Operating System Security
CS 111
Operating Systems
Peter Reiher

Outline

- Basic concepts in computer security
- Design principles for security
- Important security tools for operating systems
- Access control
- Cryptography and operating systems
- Authentication and operating systems
- Protecting operating system resources

Security: Basic Concepts

- What do we mean by security?
- What is trust?
- Why is security a problem?
  - In particular, a problem with a different nature than, say, performance
  - Or even reliability

What Is Security?

- *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
- We need to understand our goals to properly set our policies
  - And threats to achieving our goals
  - These factors drive which mechanisms we must use

Security Goals

- **C**onfidentiality
  - If it's supposed to be secret, be careful who hears it
- **I**ntegrity
  - Don't let someone change something they shouldn't
- **A**vailability
  - Don't let someone stop others from using services
- Note that we didn't mention "computers" here
  - This classification of security goals is very general

Trust

- An extremely important security concept
- You do certain things for those you trust
- You don’t do them for those you don’t
- Seems simple, but . . .

What Do We Trust?

- Other users?
- Other computers?
- Our own computer?
- Programs?
- Pieces of data?
- Network messages?
- In each case, how can we determine trust?

Problems With Trust

- How do you express trust?
- Why do you trust something?
- How can you be sure who you're dealing with?
  - Since identity and trust usually linked
- What if trust is situational?
- What if trust changes?

Why Is Security Different?

- OK, so we care about security
- Isn't this just another design dimension
  – Like performance, usability, reliability, cost, etc.
- Yes and no
- Yes, it's a separable dimension of design
- No, it's not just like the others

What Makes Security Unique?

- Security is different than most other problems in CS
- The “universe” we’re working in is much more hostile
- Human opponents seek to outwit us
- Fundamentally, we want to share secrets in a controlled way
  - A classically hard problem in human relations

What Makes Security Hard?

- You have to get everything right
  - Any mistake is an opportunity for your opponent
- When was the last time you saw a computer system that did everything right?
- Since the OS underlies everything, security errors there compromise everything

Security Is Actually Even Harder

- The computer itself isn't the only point of vulnerability
- If the computer security is good enough, the foe will attack:
  - The users
  - The programmers
  - The system administrators
  - Or something you never thought of

A Further Problem With Security

- Security costs
  - Computing resources
  - People’s time and attention
- Security must work 100% effectively
- With 0% overhead
- Critically important that fundamental, common OS operations aren’t slowed by security

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
- As OS grows, this gets harder

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
- Hardware can help here
  – E.g., memory accesses have complete mediation via paging hardware

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
- Obscurity can provide *some* security, but it's brittle
  - When the fog is cleared, the security disappears
- Windows (closed design) is not more secure than Linux (open design)

Separation of Privilege

- 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 possible security breaches
- E.g., in memory management, having separate page tables for different processes
  - Makes it hard for one process to touch memory of another

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
- Windows 7 mechanisms to prevent attacks from downloaded code worked
  - But users hated them
  - So now Windows doesn't use them

Fail-Safe Design

- 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 . . .
- In OS context, important to think about what happens with traps, interrupts, etc.

Tools For Securing Systems

- Physical security
- Access control
- Encryption
- Authentication
- Encapsulation
- Intrusion detection
- Filtering technologies

Physical Security

- Lock up your computer
  - Usually not sufficient, but . . .
  - Necessary (when possible)
- Networking means that attackers can get to it, anyway
- But lack of physical security often makes other measures pointless
  - A challenging issue for mobile computing

Access Control

- 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
- Obvious value in maintaining secrecy
- But clever use can provide other important security properties
- One of the most important tools in computer security
  - But not a panacea

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
- Especially difficult in distributed environments

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
- Operating system often plays a large role, here

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

Filtering Technologies

- Detect that there's something bad:
  - In a data stream
  - In a file
  - Wherever
- Filter it out and only deliver "safe" stuff
- The basic idea behind firewalls
  - And many other approaches
- Serious issues with detecting the bad stuff and not dropping the good stuff

Operating Systems and Security Tools

- Physical security is usually assumed by OS
- Access control is key to OS technologies
- Encapsulation in various forms is widely provided by operating systems
- Some form of authentication required by OS
- Encryption is increasingly used by OS
- Intrusion detection and filtering not common parts of the OS

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?
- The OS plays a major role in enforcing access control

Goals for Access Control

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

Common Mechanisms for Access Control in Operating Systems

- Access control lists
  - Like a list of who gets to do something
- Capabilities
  - Like a ring of keys that open different doors
- They have different properties
- And are used by the OS in different ways

A Common Problem For All Access Control Mechanisms

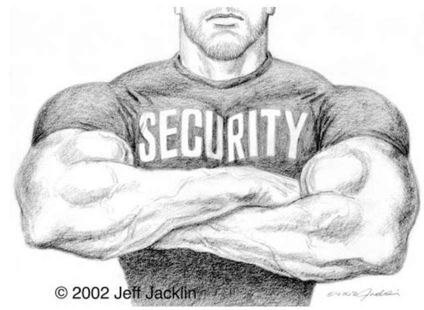
- Who gets to determine how they are set?
  - I.e., which subjects get to access which objects in what modes of use?
- How do you change the access permissions?
- In particular, who has the right to change them?
- And what mechanism is necessary to make the change?

Access Control Lists

- ACLs
- For each protected object, maintain a single list
- Each list entry specifies who can access the object
  - And the allowable modes of access
- When something requests access to a object, check the access control list

An Analogy

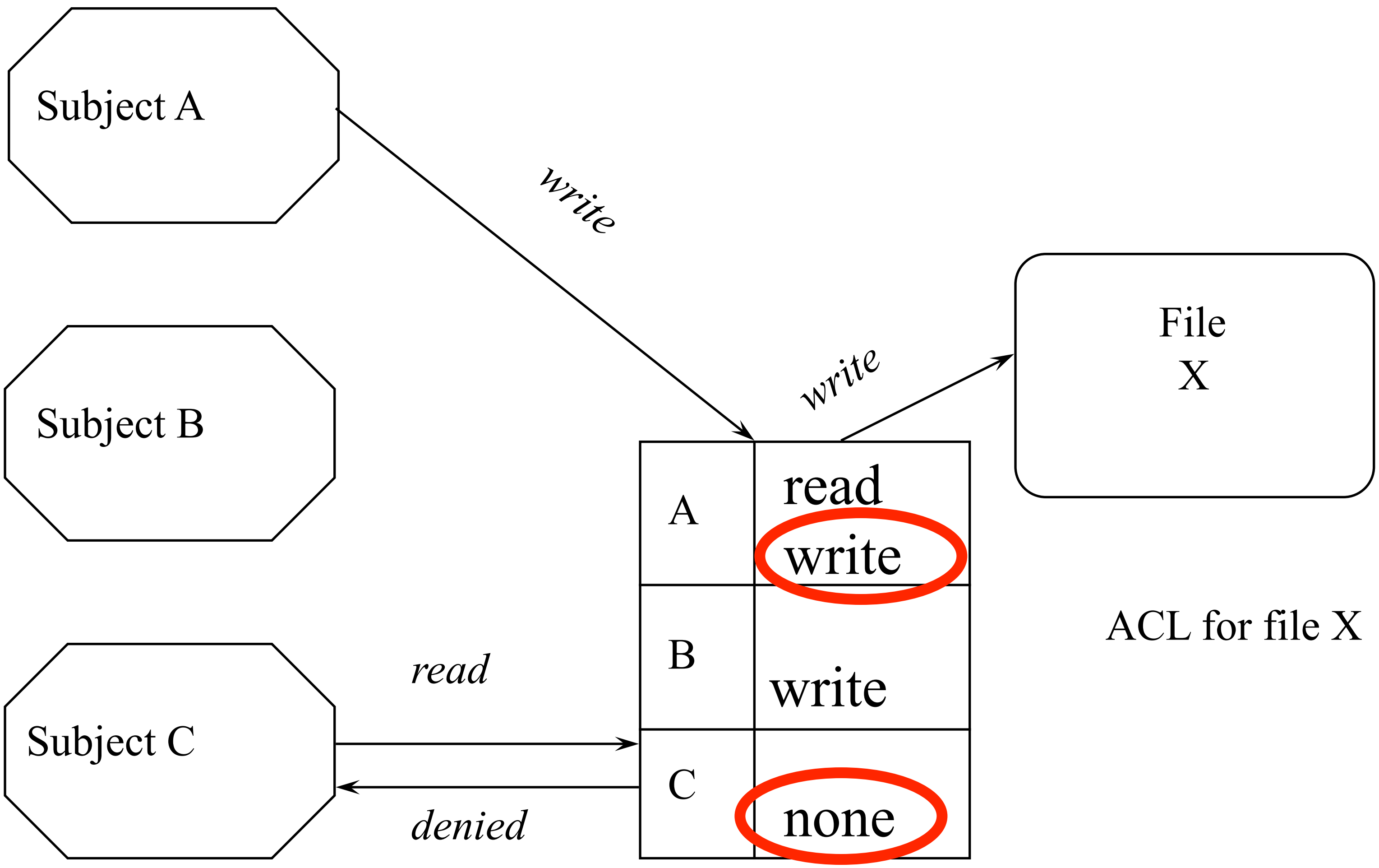
You're Not On the List!



This is an access control list

Joe Hipster

An ACL Protecting a File



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?
- Costs associated with complete mediation

An Example Use of ACLs: the Unix File System

- An ACL-based method for protecting files
  - Developed in the 1970s
- Still in very wide use today
  - With relatively few modifications
- Per-file ACLs (files are the objects)
- Three subjects on list for each file
  - Owner, group, other
- And three modes
  - Read, write, execute
  - Sometimes these have special meanings

Changing Access Permissions With ACLs

- Mechanically, the OS alone can change an ACL (in most systems)
- But who has the right to ask the OS to do so?
- In simple ACL systems, each object has an owner
  - Only the owner can change the ACL
  - Plus there's often a superuser who can do anything
- In more sophisticated ACL systems, changing an ACL is a mode of access to the object
  - Those with such access can give it to others
  - Or there can even be a meta-mode, which says if someone who can change it can grant that permission to others

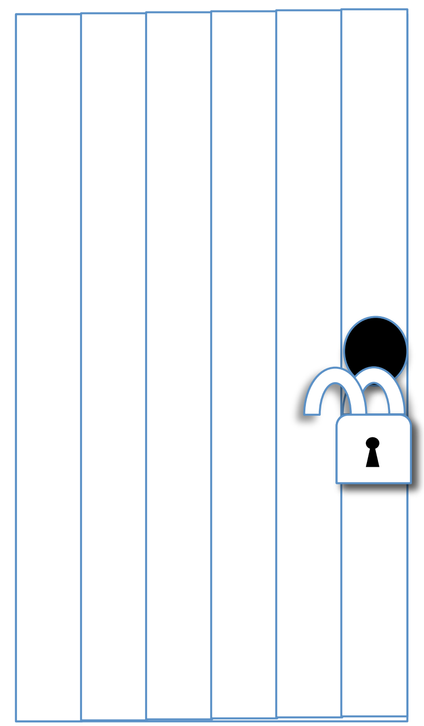
Pros and Cons of ACLs

+ 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

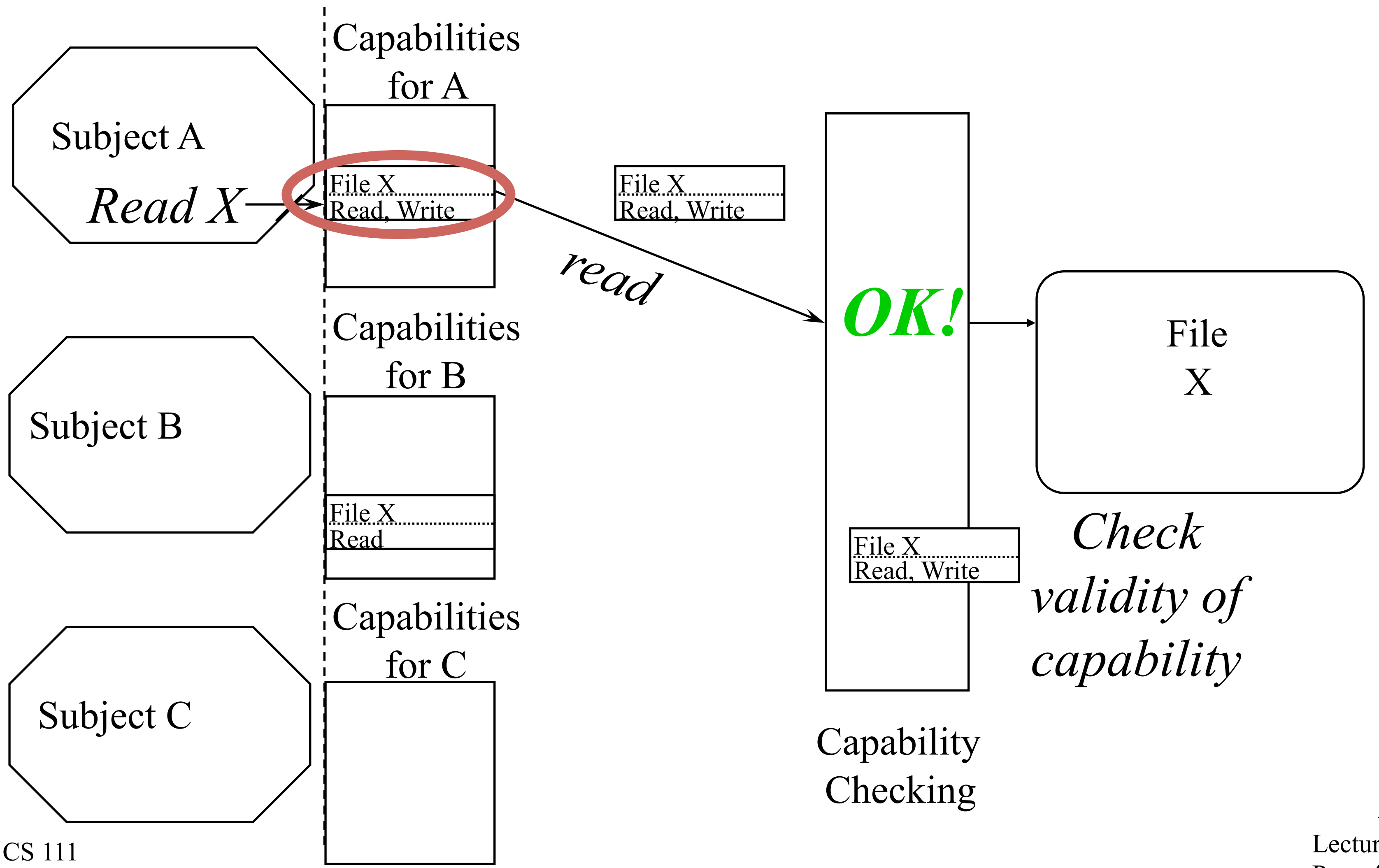
- Each entity keeps a set of data items that specify his allowable accesses
- Essentially, a set of tickets
- To access an object, present the proper capability
- Possession of the capability for an object implies that access is allowed

An Analogy

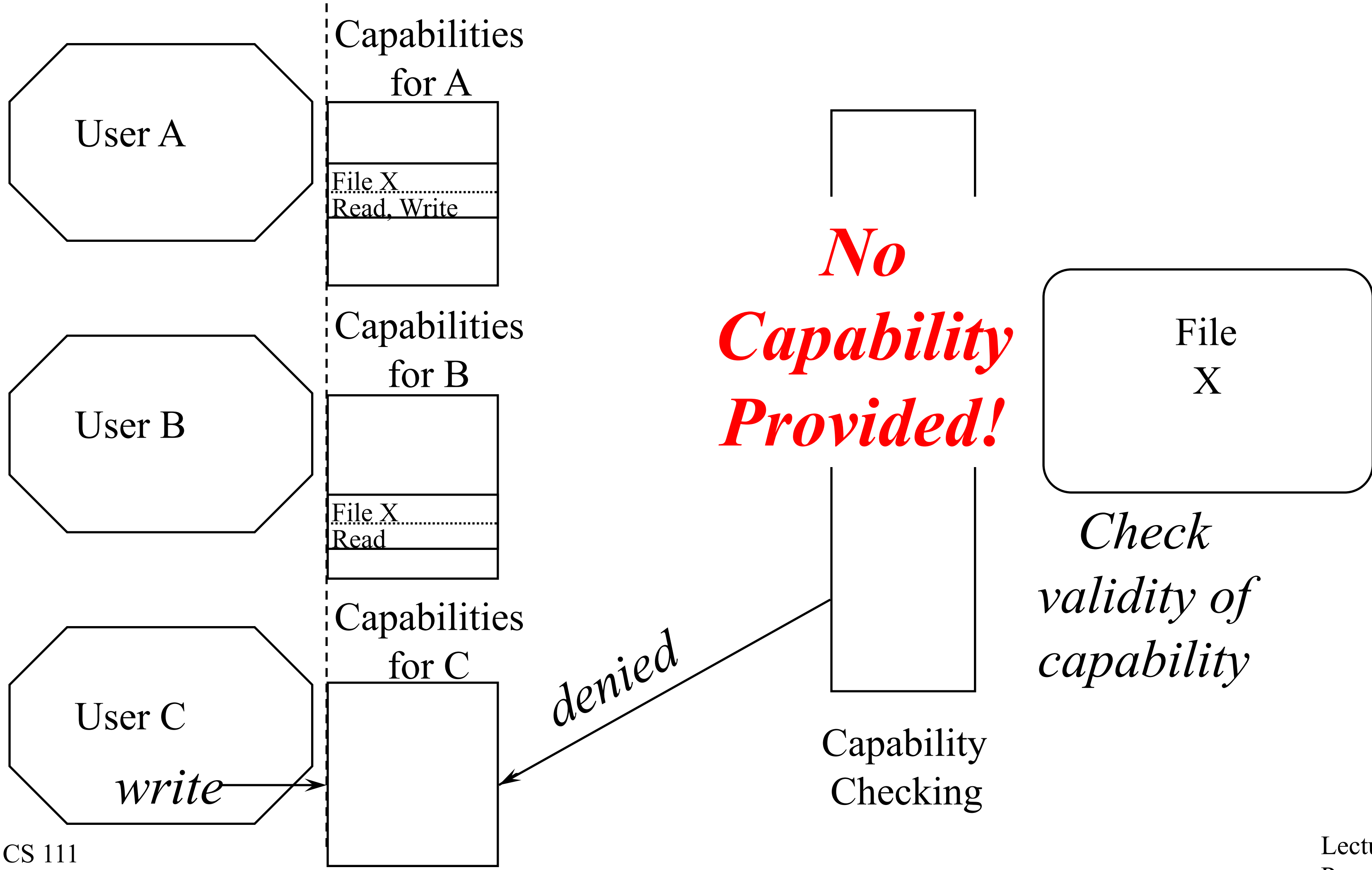


The key is a capability

Capabilities Protecting a File



Capabilities Denying Access

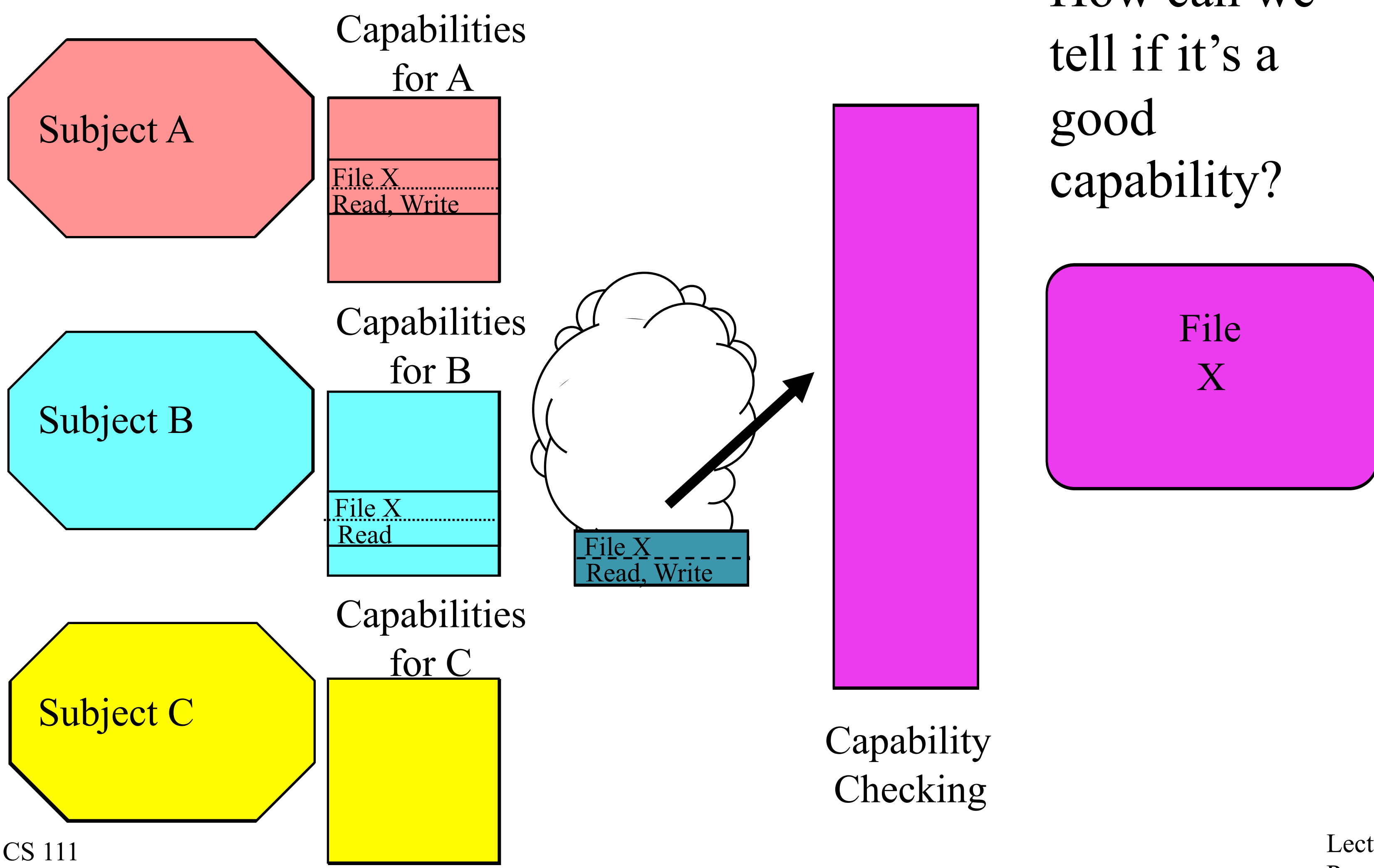


Properties of Capabilities

- Capabilities are essentially a data structure
  – Ultimately, just a collection of bits
- Merely possessing the capability grants access
  – So they must not be forgeable
- How do we ensure unforgeability for a collection of bits?
- One solution:
  – Don't let the user/process have them
  – Store them in the operating system

Capabilities and Networks

How can we tell if it's a good capability?



Cryptographic Capabilities

- Create unforgeable capabilities by using cryptography
  - We'll discuss cryptography in detail in the next lecture
- Essentially, a user CANNOT create this capability for himself
- The examining entity can check the validity
- Prevents creation of capabilities from nothing
  - But doesn't prevent copying them

Revoking Capabilities

- A simple problem for capabilities stored in the operating system
  - Just have the OS get rid of it
- Much harder if it's not in the operating system
  - E.g., in a network context
- How do we make the bundle of bits change from valid to invalid?
- Consider the real world problem of a door lock
- If several people have the key, how do we keep one of them out?

Changing Access Permissions With Capabilities

- Essentially, making a copy of the capability and giving it to someone else
- If capabilities are inside the OS, it must approve
- If capabilities are in user/process hands, they just copy the bits and hand out the copy
  - Crypto methods can customize a capability for one user, though
- Capability model often uses a particular type of capability to control creating others
  - Or a mode associated with a capability

Pros and Cons of Capabilities

+ Easy to determine what objects 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

OS Use of Access Control

- Operating systems often use both ACLs and capabilities
  - Sometimes for the same resource
- E.g., Unix/Linux uses ACLs for file opens
- That creates a file descriptor with a particular set of access rights
  - E.g., read-only
- The descriptor is essentially a capability

Enforcing Access in an OS

- Protected resources must be inaccessible
  - Hardware protection must be used to ensure this
  - So only the OS can make them accessible to a process
- To get access, issue request to resource manager
  - Resource manager consults access control policy data
- Access may be granted directly
  - Resource manager maps resource into process
- Access may be granted indirectly
  - Resource manager returns a “capability” to process

Direct Access To Resources

- OS checks access control on initial request
- If OK, OS maps it into a process' address space
  - The process manipulates resource with normal instructions
  - Examples: shared data segment or video frame buffer
- Advantages:
  - Access check is performed only once, at grant time
  - Very efficient, process can access resource directly
- Disadvantages:
  - Process may be able to corrupt the resource
  - Access revocation may be awkward
    - You've pulled part of a process' address space out from under it

Indirect Access To Resources

- Resource is not directly mapped into process
  - Process must issue service requests to use resource
  - Access control can be checked on each request
  - Examples: network and IPC connections
- Advantages:
  - Only resource manager actually touches resource
  - Resource manager can ensure integrity of resource
  - Access can be checked, blocked, revoked at any time
    - If revoked, system call can just return error code
- Disadvantages:
  - Overhead of system call every time resource is used
  - Making sure you catch every access