ECE/CS230
Computer Systems Security

Charalambos (Harrys) Konstantinou

https://sites.google.com/view/ececs230kaust

OS

Operating systems

- An operating system allows different users to access different resources in a shared way
- The operating system needs to control this sharing and provide an interface to allow this access
- Identification and authentication are required for this access control
- We will start with memory protection techniques and then look at access control in more general terms

Outline

- Protection in general-purpose operating systems
- Access control
- User authentication
- Security policies and models
- Trusted operating system design

Outline

- Protection in general-purpose operating systems
- Access control
- User authentication
- Security policies and models
- Trusted operating system design

History

- Operating systems evolved as a way to allow multiple users use the same hardware
- OS makes resources available to users if required by them and permitted by some policy
- OS also protects users from each other
  - Attacks, mistakes, resource overconsumption
- Even for a single-user OS, protecting a user from him/herself is a good thing
  - Mistakes, malware

Back to Intro: OS

- Problems in Computing Systems can be caused
  - Non-intentionally: Buffer Overflow
  - Intentionally: Viruses and Worms
- Some programs offer security challenges (more vulnerable) beyond those in more general programs
  - Ex. Operating system and Database Programs

Back to Intro: OS

- An Operating system has two goals
  - Controlling shared access
  - Implementing an interface to allow that access
- Underneath those goals are support activities:
  - identification and authentication
  - naming
  - filing objects
  - scheduling
  - communication among processes
  - reclaiming and reusing objects

Back to Intro: OS

- **Operating Systems Functions** can be categorized by
  - access control
  - identity and credential management
  - information flow
  - audit and integrity protection
- Each of these activities has security implications
- The OS range from simple ones supporting a single task at a time to complex multiuser, multitasking systems
  - security considerations increase as OSs become more complex.

Back to Intro: OS

- An OS supports multi-programming
  - the concurrent use of a system by more than one user
- OS designers have developed ways to protect one user from the other
  - memory protection
  - file protection
  - general control of access to objects
  - user authentication

Protected objects

- Memory
- Data
- CPU
- Programs
- I/O devices (disks, printers, keyboards,...)
- Networks
- OS

Security Methods of OS: Separation

- The basis of protection is separation:
  keeping one user's objects separate from other users.
- This can occur in several ways (listed in increasing order of complexity to implement):
- Physical separation
  - Use different physical resources for different users
  - Easy to implement, but expensive and inefficient
- Temporal separation
  - Execute different users' programs at different times
- Logical separation
  - User is given the impression that no other users exist
  - As done by an operating system
- Cryptographic separation
  - Encrypt data and make it unintelligible to outsiders
  - Complex

Sharing

- Sometimes, users do want to share resources
  - Library routines (e.g., libc)
  - Files or database records
- OS should allow flexible sharing, not "all or nothing"
  - Which files or records? Which part of a file/record?
  - Which other users?
  - Can other users share objects further?
  - What uses are permitted?
    - Read but not write, view but not print (Feasibility?)
    - Aggregate information only
  - For how long?

Sharing

- Sometimes, users do want to share resources
  - Library routines (e.g., libc)
  - Files or database records
- An **OS can support separation and sharing**
  (listed in increasing order of complexity to implement, increasing order of security):
  - **Do not protect**: OSs with no protection are appropriate when sensitive procedures are being run at separate times.
  - **Isolate**: different processes running concurrently are unaware of the presence of each other. Each process has its own address space, files, and other objects.
    - The OS must confine each process somehow so that the objects of the other processes are completely concealed.
  - **Share all or share nothing**: the owner of an object declares it to be public or private.
    - A public object is available to all users, whereas a private object is available only to its owner.

Separation & Sharing

- **Share via access limitation**: The operating system checks the allow-ability of each user's potential access to an object.
  - That is, access control is implemented for a specific user and a specific object
  - The operating system acts as a guard between users and objects, ensuring that only authorized accesses occur.
- **Share by capabilities**: An extension of limited access sharing, this form of protection allows dynamic creation of sharing rights for objects.
  - The degree of sharing can depend on the owner or the subject, on the context of the computation, or on the object itself.
- **Limit use of an object**: Limits not just the access to an object but the use made of that object after it has been accessed.
  - A user may be allowed to view a sensitive document, but not to print a copy of it.
  - A user may be allowed access to data in a database to derive statistical summaries (such as average salary at a particular grade level), but not to determine specific data values (salaries of individuals).

Memory and address protection

Read more: https://flylib.com/books/en/4.270.1.46/1/

- Prevent one program from corrupting other programs or data, operating system and maybe itself
- Often, the OS can exploit hardware support for this protection, so it's cheap
- Memory protection is part of translation from virtual to physical addresses
  - Memory management unit (MMU) generates exception if something is wrong with virtual address or associated request
  - OS maintains mapping tables used by MMU and deals with raised exceptions