Capabilities and Sandboxes

CMPSC 443 - Spring 2012
Introduction Computer and Network Security
Professor Jaeger
www.cse.psu.edu/~tjaeger/cse443-s12/
Too Many Permissions

• After configuring access control (whew!) some processes are still scaring you

• Case 1:
  – You downloaded a program from the Internet
  – Looks like fun, but...
  – Does it really need access to your Quicken files?

• Case 2:
  – You are running a server for lots of different clients
  – Do you really need to run with permissions for all your clients enabled all the time?

• What can you do?
Access Control Lists

- Windows and UNIX describe permissions as **Access Control Lists**
  - Permissions are associated with objects

<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>
MAC Labels - Levels and Categories

- All subjects are assigned clearance levels and compartments
  - Alice: (SECRET, {CRYPTO, NUC})
  - Bob: (CONFIDENTIAL, {INTEL})
  - Charlie: (TOP SECRET, {CRYPTO, NUC, INTEL})

- All objects are assigned an access class
  - DocA: (CONFIDENTIAL, {INTEL})
  - DocB: (SECRET, {CRYPTO})
  - DocC: (UNCLASSIFIED, {NUC})

- Do you share permissions with other users?
- Not all (DAC), but you still have lots of sharing
Scenario

• You have downloaded a program from the Internet
  – Should you run it?
  – Can you run it in an environment that protects your data?

• Not with MLS or ACLs
  – MLS will prevent secret leakage, but will not provide integrity protection

• What do you do?
Sandbox

- Specialized enforcement mechanism
- Limit permissions available to process
  - Subset of user permissions
Chroot

- Create a *domain* in which a process is *confined*
  - Process can only read/write within file system subtree
  - Applies to all descendant processes
  - Can carry file descriptors in ‘chroot jail’
Chroot Vulnerability

• Unfortunately, chroot can trick its own system
  – define a passwd file at <newroot>/etc/passwd
  – run su
    • su thinks that this is the real passwd file
  – gives root access
    • Use mknod to create device file to access physical memory

• Setup requires great care
  – Never run chroot process as root
  – Must not be able to get root privileges
  – No control by chrooted process (user) of contents in jail
  – Be careful about descriptors, open sockets, IPC that may be available
User-Space Sandboxes

• Enforce richer policy outside operating system
• Mediate system calls of process
  – Libraries protected from program (Java, Tcl)
  – Reflect system calls from OS back to user space
  – Both (reflect back to library)
• System call execution in sandbox
  – Program calls `open`
  – Call is sent to sandbox enforcer
  – Enforcer authorizes the invocation
  – If approved, system call is executed
  – Result is returned to the program
    • maybe indirectly if sandbox wants to control use
• Do you see a problem with this approach?
Sandbox Problem

- System call execution in sandbox
  - Program calls `open`
  - Call is sent to sandbox enforcer
  - Enforcer authorizes the invocation
    - Bad guy changes the system
  - If approved, system call is executed
  - Result is returned to the program
    - maybe indirectly if sandbox wants to control use

- Difficult to ensure that the system is the same when the call is executed as when it is checked
  - Time-of-check-to-time-of-use

- Solutions are non-trivial
Another Scenario

• A UNIX-like system has a compiler (SYSX)
• Normal users can run the compiler on their code
• SYSX writes two files: the compiler output and billing information
• Users can also provide the name of a file to receive debugging information
• The compiler runs with RW access to these files
• What happens when a user specifies the billing information file as the file to receive debugging info?
Confused Deputy

• The server (SYSX) runs with the rights of its clients
  – *All of its clients*

• And a client can trick the server into using another client’s permission for it
  – Or the server’s permissions, as in the billing info

• How can we prevent this problem?
Capabilities

- Capabilities are permission objects given to subjects (processes)
  - The access rights of a process are determined by the set of capabilities it has access to

<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>
How Do You Use Capabilities

• The client has a set of capabilities
• The server has a set of capabilities
• The client provides capabilities with its requests
• The server provides a set of capabilities to OS for each system call (to open a new file)
  – Server gets to choose which capabilities to use
• The server combines some of its capabilities with those of the client to service the operations that the client’s request generates
  – Write to compiler output
  – Write to billing file

• Does this solve the problem?
How to Solve the Confused Deputy Problem

• The client has a set of capabilities
• The server has a set of capabilities
• The client provides capabilities with its requests
• The server only uses the capabilities provided by the client when servicing the client’s request

• How does this solve the problem?
How to Solve the Confused Deputy Problem

• The client has a set of capabilities
• The server has a set of capabilities
• The client provides capabilities with its requests
• The server only uses the capabilities provided by the client when servicing the client’s request

• How does this solve the problem?
  – Use server capabilities only to write billing files
  – Use client capabilities only to write compiler output
Can’t We Do That with ACLs?

- Key step:
  - The server only uses the capabilities provided by the client when servicing the client’s request
- Can the server change ACLs?
  - Some of them (ones it owns)
  - Extra system calls
  - What if another thread needs those permissions?
- Not practical to solve with ACLs
Capability Systems

• Goes back to the mid-1960s
  – (Dennis and van Horn, Plessey system, CTSS)
• Idea: include accessibility with reference

• What is a normal reference?
  – Think in a program
• What defines accessibility?
  – Think in an access matrix
Capability Systems

• Goes back to the mid-1960s (Dennis and van Horn, Plessey system, CTSS)

• Idea: include accessibility with reference

• What is a normal reference?
  – Addresses or IDs

• What defines accessibility?
  – Operations

• A capability is a reference (object ID) with the operations the holder is allowed to perform on the associated object
  – Also may have an integrity value for checking validity
Capabilities

• Analogy

• Like a *house key*
  – Possession grants access
  – Need to use the right key for the right job
  – Can make copies and give those to others
  – Changing the lock invalidates all keys
  – Losing the key loses access
  – Can’t easily keep track of where the copied keys go
Capabilities

• Example

• File Descriptor
  – Possession grants access
  – Need to use the right key for the right job
  – Can make copies and give those to others
  – Changing the lock invalidates all keys
  – Losing the key loses access
  – Can’t easily keep track of where the copied keys go
Capabilities

• Capabilities must be **unforgeable**
  – Why?

• Representations of Capabilities
  – **Hardware capabilities**
    • Hardware associates permissions with reference in separate part of memory
  – **System-controlled capabilities**
    • System stores mapping of permissions to reference
  – **Cryptographic capabilities**
    • User processes hold and distribute capability objects
Current Capability Systems

• Name one?

• In the 70s/80s, people assumed that this is the way that we would go

• Still several research systems, but not widely used

• What happened?
• There is a fundamental problem between capability systems and MLS
  – Idea: I give you a capability to write to my files
    • But my files are low secrecy
• Problem: User A is high secrecy, User B is low secrecy
  – A can read B’s data
  – B’s data includes capabilities
    • To write to B’s low secrecy files
  – Since A can read the capability she can use it
    • At least Trojan horse’s running on her behalf can
• How do you fix this?
Security People’s Answer

• There is a fundamental problem between capability systems and MLS
  - Idea: I give you a capability to write to my files
    • But my files are low secrecy

• Problem: User A is high secrecy, User B is low secrecy
  - A can read B’s data
  - B’s data includes capabilities
    • To write to B’s low secrecy files
  - Since A can read the capability she can use it
    • At least Trojan horse’s running on her behalf can

• Check every capability before it is used/propagated against ACL
Real People’s Answer

- **Convenience**
  - Where do you get capabilities from?
  - Namespaces are nice
  - Programmers have to track capabilities
    - All programmers, not just the ones with confused deputy problems
  - Revocation is not easy

- Usually, capabilities are used in a hybrid way
  - Such as file descriptors
  - Make programmers’ lives easier by caching references and their associated access rights
Take Away

• Problem using MLS or ACL systems
  – Too many permissions are available to processes with no straightforward way to restrict them

• Sandboxing
  – Run program in an environment that can limit the available permissions of the process to a subset of normal rights
  – Idea is easy, but implementation has many edge conditions

• Capability Systems
  – Distribute access control tokens (capabilities) to processes
  – While capabilities can be useful, requiring all operations to be governed by capabilities is painful for programmers