Security Goals, Threats, and Principles
CS 239
Security for System Software
Peter Reiher
April 3, 2002

Basic Security Principles

- Terms and concepts
- Mechanisms

Security and Protection

- *Security* is a policy
  – E.g., "no unauthorized user may access this file"
- *Protection* is a mechanism
  – E.g., "the system checks user identity against access permissions"
- Protection mechanisms implement security policies

Design Principles for Secure Systems

- Economy
- Complete mediation
- Open design
- Separation of privileges
- Least privilege
- Least common mechanism
- Acceptability
- Fail-safe defaults

Economy in Security Design

- Economical to develop
  – And to use
  – And to verify
- Should add little or no overhead
- Should do only what needs to be done
- Generally, try to keep it simple and small

Complete Mediation

- Apply security on every access to a protected object
  – E.g., each read of a file, not just the open
- Also involves checking access on everything that could be attacked

Open Design

- Don't rely on "security through obscurity"
- Assume all potential attackers know everything about the design
  - And completely understand it
- This doesn't mean publish everything important about your security system
  - Though sometimes that's a good idea

Separation of Privileges

- Provide mechanisms that separate the privileges used for one purpose from those used for another
- To allow flexibility in security systems
- E.g., separate access control on each file

Least Privilege

- Give bare minimum access rights required to complete a task
- Require another request to perform another type of access
- E.g., don't give write permission to a file if the program only asked for read

Least Common Mechanism

- Avoid sharing parts of the security mechanism
  - among different users
  - among different parts of the system
- Coupling leads to possibilities security breaches

Acceptability

- Mechanism must be simple to use
- Simple enough that people will use it without thinking about it
- Must rarely or never prevent permissible accesses

Fail-Safe Designs

- Default to lack of access
- So if something goes wrong or is forgotten or isn't done, no security lost
- If important mistakes are made, you'll find out about them
  - Without loss of security
  - But if it happens too often . . .

Tools for Security

- Physical security
- Access control
- Encryption
- Authentication
- Encapsulation
- Intrusion detection
- Common sense

Physical Security

- Lock up your computer
  - Actually, sometimes a good answer
- But what about networking?
  - Networks poke a hole in the locked door
- In any case, lack of physical security often makes other measures pointless

Access Controls

- Only let authorized parties access the system
- A lot trickier than it sounds
- Particularly in a network environment
- Once data is outside your system, how can you continue to control it?
  - Again, of concern in network environments

Encryption

- Algorithms to hide the content of data or communications
- Only those knowing a secret can decrypt the protection
- One of the most important tools in computer security

Encryption is Not a Panacea

- Encryption is usually breakable
  - Given enough time and resources
- Encryption can't protect everything
- Encryption is only as good as the security measures that use it

Authentication

- Methods of ensuring that someone is who they say they are
- Vital for access control
- But also vital for many other purposes
- Often (but not always) based on encryption

Encapsulation

- Methods of allowing outsiders limited access to your resources
- Let them use or access some things
  - But not everything
- Simple, in concept
- Extremely challenging, in practice

Intrusion Detection

- All security methods sometimes fail
- When they do, notice that something is wrong
- And take steps to correct the problem
- Reactive, not preventative
  - But unrealistic to believe any prevention is certain
- Must be automatic to be really useful

Common Sense

- A lot of problems arise because people don't like to think
- The best security tools generally fail if people use them badly
- If the easiest way in is to fool people, that's what attackers will do

The Depressing Truth

- Ultimately, computer security is a losing battle
- Nothing will ever work 100%
- Nothing will work forever
- All your efforts will eventually be undone
- It's like housework – doing it doesn't make the house clean tomorrow, but not doing guarantees the house is dirty today

Access Control

- Security could be easy
  - If we didn't want anyone to get access to anything
- The trick is giving access to only the right people
- How do we ensure that a given resource can only be accessed by the proper people?

Goals for Access Control

- Complete mediation
- Least privilege
- Useful in a networked environment
- Scalability
- Cost and usability

Access Control Mechanisms

- Directories
- Access control lists
- Capabilities
- Access control matrices

Directories

- Each user has a list of the items he can access
  - With the associated rights
- When a user wants to access an item, look it up in his directory

Problems With the Directory Approach

- Per-user directories get very large
  - Overhead and performance problems
- Universal revocation of access
- Pseudonym problems
- Works poorly in networks
- This method is not widely used

Access Control Lists

- For each protectable resource, maintain a single list
- Each list entry specifies a user who can access the resource
  - And the allowable modes of access
- When a user requests access to a resource, check the access control list (ACL)

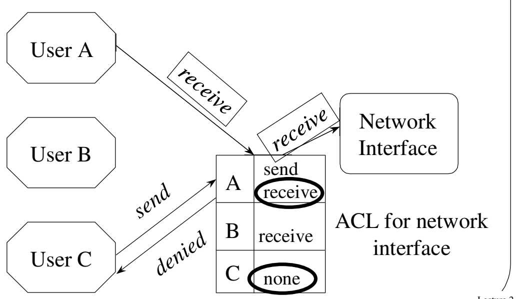
ACL Objects and Subjects

- In ACL terminology, the resources being protected are *objects*
- The entities attempting to access them are *subjects*
  - Allowing finer granularity of control than per-user

ACL Example

- An operating system example:
  - Using ACLs to protect a network interface device
- User A is allowed to receive from and send to the device
- User B may only receive from it
- User C may not access it

An ACL Protecting a Device



Issues for Access Control Lists

- How do you know the requestor is who he says he is?
- How do you protect the access control list from modification?
- How do you determine what resources a user can access?

ACLs in Practice

- Unix file permissions are a limited form of an ACL
  – Only owner, group, and all can have ACL entries
  – Only read/write/execute controls are available
- Other systems (like NT) have more general ACL mechanisms

Pros and Cons of ACLs

\+ Easy to figure out who can access a resource

\+ Easy to revoke or change access permissions

– Hard to figure out what a subject can access

– Changing access rights requires getting to the object

Capabilities

- Each subject keeps a set of data items that specify his allowable accesses
- Essentially, a set of tickets
- Possession of the capability for an object implies that access is allowed

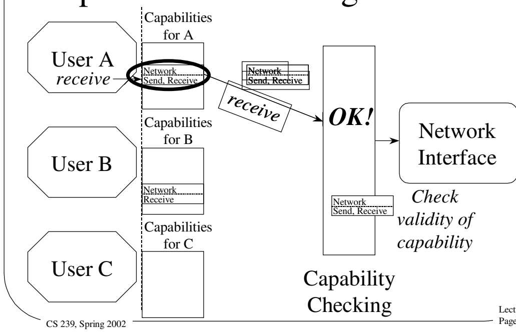
Properties of Capabilities

- Must be unforgeable
  – In single machine, keep under control of OS
  – What about in a networked system?
- In most systems, some capabilities allow creation of other capabilities
  – Process can pass restricted set of capabilities to a subprocess

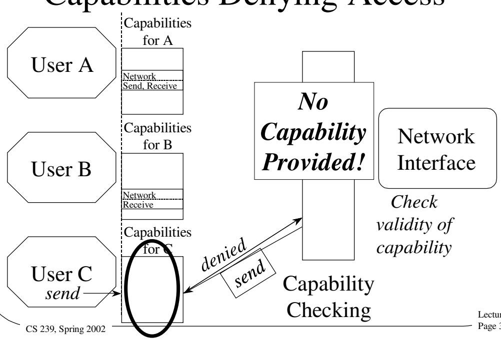
Capabilities and Domains

- The set of objects a subject can access at a given moment is its domain
  - The subject has a capability for each object in its domain
- Domains can be expanded by obtaining new capabilities
- New domains can be created for subprocesses
- Where do we keep capabilities?

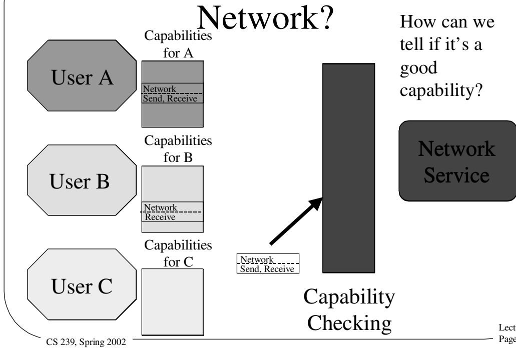
Capabilities Protecting a Device



Capabilities Denying Access



How Will This Work in a Network?



Revoking Capabilities

- A major challenge in capability systems
- Several methods available:

1). Search and destroy
2). Invalidation at use
3). Indirection through a token
4). Generation numbers

Revocation By Destroying Capabilities

- Find the capability you want to revoke
- Destroy it
- Easy if all capabilities live only in system-controlled memory
- But most systems allow storage of capabilities on disk
- And what about networked systems?

Revocation By Invalidation on Use

- Keep a list of revoked capabilities
  - Usually one list per object
- When a capability is presented for use, check it against the list
- Expensive, especially if the list is long or complete mediation is used
  - Not feasible on every access
- And what about networked systems?

Revocation by Indirection Through a Token

- Capability points to token under system control
- Token is set up on first access to object
- To revoke access, destroy the token
- Adds cost to checking access
- Usually hard to provide selective revocation

Revocation By Generation Numbers

- Store a random number in each capability
- Store the same random number with the protected object
- On access, check the numbers
- To revoke access, change the number
- No selective revocation
- Requires some control of capabilities

Pros and Cons of Capabilities

+ Easy to determine what a subject can access
+ Potentially faster than ACLs (in some circumstances)
+ Easy model for transfer of privileges
– Hard to determine who can access an object
– Requires extra mechanism to allow revocation
– In network environment, need cryptographic methods to prevent forgery

ACLs, Capabilities, Complete Mediation, & Performance

- Ideally, every data access should have access control independently applied
- Practicality of doing so depends on the performance costs
- What does it cost to use ACLs?
  - Capabilities?

Performance Issues of Access Control

- What if the status of the access control mechanism changed between when last checked and current access?
- Common case is nothing changes
- Different approaches possible
  - Actually check changeable data structure on each access
  - Give process something cheap and revocable that allows access

Access Control and ACLs

- The ACL is a list
- Initially, checking an ACL involves searching a list
- For later checks, maintain pointer to list entry
- Be sure that changing the permissions changes what's pointed to

Access Controls and Capabilities

- Attach the capability (or pointer to it) to each request
- Use attached information to determine if current access is permissible
- This approach is hard to use with revocation

An Alternate Approach To Using Capabilities

- On first access, use a capability to obtain an access token
  - Using careful, expensive checks to see if capability was revoked
- If revocation required, destroy the access token
- Can also be done with pointers

Access Control in the Distributed World

- ACLs still work OK
  - Provided you have a global namespace for subjects
- Capabilities are more problematic
  - Their security relies on unforgeability

Using Cryptographic Capabilities

- Can cryptography make capabilities unforgeable?
- It can make it impossible to create them from nothing
  - And only usable by their owner
- But it can't make them uncopyable
- So cryptographic capability systems must assume they can be freely copied

Access Control Matrices

- A very general access control concept
- In principal, ACLs are a 1-D list of who is permitted to access one object
- And capabilities are a 1-D list of what one subject can access
- Access control matrices are a 2-D description of access rights

Access Control Matrix Example

	File A	File B	Network	Printer
User 1	rw	r		w
User 2	r		sr	w
Sysadmin	rw	rw	sr	rw configure
Guest			sr	

Objects

User 2's Capabilities

Subjects

File B's ACL

Pros and Cons of Access Control Matrices

+ Makes all access issues explicit and easy to find
+ Easy to tell who can access a resource, and what resources anyone can access
– Matrix very sparse, so inefficient
– Hard to achieve good performance
• More important conceptually than in implementations

Why Is This Important?

- Much of security relates to allowing some people access to some resources
- While preventing the same access to others
- Without some method of determining who should access what . . .

  You can't do that