belongs to some principal
 - OS can check principal against access control permissions at syscall time

CS 239, Winter 2003

Lecture 3
Page 25

So When Is It Hard?

- Always possible that there's a bug in the operating system
 - Allowing masquerading, eavesdropping, etc.
 - Or, if the OS itself is compromised, all bets are off
- What if the OS has to prevent cooperating processes from sharing information?

CS 239, Winter 2003

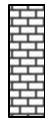
Lecture 3
Page 26

The Hard Case

Process A



Process B



Process A wants to tell the secret to process B
But the OS has been instructed to prevent that
Can the OS prevent A and B from colluding
to get the secret to B?

CS 239, Winter 2003

Lecture 3
Page 27

Dangers for Operating System Security

- Bugs in the OS
 - Not checking security, allowing access to protected resources, etc.
- Privileged users and roles
 - Superusers often can do anything
- Untrusted applications and overly broad security domains

CS 239, Winter 2003

Lecture 3
Page 28

File Protection

- How do we apply these access protection mechanisms to a real system resource?
- Files are a common example of a typically shared resource
- If an OS supports multiple users, it needs to address the question of file protection

CS 239, Winter 2003

Lecture 3
Page 29

Unix File Protection

- A model for protecting files developed in the 1970s
- Still in very wide use today
 - With relatively few modifications
- But not very flexible

CS 239, Winter 2003

Lecture 3
Page 30

Unix File Protection Philosophy

- Essentially, Unix uses a limited ACL
- Only three subjects per file
 - Owner
 - Group
 - Other
- Limited set of rights specifiable
 - Read, write, execute
 - Special meanings for some file types

CS 239, Winter 2003

Lecture 3
Page 31

Unix Groups

- A set of Unix users can be joined into a group
- All users in that group receive common privileges
 - Except file owners always get the owner privileges
- A user can be in multiple groups
- But a file has only one group

CS 239, Winter 2003

Lecture 3
Page 32

Setuid and Setgid

- Unix mechanisms for changing your user identity and group identity
- Either indefinitely or for the run of a single program
- Created to deal with inflexibilities of the Unix access control model
- But the source of endless security problems

CS 239, Winter 2003

Lecture 3
Page 33

Why Are Setuid Programs Necessary?

- The print queue is essentially a file
- Someone must own that file
- How will other people put stuff in the print queue?
 - Without making the print queue writeable for all purposes
- Typical Unix answer is run the printing program setuid
 - To the owner of the print queue

CS 239, Winter 2003

Lecture 3
Page 34

Why Are Setuid Programs Dangerous?

- Essentially, setuid programs expand a user's security domain
- In an encapsulated way
 - Abilities of the program limit the operations in that domain
- Need to be damn sure that the program's abilities are limited

CS 239, Winter 2003

Lecture 3
Page 35

Some Examples of Setuid Dangers

- Setuid programs that allow forking of a new shell
- Setuid programs with powerful debugging modes
- Setuid programs with interesting side effects
 - E.g., lpr options that allow file deletion

CS 239, Winter 2003

Lecture 3
Page 36

Domain and Type Enforcement

- A limited version of capabilities
- Meant to address the dangers of setuid
- Allows system to specify security domains
 - E.g., the printing domain
- And to specify data types
 - E.g., the printer type

CS 239, Winter 2003

Lecture 3
Page 37

Using DTE

- Processes belong to some domain
 - Can change domains, under careful restrictions
- Only types available to that domain are accessible
 - And only in ways specified for that domain

CS 239, Winter 2003

Lecture 3
Page 38

A DTE Example

- Protecting the FTP daemon from buffer overflow attacks
- Create an FTP domain
- Only the FTP daemon and files in the FTP directory can be executed in this domain
 - And these executables may not be written within this domain
- Executing the FTP daemon program automatically enters this domain

CS 239, Winter 2003

Lecture 3
Page 39

What Happens On Buffer Overflow?

- The buffer overflow attack allows the attacker to request execution of an arbitrary program
 - Say, /bin/sh
- But the overflowed FTP daemon program was in the FTP domain
 - And still is
- /bin/sh is of a type not executable from this domain
 - So the buffer overflow can't fork a shell

CS 239, Winter 2003

Lecture 3
Page 40

Unix File Access Control and Complete Mediation

- Unix doesn't offer complete mediation
- File access is checked on open to a file
 - For the requested modes of access
- Opening program can use the file in the open mode for as long as it wants
 - Even if the file's access permissions change
- Substantially cheaper in performance

CS 239, Winter 2003

Lecture 3
Page 41

Physical Implementation of Unix Access Control

- Effectively, requires 9 bits per file
 - Setuid and setgid adds two bits
- Stored in the file's inode
 - Possible because they're so small
- Checking them again requires re-examining the inode

CS 239, Winter 2003

Lecture 3
Page 42

Pros and Cons of Unix File Protection Model

- + Low cost
- + Simple and easy to understand
- + Time tested
- Lacking in flexibility
 - In granularity of control
 - Subject and object
 - In what controls are possible
- No complete mediation

CS 239, Winter 2003

Lecture 3
Page 43

Access Control Lists for File Systems

- The file system access control mechanism of choice in modern operating systems
- Used in many systems -
 - Andrew
 - Windows NT
 - Solaris 2.5

CS 239, Winter 2003

Lecture 3
Page 44

Windows NT ACLs for Files

- Integrated into the overall NT access control mechanism
- Uses NT concept of security descriptors
 - Specifying objects
- And security IDs
 - Specifying subjects

CS 239, Winter 2003

Lecture 3
Page 45

More On Windows NT File ACLs

- The NT model also allows creation of groups
 - With their own security IDs
- The security model is object-based
 - So the types of permissions that can be granted are flexible and extensible

CS 239, Winter 2003

Lecture 3
Page 46