CSCI 3753
Operating Systems

Protection

Chapters 14

Lecture Notes By
Shivakant Mishra
Computer Science, CU-Boulder
Last Update: 11/29/16

Security and Protection

- This is a broad and deep topic
- Relevance of security to operating systems
  - Authentication: login/password
  - Authorization: once authenticated, the OS must keep track of what rights a user has to each file, object, and service
  - There are other concerns
    - Confidentiality: Sensitive data should be visible to a small set of users - encryption.
    - Availability: Malicious adversaries may wish to prevent access to some services, engaging in distributed denial-of-service attacks (DDOS).
    - Integrity: Detect whether data, e.g. in a file, has been tampered with.

Authorization

- Once a user has been password-authenticated, the OS must determine what files and services the user/process is authorized to access
  - login shell or process operates in a *protection domain* that specifies which resources it may access
  - a domain is a collection of access rights, each of which is an ordered pair <object, set of rights>
    - rights can include read, write, execute, print privileges, etc.
  - in UNIX, a domain is associated with a user
- can collect object and access rights into an *access matrix*

Access Matrix

objects

domains, e.g. users

	file F1	file F2	file F3	printer	D1	D2	D3	D4
D1	read		read			switch		
D2		owner read		print				switch control
D3		read	execute					
D4	read, write		read, write					

- A process executing in protection domain D1, e.g. as user U1, has permission to read files F1 and F3, and *switch* to another domain D2
- A process in domain D2 has *control* right to modify permissions in *row* D4 and *owner* right to modify permissions in the *column* F2

Access Matrix: Implementation

- As a single global table
  - Large, may be difficult to keep it all in memory
    - could use VM-like demand paging to keep only active portions of access matrix in memory
  - Still difficult to exploit relationships
    - e.g. changing the read access to a given file for an entire group of users - have to change each entry in the matrix
  - Difficult to compress
    - Matrix may be very sparse, with few entries filled in, yet would have to allocate space for all matrix entries

Access Matrix: Implementation

- As an *access control list* (ACL)
  - Each *column* of the access matrix defines access rights to a particular object, e.g. a file
  - Store the access permissions in an ACL with the file
    - Empty entries can be discarded, resulting in savings on space
  - When a process tries to access the file, search the ACL for the proper permissions
  - UNIX and Windows NT/2000 use a form of ACL
    - access permissions stored in the FCB
  - Determining the set of access rights across a domain is difficult, while determining the set of access rights for a given file is easy

Access Matrix: Implementation

- As a *capability list*
  - Each *row* of the access matrix defines access permissions for a particular user/domain
    - Create a capability list for each user/domain
    - The capability list is consulted whenever a process in a user domain tries to access a file
    - Also allows for compression of empty entries
  - Have to create a new data structure to store per-user capability lists
    - in comparison, ACLs exploit existing data structures, e.g. FCBs
  - Determining the set of access rights for a given file is difficult, while determining the set of access rights across a given domain is easy
  - Hydra OS and Mach OS use capability lists

UNIX Protection Mechanisms

- UNIX-style OSs implement ACLs
  - in UNIX, *ls -lg* will reveal the file permissions
    - "-rwxrwxrwxs filename" is the format returned
    - The last bit "s" is often called the *sticky bit*. If it is set, then only the owner/creator of the directory can delete or rename files. For example, /tmp often has the sticky bit set so normal users can't delete other users' files in /tmp.
  - *chmod* will change file permissions to files that the user owns
    - e.g. chmod 700 foo.txt