Lecture 4 - Authorization

CMPSC 443 - Spring 2012
Introduction Computer and Network Security
Professor Jaeger
www.cse.psu.edu/~tjaeger/cse443-s12/
Why authenticate?

• Why do we want to verify the identity of a user?
A Brief History

• Early computing systems had no isolation
  – Shared memory space
  – Shared file space

• Some physical limitations made this OK
  – Batch processing
  – Load the tape/disk for the application
  – Network? What network?

• In the mid-60s people started to work on ‘multiuser’ or ‘time-sharing’ systems
  – What about a bug?
  – What about my data?
Multiprogrammed Systems

• Multics project
  – AT&T, MIT, Honeywell, etc.
  – General purpose, multi-user system
  – Comprehensive security
    • Hardware protection
    • Subject labeling
    • Permission management

• UNIX project
  – Spin-off of Multics project
    • When AT&T left
  – A stripped-down multiuser system
Control Access

• An identity permits access to resources
• In computer security this is called
  – Access control
  – Authorization
• In authorization, we talk about:
  – Subjects (for whom an action is performed)
  – Objects (upon what an action is performed)
  – Operations (the type of action performed)
• Authorization limits a subject’s access perform an operation on an object
  – The combination of object and operations allowed are called a permission
Access Matrix

- Describe all possible accesses
  - Operations of $(S_2,O_2)$
  - E.g., read, write, execute
- Specify which users’ processes can access which files
- Necessary to specify policy to protect users

<table>
<thead>
<tr>
<th></th>
<th>O₁</th>
<th>O₂</th>
<th>O₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₂</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₃</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Access Control Lists

• **System stores**
  – Which operations can subjects perform
  – For each object

• *Advantage*: Makes you think about how to protect each object
  – Also, easier to confine subjects as we’ll discuss later

• *Disadvantage*: Cannot tell what permissions a particular subject has without looking at each object
  – Process always uses all of its permissions, as we’ll discuss later

<table>
<thead>
<tr>
<th></th>
<th>O₁</th>
<th>O₂</th>
<th>O₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₂</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₃</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Capabilities

- System stores
  - Which operations can be performed on each object
  - For each subject
- Advantages and disadvantages are reverse of ACL case, naturally

<table>
<thead>
<tr>
<th></th>
<th>O₁</th>
<th>O₂</th>
<th>O₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₂</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₃</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Authentication and Access

• Authenticate user
  – E.g., login and ssh
  – Verify password or ...

• Create processes with appropriate identity (subject)
  – E.g., UNIX user id

• Limit access of these processes using subject
  – E.g., Access control of files based on subject

• Protect one user from another
Sharing in the Access Matrix

- How do you give someone access to your file?
- Access matrix also has management permissions
  - owner permission
- A subject with owner permission can
  - Give another user permissions to an object
  - Even the owner permission itself
- This seems necessary, right?

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>O2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>O3</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
Authorization Challenges

- Sounds pretty easy, but there are several challenges
  - What’s an object?
  - What’s an operation?
  - What’s a subject?
  - Who’s going to manage permissions?
Operating Systems and Authorization

• Traditionally, all true authorization was performed operating systems
  – But, that is no longer the case

• Operating systems are not fully trusted
  – Commercial operating systems are immense
    – Thus, system trust is being focused lower layers (VMM, microkernel, ...)

• Security-critical decisions are often made by user-space programs
  – We depend on several now (X, Apache, DBs, DBus, ...)

• Applications may span multiple hosts, so Internet services do authorization
Objects

• What’s an object?
  – OS: Many things are files
  – Although not all

• Different software components have their own objects
  – Virtualization
  – Microkernels
  – X Windows
  – Database
  – Apache
  – Logrotate
  – Clouds
  – Social Networks
Operations

• What’s an operation?
  – OS: System call
  – Well, not really because many things can happen in a single system call
    • What happens on a file open?

• Security-sensitive operations
  – Any operation that may impact the security of your system
    • Confidentiality, Integrity, Availability
  – A little bit imprecise, but enables some interaction between subjects

• Lots of security-sensitive operations
  – Communication between VMs
  – Cut-and-paste between windows
  – Update a database record
  – Post a message to a social network
Subjects

• What’s a subject?
  – OS: **System** (root/administrator) and **Regular Users** (you and me)
  – However, even for operating systems this distinction is unsatisfactory
    • System is too coarse
    • User is too coarse/fine

• Why is system too coarse?
  – Might that be the same problem for users?

• Do users even matter to operating systems anymore?
  – How many users on your devices?
Who Are You?

- Identity vs. Permission
Root/Administrative User

- Subjects with full system access
  - Initialize the system
  - Modify the kernel
  - Install software

- Need extra permissions to perform administrative tasks
  - Ends up being a lot of processes

- All are part of the trusted computing base
Regular Users

• An unprivileged user
  – However, all your processes run with the same permissions

• What are all the programs that you run?
  – Should they all have full access to any file you can access?

• Sandboxing
  – Run a program with a subset of your permissions
Role-Based Access Control

• Associate permissions with job functions
  – Each job defines a set of tasks
  – The tasks need permissions
  – The permissions define a role

• Bank Teller
  – Read/Write to client accounts
  – Cannot create new accounts
  – Cannot create a loan
  – Role defines only the permissions allowed for the job

• What kind of jobs can we define permission sets for?
Role-based Access Control

• Model consists of two relationships
  – Role-permission assignments
  – User-role assignments

• Assign permissions to roles
  – These are largely fixed

• Assign a user to the roles they can assume
  – These change with each user
  – Administrators must manage this relationship
Managing Access Control

• Who’s going to manage?
  – Formerly, you (and your programs)
  – But, then it was easy
    • Subjects: All, Group, Just You
    • Operations: Read, Write, Execute
  – But, this approach does not provide security

• Now, we have gone overboard
  – Models with multiple types of subjects, objects, operations are common
  – Policies with 10,000+ rules

• Too complex for users -- even system admins
  – OS Distributors can write fixed permissions
  – But what if we need to change permissions?
    • Make the programmer manage it?
Take Away

• We have just looked at the most common mechanisms
  – Password Authentication
  – User-based Authorization

• There are a slew of problems with each

• But, this is what the world uses
  – What can we do?

That Is the Topic of This Course