

Multics

Robert Grimm
New York University

Last Time: Unix

- * What is the key innovation of Unix?
 - * Unified I/O: Files, pipes, sockets all use same API
- * What other important feature offers considerable power?
 - * Inheritance of open file descriptors in forked processes combined with notion of standard in and out
- * How does protection work in Unix?
 - * Through access control lists on file system
 - * Owner, group, everyone else
 - * Set-user-ID for protection domain transfer

Today: Multics

The Three Questions

- * What is the problem?
- * What is new or different?
- * What are the contributions and limitations?

Diggin' Deeper

- * How to design for protection (security)?
 - * Design principles
- * How to specify access restrictions?
 - * Access control lists
- * How to authenticate users?
 - * Passwords
- * How to protect memory?
 - * Segments, rings of protection
- * Where did we go wrong?
 - * Size of *trusted computing base*, complexity of UI

Design Principles

- * Five principles
 - * Require explicit permission instead of exclusion
 - * Check every access (unlike capability systems)
 - * Do not rely on security by obscurity
 - * Operate with the least necessary privilege
 - * Make the UI easy to use
- * Two functional objectives
 - * Allow decentralized control
 - * Make the protection system extensible
 - * We cannot anticipate all ways of enforcing protection

Storage System & Access Control

- * A hierarchical storage system
 - * Comprising catalogs (directories) and segments (files)
 - * Later message queues and removable media descriptors
 - * Access through memory-mapped I/O
- * Each storage system object has an ACL
 - * Mapping from principals to access rights
 - * Principals partitioned into three parts
 - * User name
 - * Project
 - * Compartment
 - * ACL entries may contain wildcards for partitions

More Details on Access Control

- * Access control lists are ordered
 - * First match determines rights
 - * Need to carefully insert entries, favoring more specific ones
- * Each directory has a default ACL
 - * Copied onto new objects
- * Users and projects never go away
 - * Otherwise, would have to delete invalid ACL entries (?)
- * Object owner has full control over ACL
 - * Others cannot restrict it

Even More Details

- * Access control is hierarchical
 - * Write access to directory implies right to modify ACLs
 - * Can be used to provide centralized control
- * Detour: Access control in Windows 2000
 - * Swift et al., ACM TISSEC 5(4):398–437, Nov. 2002
 - * Supports both centralized and local control
 - * Using inherited as well as local ACL entries
 - * Favors speed of access checks over space used by ACLs
 - * Policy propagated down the tree after ACL changes
 - * Places local entries before inherited entries
 - * First matching entry applies

Some Rejected Design Points

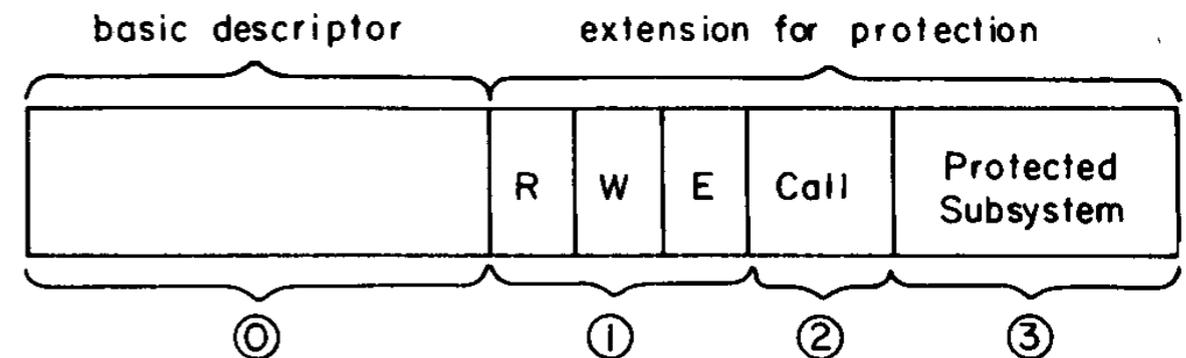
- * Placing permissions on objects (instead of in ACLs)
 - * Too inflexible as different users need different rights
 - * CTSS put permissions on objects, used (hard) links as separate entry points (capabilities)
 - * Hard to revoke access
 - * Access modes to same object depend on name
 - * Hard to determine rights
- * Using user-specified *reference monitors*
 - * Hard to isolate the owner-specified procedure
 - * Cannot run as user seeking access—why?
 - * Cannot run as supervisor—why?

User Authentication

- * Based on user names and passwords
 - * All operations require (interactive) authentication
- * Supports proxy logins (?)
- * Generates passwords automatically
 - * With English digraph statistics
- * Stores passwords in encrypted form
- * Does not print passwords
- * Automatically logs out idle users
- * Tracks (un)successful logins
- * Supports anonymous logins as well as stronger checks

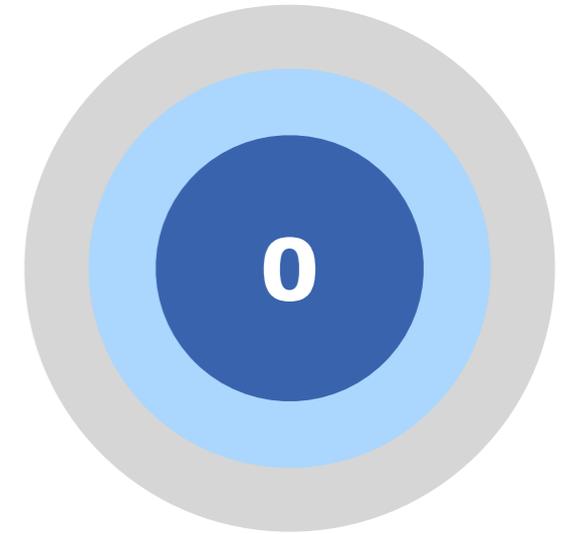
Memory Protection

- * Memory organized into segments
- * Memory accessed through segment descriptors
 - * Per-process descriptor segment lists all descriptors
 - * Pointer to physical memory
 - * List of access rights
 - * Number of *gates*
 - * ID of protected subsystem (0-7)
 - * Which modern processor also uses *rings of protection*?
 - * Is this enough?



Detour: Rings of Protection

- * 8 rings, with kernel at 0 and users at 4
- * CPU tracks current execution level
- * Descriptor contains max read/write/execute levels
 - * Though lower ring has full power over itself, outer rings
- * Call gate
 - * Lowers ring level, but only at well-defined entry points
- * Stack per ring



Back to Memory Protection

- * I/O channel programs (drivers) run as supervisor
- * Overall, memory protection relies on economy of mechanism
 - * Controls access to data and execution of code
 - * Controls everyone, including the supervisor

Two Major Weaknesses

- * Size of trusted computing base
 - * 15% of all code, 300 modules with ~200 lines (60,000 lines)
 - * Three factors
 - * Presumed (lack of) execution speed
 - * Tight deadlines
 - * Lack of understanding
- * Complexity of user interface to access control lists
 - * Permitted users and projects
 - * Corresponding permissions
 - * Right to backup, bulk I/O
 - * Protected subsystem

More Weaknesses

- * Communication lines provide insufficient feedback
- * Operator interface provides direct access to hardware
- * Users may specify their own passwords
- * No semantic checks on supervisor interface
- * Secondary storage is cleared on reuse, no deallocation
- * Administrators tend to be overprivileged
- * Backups expose other users' directory and file names

Even More Weaknesses

- * Counter-intelligence is lacking
- * Hardware protection may fail
- * "Relatively straight-forward modification can easily strengthen any of the areas"
 - * Do we believe this?

Discussion