Access Controls

Peter J. Denning

Access Rights

- Object owners specify who has what access to their objects
- No access allowed without authorization

Reference Monitor Principle

- Every reference to an object is validated
- To attempt access
  - process calls a function in the API of an object manager
  - parameter is pointer to object
- Each attempted access checked for authorization
- Authorization checks performed by object managers
  - Virtual memory system checks accesses to frames
  - File system checks access to files
  - Directory system checks access to directories

Example: Page Access

- Virtual memory system is reference monitor
- 3-bit access code specifies which of rwe are allowed for each page
- Page access codes stored in page table, enforced by MMU

Example: File Access

- File system is reference monitor
- Operations: open, close, read, write
- 4-bit access code specifies which of ocrw are allowed

Access Matrix

A conceptual model to represent what access rights each user has for each object in the system

	O1 page	O2 file	O3 pipe	O4 dir	O5 VM
U1	r	o, c, r		search	start exit
U2	r, w		r	search insert	suspend resume
U3		w	w	search	start exit
U4	e	o, c, r, w		search rename	

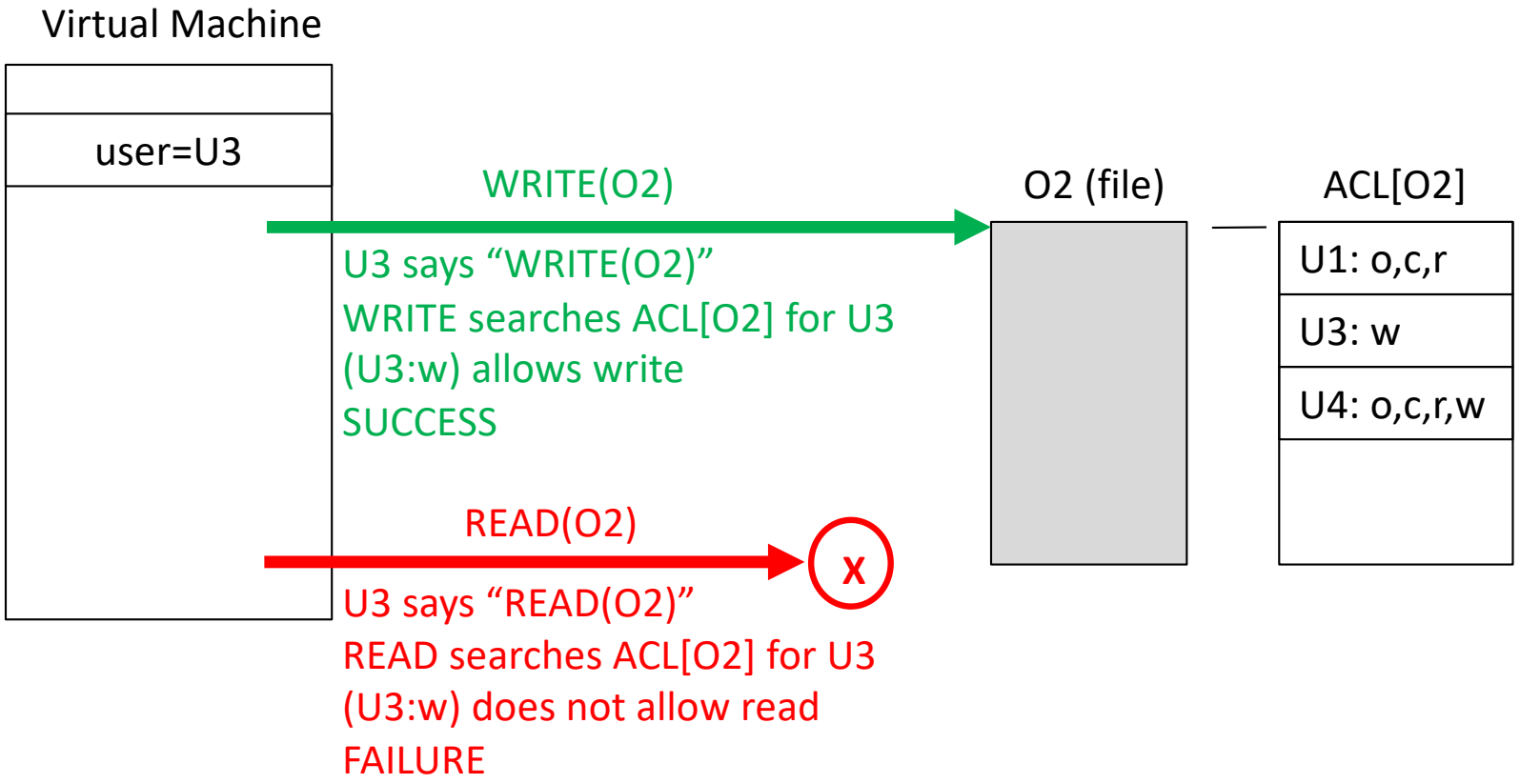
Representing Access Matrix

Storage by columns

- ACL -- Access Control List for particular object
- ACL entries of form (U,R) declaring that user U has rights R for the object
- ACL does not store blank access matrix entries

ACL linked to object

- Easy verification of requested access to object
- Edits take effect immediately



ACL Implementation Expensive

- Each attempted access requires a search of ACL to validate
- ACL can contain thousands of entries
- Need low overhead implementation

UNIX Method

- Classify users into three groups
  - Owner
  - Group selected by owner (by special commands)
  - Everyone else (the world)
- Specify 3-bit code (rwe) for each group
- The resulting 9-bit code is an ACL
- Store the 9-bit ACL in the directory entry

UNIX Method Critique

- Very fast … BUT
- No granularity
  - Typically owner v. world
  - Few use “groups”, not easy to update
- Cannot confine untrusted software
  - External objects must be enabled for “world”
  - Makes web-page based malware attacks easy

Capabilities: efficient alternative

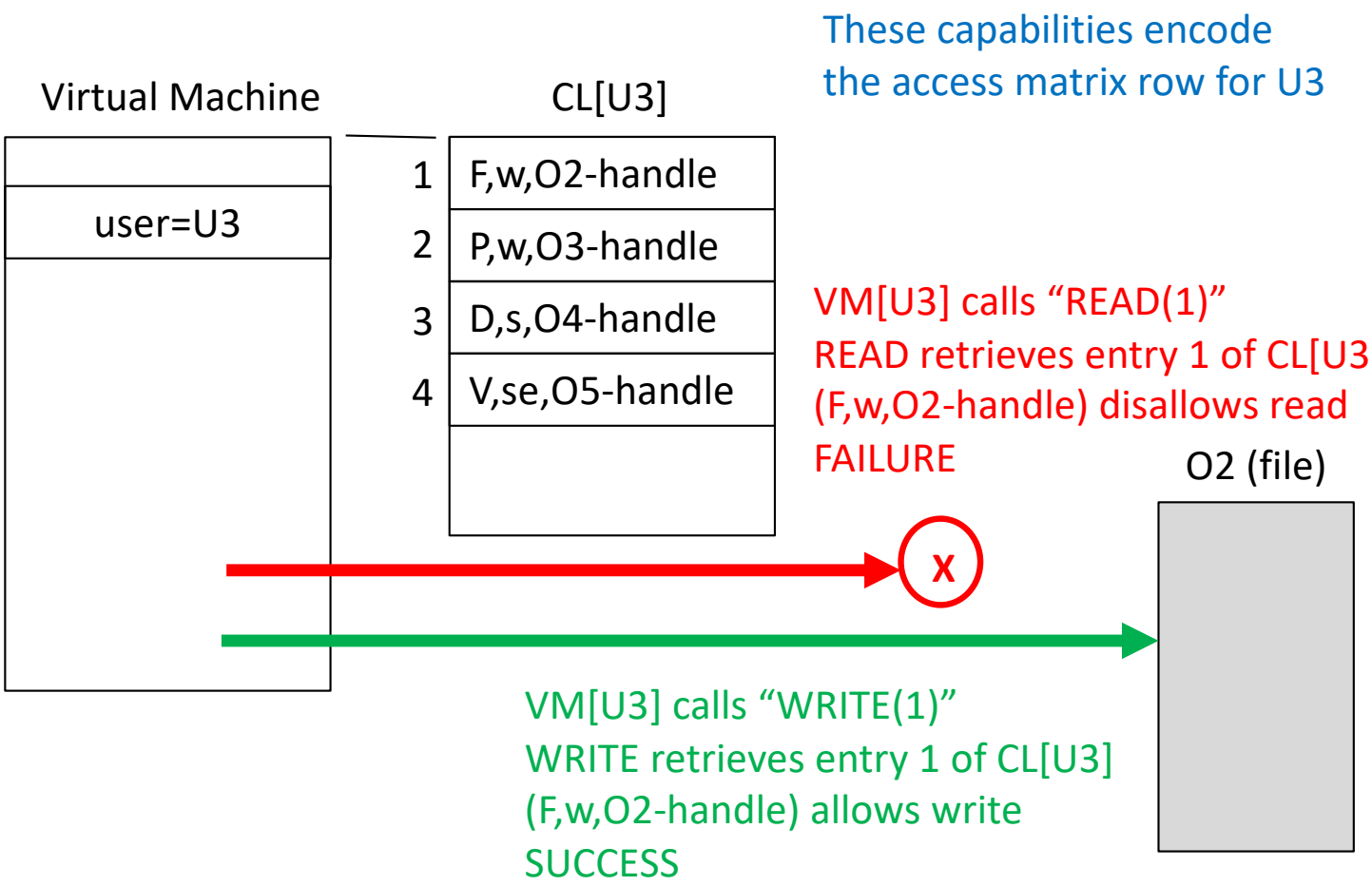
- File system a running example
- Virtual machine of user U asks to read a file with handle h
- File system verifies that U has read access to h
- Encode this verification as c=(file, r, h)
- This bundle c is called a capability

Capability is Access Ticket

- Now VM[U] uses c as pointer, READ_FILE(c)
  - Verify that c.type = file
  - Verify that c.access allows read
  - Map c.handle to file
- Very efficient, like UNIX, and allows for very fine-grained access controls, unlike UNIX
- Capability must be **unforgeable** -- protected from alteration after creation

Protecting Capabilities

- Keep them in protected tables in kernel space … give processes indirect address via the tables
  - Just like page tables
- The protected table of a process is called its C-list (capability list) and is stored in kernel memory
- Process can access only the objects listed in its C-list
- Pointer to C-list is in the process's Virtual Machine



Storage by rows

Capability lists can be viewed as encodings of the rows of access matrix

Summary

- Access control essential but difficult
- ACL
  - Natural and intuitive
  - Modified permissions immediately effective
- Capabilities
  - Pre-validated capabilities efficient
  - Easy to share but not easy to revoke