

CIS/TCOM 551

Computer and Network Security

Spring 2005

Lecture 12

Announcements

- Final project is on the web site.
 - Updated web site with examples and test files
 - Due last day of classes: Friday, April 22nd.
- Final exam:
 - Thurs. April 28th 11am - 1pm
 - Chemistry B13

Outline

- Access Control Concepts
 - Matrix, ACL, Capabilities
- OS Mechanisms
 - Multics
 - Ring structure
 - Amoeba
 - Distributed, capabilities
 - Unix
 - File system, Setuid
 - Windows
 - File system, Tokens, EFS
 - SE Linux
 - Role-based, Domain type enforcement
- Secure OS
 - Methods for resisting stronger attacks
- Assurance
 - Multi-level security (MLS)
 - Orange Book, TCSEC
 - Common Criteria
 - Windows 2000 certification
- Some slides courtesy of John Mitchell

Access Control Matrices

[Lampson]

A[s][o]	Obj ₁	Obj ₂	...	Obj _N
Subj ₁	{r,w,x}	{r,w}	...	{}
Subj ₂	{w,x}	{}	...	
...	
Subj _M	{x}	{r,w,x}	...	{r,w,x}

Each entry contains a set of rights.

Storing the Access Control Matrix

- Subjects >> # users
 - Processes
- Objects >> # files
 - Potentially could have permissions on any resource
- The matrix is typically sparse
 - Store only non-empty entries

Access Control Lists

A[s][o]	Obj ₁	Obj ₂	...	Obj _N
Subj ₁	{r,w,x}	{r,w}	...	{}
Subj ₂	{w,x}	{}	...	{r}
...
Subj _M	{x}	{r,w,x}	...	{r,w,x}

For each object, store a list of (Subject x Rights) pairs.

Access Control Lists

- Resolving queries is linear in length of the list
- Revocation w.r.t. a single object is easy
- Delegation of rights is hard
- “Who can access this object?” is easy
 - Useful for auditing
- Lists could be long
 - Factor into groups (lists of subjects)
 - Give permissions based on group
 - Introduces consistency question w.r.t. groups
- Authentication critical
 - When does it take place? Every access would be expensive
- Access Control Lists are commonly used

Capabilities

A[s][o]	Obj ₁	Obj ₂	...	Obj _N
Subj ₁	{r,w,x}	{r,w}	...	{}
Subj ₂	{w,x}	{}	...	{r}
...
Subj _M	{x}	{r,w,x}	...	{r,w,x}

For each subject, store a list of (Object x Rights) pairs.

Capabilities

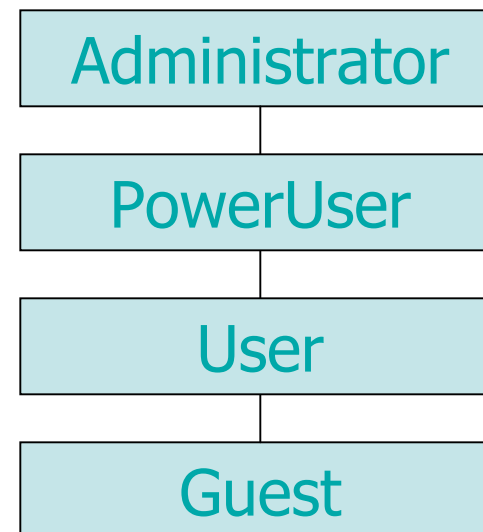
- A capability is a (Object, Rights) pair
 - Often has an expiration date
 - Used like a movie ticket e.g.:
 (“The Incredibles”, {admit one}, 7:00pm show)
- Should be unforgeable
 - Otherwise, subjects could get illegal access
- Authentication takes place when the capabilities are granted (not needed at use)
- Harder to do revocation (must find all tickets)
- Delegation is easy to do (just hand off the capability)
- Easy to audit a subject, hard to audit an object

Capabilities

- Operating system concept
 - “... of the future and always will be ...”
- Less commonly used, although capabilities show up in Kerberos
- Other Examples
 - Dennis and van Horn, MIT PDP-1 Timesharing
 - Hydra, StarOS, Intel iAPX 432, Eros, ...
 - Amoeba: distributed, unforgeable tickets
- References
 - Henry Levy, Capability-based Computer Systems
<http://www.cs.washington.edu/homes/levy/capabook/>
 - Tanenbaum, Amoeba papers

Roles (also called Groups)

- Role = set of users
 - Administrator, PowerUser, User, Guest
 - Assign permissions to roles; each user gets permission
- Role hierarchy
 - Partial order of roles
 - Each role gets permissions of roles below
 - List only new permissions given to each role



Groups for resources, rights

- Permission = $\langle \text{right}, \text{resource} \rangle$
- Permission hierarchies
 - If user has right r , and $r > s$, then user has right s
 - If user has read access to directory, user has read access to every file in directory
- Big problem in access control
 - Complex mechanisms require complex input
 - Difficult to configure and maintain
 - Roles, other organizing ideas try to simplify problem

Example OS Mechanisms

- Multics
- Amoeba
- Unix
- Windows
- SE Linux (briefly)

Multics

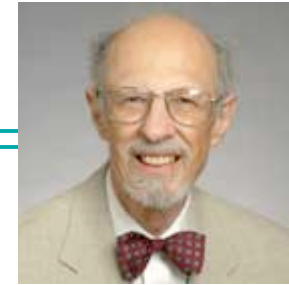
- Operating System
 - Designed 1964-1967
 - MIT Project MAC, Bell Labs, GE
 - At peak, ~100 Multics sites
 - Last system, Canadian Department of Defense, Nova Scotia shut down October, 2000
- Extensive Security Mechanisms
 - Influenced many subsequent systems



<http://www.multicians.org/security.html>

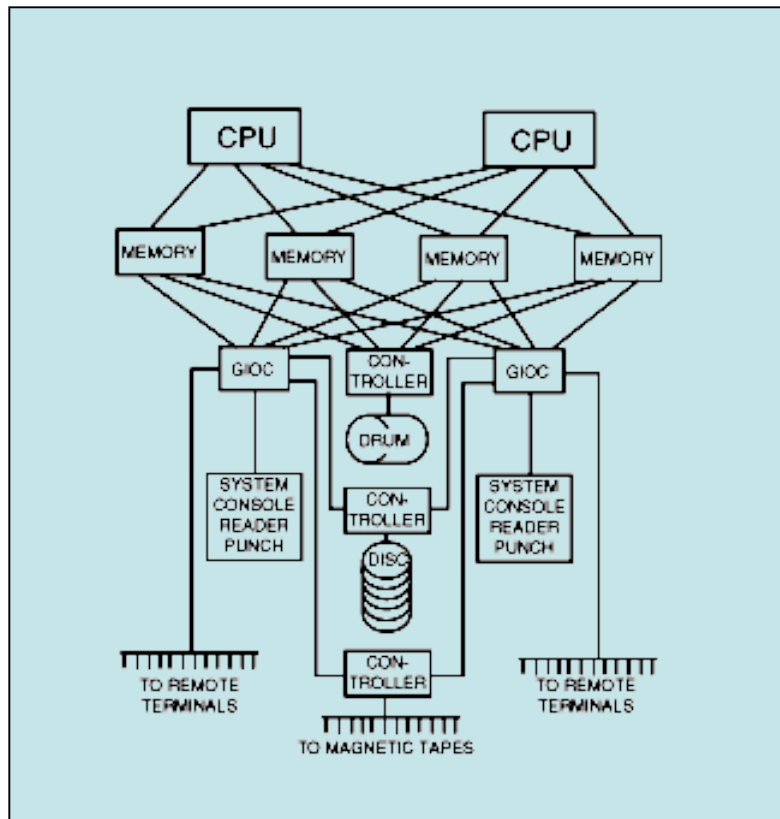
E.I. Organick, *The Multics System: An Examination of Its Structure*, MIT Press, 1972

Multics time period



F.J. Corbato

- Timesharing was new concept
 - Serve Boston area with one 386-based PC



Multics Innovations

- Segmented, Virtual memory
 - Hardware translates virtual address to real address
- High-level language implementation
 - Written in PL/1, only small part in assembly lang
- Shared memory multiprocessor
 - Multiple CPUs share same physical memory
- Relational database
 - Multics Relational Data Store (MRDS) in 1978
- Security
 - Designed to be secure from the beginning
 - First B2 security rating (1980s), only one for years
 - More about government certification levels later

Multics Access Model

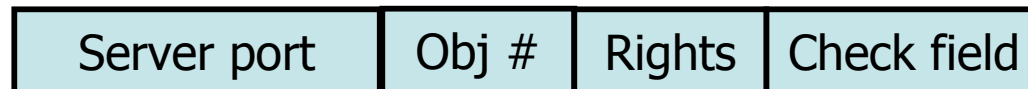
- Ring structure
 - A ring is a domain in which a process executes
 - Numbered 0, 1, 2, ... ; Kernel is ring 0
 - Graduated privileges
 - Processes at ring i have privileges of every ring $j > i$
- Segments
 - Each data area or procedure is called a segment
 - Segment protection $\langle b1, b2, b3 \rangle$ with $b1 \leq b2 \leq b3$
 - Process/data can be accessed from rings $b1 \dots b2$
 - A process from rings $b2 \dots b3$ can only call segment at restricted entry points

Multics processes

- Multiple segments
 - Segments are dynamically linked
 - Linking process uses file system to find segment
 - A segment may be shared by several processes
- Multiple rings
 - Procedure, data segments each in specific ring
 - Access depends on two mechanisms
 - Per-Segment Access Control
 - File author specifies the users that have access to it
 - Concentric Rings of Protection
 - Call or read/write segments in outer rings
 - To access inner ring, go through a “gatekeeper”
- Interprocess communication through “channels”

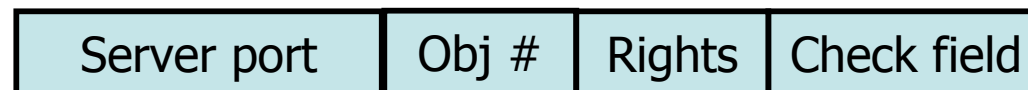
Amoeba

- Distributed system
 - Multiple processors, connected by network
 - Process on A can start a new process on B
 - Location of processes designed to be transparent
- Capability-based system
 - Each object resides on server
 - Invoke operation through message to server
 - Send message with capability and parameters
 - Server uses object # to identify object
 - Server checks rights field to see if operation is allowed
 - Check field prevents processes from forging capabilities



Capabilities

- Owner capability
 - When server creates object, returns owner cap.
 - All rights bits are set to 1 (= allow operation)
 - Check field contains 48-bit rand number stored by server
- Derived capability
 - Owner can set some rights bits to 0
 - Calculate new check field
 - XOR rights field with random number from check field
 - Apply one-way hash function to calculate new check field
 - Server can verify rights and check field
 - Without owner capability, cannot forge derived capability



Protection by user-process at server; no special OS support needed