Access Control Mechanisms

- Access control lists
- Capabilities
- Locks and keys
  - Secret sharing

What's Wrong with ACM?

- Suppose we have 1k 'users' and 100k 'files' and users should only read/write their own files
  - The ACM will have 101k columns and 1k rows
  - Most of the 101M elements are either empty or identical
- Good for theoretical study but bad for implementation
- Why bother with the empty elements?

Access Control Lists

- Columns of access control matrix

	file1	*file2*	*file3*
Andy	rx	r	rwo
Betty	rwxo	r	
Charlie	rx	rwo	w

ACLs:

- file1: { (Andy, rx) (Betty, rwxo) (Charlie, rx) }
- file2: { (Andy, r) (Betty, r) (Charlie, rwo) }
- file3: { (Andy, rwo) (Charlie, w) }

3

Default Permissions

- Normal: if not named, *no* rights over file
  - Principle of Fail-Safe Defaults
- If many subjects, may use groups or wildcards in ACL
  - UNICOS: entries are (*user*, *group*, *rights*)
    - If *user* is in *group*, has rights over file
    - '*' is wildcard for *user*, *group*
      - (holly, *, r): holly can read file regardless of her group
      - (*, gleep, w): anyone in group gleep can write file

4

Abbreviations

- ACLs can be long ... so combine users
  - UNIX: 3 classes of users: owner, group, rest
  - rwx rwx rwx

    rest
    group
    owner
  - Ownership assigned based on creating process
  - Limited granularity
  - Cannot “exclude”

5

ACLs + Abbreviations

- Augment abbreviated lists with ACLs
  - Intent is to shorten ACL
- ACLs override abbreviations
  - Exact method varies
- Example: IBM AIX
  - Base permissions are abbreviations, extended permissions are ACLs with user, group
  - ACL entries can add rights, but on deny, access is denied

6

Permissions in IBM AIX

```
attributes:
base permissions
  owner(bishop):  rw-
  group(sys):     r--
  others:         ---
extended permissions enabled
  specify         rw-   u:holly
  permit          -w-   u:heidi, g=sys
  permit          rw-   u:matt
  deny            -w-   u:holly, g=faculty
```

7

ACL Modification

- Who can do this?
  - Creator is (typically) given *own* right that allows this
  - System R provides a *grant* modifier (like a copy flag) allowing a right to be transferred, so ownership not needed
    - Transferring right to another modifies ACL

8

Privileged Users

- Do ACLs apply to privileged users (*root* in UNIX or *administrator* in Windows)?
  - Solaris UNIX: abbreviated lists are ignored for root subjects, but full-blown ACL entries are not
  - Other vendors: varies

9

ACLs and Groups and Wildcards

- Classic form: neither; in practice, usually
  - AIX: base perms gave group sys read only

    ```
    permit    -w-    u:heidi, g=sys
    ```

    line adds write permission for heidi when in that group
  - UNICOS:
    - holly : gleep : r
      - user holly in group gleep can read file
    - holly : * : r
      - user holly in any group can read file
    - * : gleep : r
      - any user in group gleep can read file

10

Conflicts: three possibilities

- Allow access if any entry would allow access
- Deny access if any entry would deny access
  - AIX: if any entry denies access, *regardless of where the entry is*, access is denied
- Apply first entry matching subject
  - Cisco routers: run packet through access control rules (ACL entries) in order; on a match, stop, and forward the packet; if no matches, deny
    - Note default is deny so honors principle of fail-safe defaults

11

Revocation Question

- How do you remove subject's rights to a file?
  - Owner deletes subject's entries from ACL, or rights from subject's entry in ACL
- What if ownership not involved?
  - Depends on system
  - System R: restore protection state to what it was before right was given
    - May mean deleting descendent rights too …

12

Windows NT ACLs

ACL for files on NTFS partition

Different sets of rights

- **Basic**: read, write, execute, delete, change permission, take ownership
  - grouped for files into sets called:
- **Generic**: no access, read (read/execute), change (read/write/execute/delete), full control (all), special access (assign any of the basics)
  - And for directories into sets called
- **Directory**: no access, read (read/execute files in directory), list, add, add and read, change (create, add, read, execute, write files; delete subdirectories), full control, special access

Accessing Files in WindowsNT

1. User not in file's ACL nor in any group named in file's ACL: deny access
2. ACL entry denies user access: deny access (overrides grants)
3. If user is present, take union of rights of all ACL entries giving user access: user has this set of rights over file

Capability Lists

- Rows of access control matrix

	file1	*file2*	*file3*
Andy	rx	r	rwo
Betty	rwxo	r	
Charlie	rx	rwo	w

C-Lists:

- Andy: { (file1, rx) (file2, r) (file3, rwo) }
- Betty: { (file1, rwxo) (file2, r) }
- Charlie: { (file1, rx) (file2, rwo) (file3, w) }

Semantics

- Like a bus ticket
  - Mere possession indicates rights that subject has over object
  - Object identified by capability (as part of the token)
    - Name may be a reference, location, or something else
  - Architectural construct in capability-based addressing; this just focuses on protection aspects
- ACL controlled by OS, capabilities in part by process
  - Must prevent process from altering capabilities
    - Otherwise subject could change rights encoded in capability or object to which they refer

Implementation

Three mechanisms to protect capabilities

- Tagged architecture
  - Bits protect individual words (only read or change too)
- Paging/segmentation protections
  - Like tags, but put capabilities in a read-only segment or page
  - Programs must refer to them by pointers
    - Otherwise, program could use a copy of the capability—which it could modify

17

Implementation (*cont.*)

- Cryptography
  - Associate with each capability a cryptographic checksum enciphered using a key known to OS
  - When process presents capability, OS validates checksum
  - Example: Amoeba, a distributed capability-based system
    - Capability is (*name*, *creating_server*, *rights*, *check_field*) and is given to owner of object
    - *check_field* is 48-bit random number; also stored in table corresponding to *creating_server*
    - To validate, system compares *check_field* of capability with that stored in *creating_server* table
    - Makes forging a capability difficult
    - ***Vulnerable if capability disclosed to another process***

18

Revocation

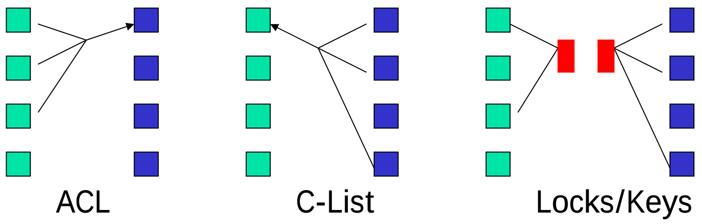
- Scan all C-lists, remove relevant capabilities
  - Far too expensive!
- Use indirection
  - Each object has entry in a global object table
  - Names in capabilities name the entry, not the object
    - To revoke, invalidate the entry in the table
    - Can have multiple entries for a single object to allow control of different sets of rights and/or groups of users for each object

ACLs vs. Capabilities

- Theoretically equivalent; consider 2 questions
  1. Given a subject, what objects can it access, and how?
  2. Given an object, what subjects can access it, and how?
  - ACLs answer second easily; C-Lists, first
- Probably the second question has been of most interest in the past, hence ACL-based systems more common than capability-based systems
  - As first question becomes more important (in incident response, for example), this may change

Locks and Keys

- Associate *lock* with object and *key* with subject
  - Latter controls what the subject can access and how
  - Subject presents key; if it corresponds to any of the locks on the object, access granted
- This is more flexible
  - Change either locks or keys



Cryptographic Implementation

- Enciphering key is lock; deciphering key is key
  - Encipher object o; store $E_k(o)$
  - Use subject's key k' to compute $D_{k'}(E_k(o))$
  - Any of n subjects can access o (**OR**-access): store

$$o' = (E_1(o), \ldots, E_n(o))$$

  - Requires consent of all n to access o (**AND**-access): store

$$o' = (E_1(E_2(\ldots(E_n(o))\ldots))$$

Type Checking

- Lock is type, key is operation
  - Example: UNIX system call *write* cannot work on directory object but does work on file
  - Example: distinguish Instruction and Data spaces of PDP-11
    - execute only on instructions,
    - read/write only on data
  - Example: countering buffer overflow attacks on the stack by putting stack on non-executable pages/segments
    - Then code uploaded to buffer will not execute
    - Does not stop other forms of this attack, though ...

Sharing Secrets

Related to locks and keys: How to construct a control to allow certain subsets of subjects to access an object

- Implements separation of privilege
- Use (*t*, *n*)*-threshold scheme*
  - Data divided into *n* parts
  - Any *t* parts sufficient to derive original data
- Cryptographic approaches are a common way to implement it

Shamir's Scheme

- Goal: use (*t*, *n*)-threshold scheme to share cryptographic key encoding data
  - Based on Lagrange polynomials
  - Idea: take polynomial *p*(*x*) of degree *t*–1, set constant term (*p*(0)) to key
  - Compute value of *p* at *n* points, *excluding* x = 0
  - By algebra, need values of *p* at any *t* distinct points to derive polynomial, and hence constant term (key)

25

Key Points

- Access control mechanisms provide controls for users accessing files
- Many different forms
  - ACLs, capabilities, locks and keys
    - Type checking too
  - ~~Ring-based mechanisms (Mandatory)~~
  - ~~PACLs (ORCON)~~

26