CIS 551 / TCOM 401

Computer and Network Security

Spring 2006

Lecture 3

Announcements

- Prof. Zdancewic's office hours:
  9:30 - 10:30 am on Thursdays (& by appointment)
  Levine 511

- Karl Mazurak's office hours:
  noon-1:00pm Monday & Wednesday
  Levine 575

- Stack randomization on eniac-l has been turned off (for the duration of this project)

The “Gold” Standard

- *Au*thentication
  - Identify which principals take which actions
- *Au*dit
  - Recording the security relevant actions
- *Au*thorization
  - Determine what actions are permissible

  - This lecture is about authorization.
  - We'll get to authentication & audit later.

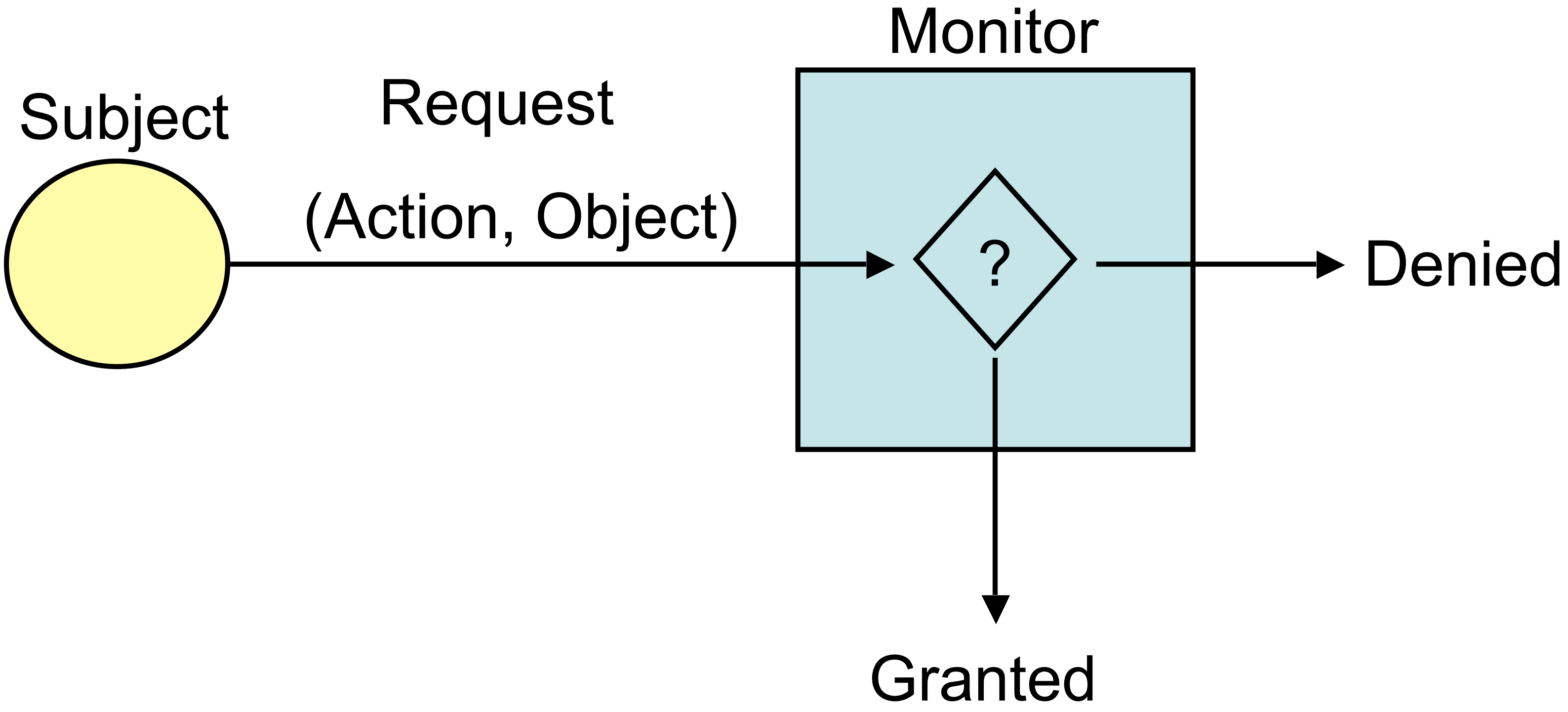
Authorization

- Authorization is the process of determining whether a principal is permitted to perform a particular action.
- Access control
  - Example: Read/Write/Execute permissions for a file system.
  - Example: Java applets have restricted authorization to perform network & disk I/O.

Policy vs. Mechanism

- Access control policy is a *specification*
  - Given in terms of a model of the system
  - Subjects: do things (i.e. a process writes to files)
  - Objects: are passive (i.e. the file itself)
  - Actions: what the subjects do (i.e. read a string from a file)
  - Rights: describe authority (i.e. read or write permission)
- Mechanisms are used to *implement* a policy
  - Example: access control bits in Unix file system & OS checks
  - Mechanism should be general; ideally should not constrain the possible policies.
  - Complete mediation: every access must be checked

Reference Monitors



Example Reference Monitors

- Operating Systems
  - File system
  - Memory (virtual memory, separate address spaces)
- Firewalls
  - Regulate network access
- Java Virtual Machine
  - Regulates Java programs’ resource usage

- Operate at different levels of abstraction
  - Interface (Subjects, Objects, Actions) varies

Access Control Matrices

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.

Access Control Checks

- Suppose subject s wants to perform action that requires right r on object o:

- If ($r \in A[s][o]$) then *perform action*
  else *access is denied*

Rights and Actions

- Besides read, write, execute actions there are many others:
- Ownership
- Creation
  - New subjects (i.e. in Unix add a user)
  - New objects (i.e. create a new file)
  - New rights: Grant right r to subject s with respect to object o (sometimes called delegation)
- Deletion of
  - Subjects
  - Objects
  - Rights (sometimes called revocation)

Example

- Assume OS is a subject with all rights
- To create a file f owned by Alice:
  - Create object f
  - Grant own to Alice with respect to f
  - Grant read to Alice with respect to f
  - Grant write to Alice with respect to f

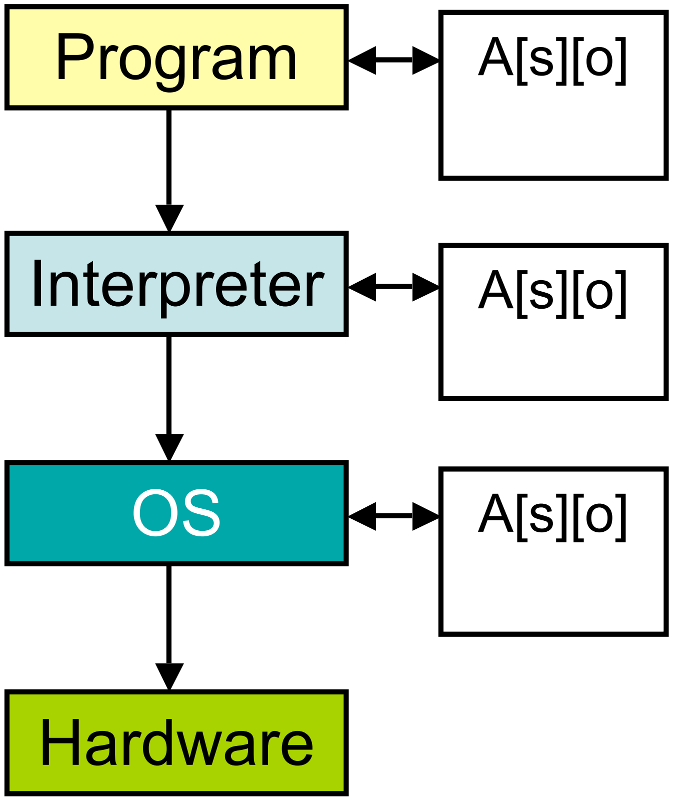
Reference Monitors

- Criteria
  - Correctness
  - Complete mediation (all avenues of access must be protected)
  - Expressiveness (what policies are admitted)
  - How large/complex is the mechanism?

- Trusted Computing Base (TCB)
  - The set of components that must be trusted to enforce a given security policy
  - Would like to simplify/minimize the TCB to improve assurance of correctness

Software Mechanisms

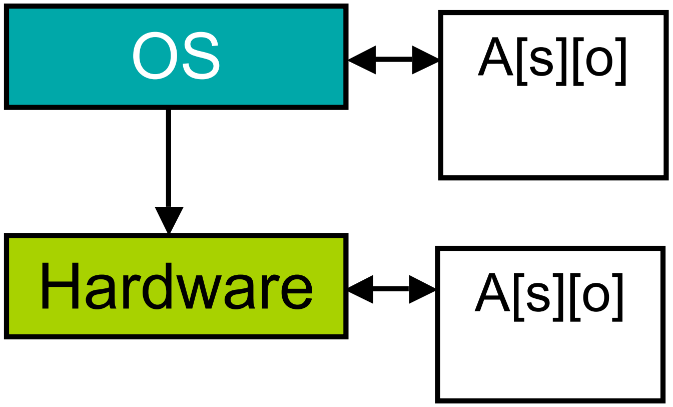
- Interpreters
  - Check the execution of every instruction
  - Hard to mediate high-level abstractions
- Wrappers
  - Only “interpret” some of the instructions
  - What do you wrap?
  - Where do you wrap? (link-time?)
- Operating Systems
  - Level of granularity?
  - Context switching overheads?



Hardware Mechanisms

- Multiple modes of operation
  - User mode (problem state)
  - Kernel mode (supervisor state)

- Specialized hardware
  - Virtual memory support (TLB's, etc.)
  - Interrupts



Protecting Reference Monitors

- It must not be possible to circumvent the reference monitor by corrupting it
- Mechanisms
  - Type checking
  - Sandboxing: run processes in isolation
  - Software fault isolation: rewrite memory access instructions to perform bounds checking
  - User/Kernel modes
  - Segmentation of memory (OS resources aren’t part of virtual memory system)

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
- “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.

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

Capabilities

- A capability is a (Object, Rights) pair
  - Used like a movie ticket e.g.: (“Matrix Revolutions”, {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)
- Easy to audit a subject, hard to audit an object

Implementing Capabilities

- Must be able to name objects
- Unique identifiers
  - Must keep map of UIDs to objects
  - Must protect integrity of the map
  - Extra level of indirection to use the object
  - Generating UIDs can be difficult
- Pointers
  - Name changes when the object moves
  - Remote pointers in distributed setting
  - Aliasing possible

Unforgeability of Capabilities

- Special hardware: tagged words in memory
  - Can't copy/modify tagged words
- Store the capabilities in protected address space
- Could use static scoping mechanism of safe programming languages.
  - Java's "private" fields
- Could use cryptographic techniques
  - OS kernel could sign (Object, Rights) pairs using a private key
  - Any process can verify the capability