Lecture Week 9
Protection in General-Purpose Operating Systems
Operating Systems

- Operating Systems and Databases present unique challenges in security.
  - Databases are covered in Chapter 7

- Modern operating systems (and even some older systems) are multiuser and multiproccess systems.
  - Program resources need to be protected from
    - Other users
    - Other programs
• The earliest systems were not multiuser nor multiprocess.
  – Resources were protected through the simple fact that all resources had to be scheduled in dedicated blocks of time.
  – Required a lot of human intervention to implement change-overs and execution.
  – No external communication
• The environment itself facilitated security at the expense of convenience and flexibility
Next step of evolution

- The first operating systems evolved from these system.
  - Really just simple utilities called “executive”
  - Designed to automate some of the routine activities performed by batch operators.

- Multiuser systems resulted but brought with them a host of security and resource problems to tackle.

- Now Security is a requirement rather than a benefit.
Protected Objects

- Multiprogramming gave rise to the need to protect many types of resources
  - Memory
  - Sharable I/O devices, such as disks
  - Serial reusable devices such as printers and tape drives
  - Sharable programs and subprocedures (libraries)
  - Networks!
  - Sharable Data (Databases)
• Sometime during this evolution the Operating Systems assumed the responsibility for controlling access to these devices.
  – Because it had to? Probably.

• Operating Systems have developed methods for providing such security...
Security Methods

- Physical Separation
  - Different processes use difference resources

- Temporal Separation
  - Different processes execute at different times

- Logical Separation
  - Maintain the illusion of Physical or Temporal Separation

- Cryptographic Separation
  - Processes conceal their data and computations so that external processes cannot gain access.

- [ Combinations are possible ]
The four methods presented on the previous slide are listed more, or less, in increasing order of complexity.

Physical and temporal separation are relatively easy to implement but can lead to poor resource utilization.

Separation isn't enough. We want to be able to allow users to share resources.
Providing Sharing

• Operating Systems use a variety of methods to enable sharing of resources
  – Do not protect
  – Isolate
  – Share all or Share nothing
  – Share via access limitations
  – Share by capabilities
  – Limit use of an object
Do not protect

- No protection is provided.
- Only appropriate when sensitive procedures are executed at separate times.
Isolation

- Different processes execute concurrently
- Processes are completely unaware of the presence of other processes.
- Each process has its own:
  - Address space
  - Files
  - Other objects
Share all or nothing

- Owner of an object declares the access as either
  - Public: Everybody has access
  - Private: Only available to its owner.
Share via access limitations

- Operating checks the allowability of each user's potential access to an object.
- Lists of rights are used to determine whether access is allowed or denied.
  - File system permissions
Share by capabilities

- An extension of access limitation
- Access rights can be dynamic
- The degree of sharing can depend on:
  - The owner
  - The subject
  - The context of computation
  - The object itself
Limit use of an object

• Controls not just the access to an object but how it is used once access has been granted.
  - (Digital Rights Management for a scary example on the horizon)
  - Users might be able to view a sensitive document but not be able to print it.
Granularity

- Access can be controlled at various levels.
- Modern operating systems continue in attempting to provide finer and finer granulation of access.
- Finer granularity typically results in an increase in the complexity to support it.
Memory Protection

• One of the most common resource that needs protection is memory. Several methods have evolved to handle protection of memory:
  - Fence
  - Relocation
  - Base/Bound
  - Tagged Architecture
  - Segmentation
  - Paging
Fence

• An early attempt at providing memory protection

• Memory is strictly divided into two separate areas by a specific memory location (or fence.
  - Very restrictive
  - Predefined an inflexible space reserved for operating system.

• Fence register: a technique used to provide an arbitrary and adjustable fenced area. Fence register is added to all addresses
Relocatable

- A predetermined offset is added to all address location in a program.
- Remember: not all computers were like they are today... memory addresses use to be hard coded regularly.
- Fence register can be used as a convenient method of supporting a single, fixed relocation mechanism
Base/Bound registers

- Fence registers do not prevent programs from going beyond their scope.
- Inclusion of an upper bounds register allows a second check to determine if a program has also gone too far.
- The base and bound registers provide an essential capability of context switching.
- Users are protected from their own programs as well as the programs of other users.
- Can be used to relocate code and data separately.
Tagged Architectures

- Systems such as IBM System/38, Burroughs B6500 and I960 match words in memory with some dedicated bits to control access.

- The setting of the bit indicates the level of access granted.

- Such systems though are difficult to justify economically especially in the presence of more flexible, general and modern protection mechanisms.
Segmentation

- Unlimited base/bounds registers
- Table of <name, offset> pairs for each segment
- OS can place any segment anywhere
- Segments can be removed when not in immediate use
- All address references pass through the operating system so they can be validated.
- Computational expensive to support
- Varying segment sizes cause memory fragmentation