Access Control

- Chapter 2
  - Access Control Matrix Model
- Chapter 14
  - Access Control Mechanisms
- Chapter 4
  - Access Control Models

Overview

- Protection (secure) state of system
  - Describes current settings, values of system relevant to protection
- Access control matrix
  - **Describes protection state precisely**
  - Matrix describing rights of subjects
  - State transitions change elements of matrix

Description

objects (entities)

subjects

	o_1	…	o_m	s_1	…	s_n
s_1						
s_2						
…						
s_n						

- Subjects $S = \{ s_1,\ldots,s_n \}$
- Objects $O = \{ o_1,\ldots,o_m \}$
- Rights $R = \{ r_1,\ldots,r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

Description

- Objects (destinations)
  - Entities to be protected
  - Files, devices, memory blocks, instructions, functions, processes (services), hosts (addresses)
- Subjects (sources)
  - Entities that access the objects
  - Processes, users, hosts (addresses)

Description

- Rights (access rights)
  - Category of rights
    - Unix: read, write, execute, own (rights to change rights other than ownership)
    - Windows: ...
    - Network: connect, download, upload, ...
  - How are the rights related to confidentiality, integrity, availability?

Description

- Interpretation of rights
  - Read a file
  - Read a directory
    - In a file system
    - In a web server
  - Execute a file
  - Execute a directory
    - In a file system
    - In a web server

Case (homework)

- Policy: no copy of other's homework
- Subject: Alice, Bob, Charlie
- Object: A's hw, B's hw, C's hw
- Right: copy(read)
- ACM: ?
- Mechanism: ?

for directories
-x can't go inside (ideal)
-r ls in dir
-w name new object in dir

Case (continue)

- Incident: Charlie deleted Alice's homework.
- What's the security problem? Policy or mechanism?
- Improvement: policy, subjects, objects, rights, acm, mechanisms.

Case (web)

- Policy
  - All students and instructors can access TRACs.
  - Only instructors can access solutions.
- ACM?
  - Subjects, objects, rights
  - ACM
  - Mechanisms

Example 1 : File System

- Processes *p*, *q*
- Files *f*, *g*
- Rights *r*, *w*, *x*, *a*, *o*

	f	*g*	*p*	*q*
p	*rwo*	*r*	*rwxo*	*w*
q	*a*	*ro*	*r*	*rwxo*

Example 2 : Programming

- Procedures *inc_ctr*, *dec_ctr*, *manage*
- Variable *counter*
- Rights +, –, *call*

	counter	*inc_ctr*	*dec_ctr*	*manage*
inc_ctr	+			
dec_ctr	–			
manage		*call*	*call*	*call*

Access Control

- Chapter 2
  - Access Control Matrix Model
- Chapter 14
  - Access Control Mechanisms
- Chapter 4
  - Access Control Models

Access Control Mechanisms

- Access control lists
- Capabilities
- Ring-based access control

Implementation of Access Control

Policy

- Define security policy
- Create the access control matrix

Mecha-nism

- Implement and put the ACM in a storage
- When a user (s) requests an operation (p) on an object (o)
  - Check if p is in the entry A(s,o) of the ACM.
  - Yes, proceed; No, deny.

Problems

- Problems using a raw access control matrix
  - Too many subjects and objects
  - Too many blank entries
  - Too complicated operations to create and delete entries
- Problems in solutions
  - Restrict access control matrix in some manner
  - No precise solution

Access Control Lists

- Columns of access control matrix

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

ACLs: no empty entries

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

Access Control Lists

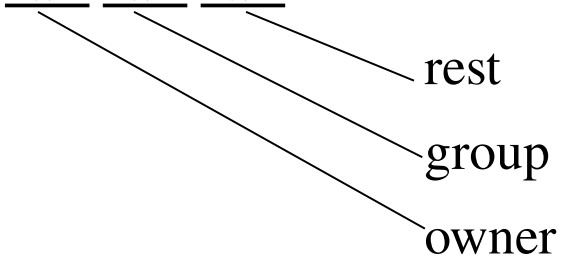
- Define security policy
- Create the access control matrix
- Implement and put the ACL in a storage
- When a user (s) requests an operation (p) on an object (o)
  - Retrieve the ACL(o)
  - Check if (s,p) matches ($\in$) an entry of ACL(o).
  - Yes, proceed; No, deny.

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

Abbreviations

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



  - Ownership assigned based on creating process
    - Some systems: if directory has setgid permission, file group owned by group of directory (SunOS, Solaris)

Problem of Abbreviation

- Problem : loss of granularity
  - "Everyone but Alice can read"
    - Create a group excluding Alice
    - Set r on the group
  - "Everyone can read" + "Everyone but Alice can write"
    - Create a group excluding Alice
    - Set rw on the group, r on others
  - "Everyone but Alice can read" + "Everyone but Bob can write"
    - Abbreviation?

Abbreviations + ACLs

- 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

Abbreviations + ACLs

- Procedure
  - Get the base permission S from the abbreviated base permissions.
  - Modify S according the entries of the ACL
    - If an entry denies the requested permission, stop.
    - If no more entry, S is the permissions granted to the requesting user.

Abbreviations + ACLs

```
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
```

Abbreviations + ACLs in Linux

- Enable ACL to the file system
  - /etc/fstab
- Grant or revoke permissions in the ACL
  - getfacl
  - setfacl

```
# file: foo
# owner: qijun
# group: qijun
user::rw-
user:root:r--
group::---
mask::r--
other::---
```

ACL Issues

- Who can modify the ACL of an object?
- What is the role of a privileged user?
- Does ACL support groups and wildcards?
- How are contradictory permissions handled?
- How to handle default permissions?
- How to revoke a subject's permission?

ACL Modification

- Which subjects can modify an object's ACL?
  - The right to change rights
  - Creator is 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

Privileged Users

- Do the ACLs apply to privileged users (*root*)?
  - Solaris: abbreviated lists do not, but full-blown ACL entries do
    - If abbreviation denies read to root, root can read.
    - If ACL denies read to root, root cannot read
  - Linux: no, in most cases
  - Other vendors: varies

Groups and Wildcards

- Does the ACL support groups and wildcards?
- Classic form: no; in practice, usually
  - AIX: base perms gave group sys read only

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

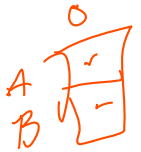
    line adds write permission for heidi when in that group
  - UNICOS:
    - holly : gleep : r
      - user holly in group gleep can read file
    - holly : * : r
      - user holly in any group can read file
    - * : gleep : r
      - any user in group gleep can read file

Conflicts

- Deny access if any entry would deny access
  - AIX: if any entry denies access, *regardless or rights given so far*, access is denied
- All access if any entry would allow access
- 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

Conflicts

- Examples
  - Subjects: A is a subset of B
  - Object: O
  - A is allowed to read C, but others in B cannot.
  - O's ACL should be {(A, r), (B-A, -)}
    - First apply: {(A, r) (B, -)} or {(B, -) (A, r)}?
    - Deny access: {(A, r) (B, -)} or {(B, -) (A, r)}?
    - Allow access: {(A, r) (B, -)} or {(B, -) (A, r)}?



Handling Default Permissions

- Apply ACL entry, and if none use defaults
  - Cisco router: apply matching access control rule, if any; otherwise, use default rule
- Augment defaults with those in the appropriate ACL entry
  - AIX: extended permissions augment base permissions

Revocation Question

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

Example: iptables

- Linux firewall
  - Commands in Fedora/Centos/Redhat
    - Must be root
  - Enable or disable
    - chkconfig iptables on|off
    - systemctl enable|disable iptables
  - Status
    - service iptables status
    - systemctl status iptables
  - Start or stop
    - service iptables start|stop
    - systemctl start|stop iptables

iptables

- Firewall rules are organized in chains
  - Three default chains
  - INPUT: for incoming packets
  - FORWARD: for traversing packets
  - OUTPUT: for outgoing packets
- All network packets are inspected according to firewall rules in order
- The packet is accepted or rejected on the first matching rule

Example

- Built-in chains

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

- Other

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
f2b-SSH    tcp  --  anywhere             anywhere          tcp dpt:ssh
ACCEPT     all  --  anywhere             anywhere          state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere          state NEW tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere          state NEW tcp dpt:http
REJECT     all  --  anywhere             anywhere          reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  anywhere             anywhere          reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain f2b-SSH (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

Chain operations

- iptables -? chain [rule-specs]
- -P target : default chain policy
  - target : ACCEPT, DROP, RETURN, another chain
- -F [chain] : delete(flush) rules of the chain
- -X chain : delete the chain
- -N chain : create a new chain
- -E old-chain new-chain : rename the chain
- -L chain : list the chain
- -S chain : print the chain
- -A chain rule-spec : add a rule to the chain
- -D chain rule-num : delete the num-th rule of the chain
- -I chain rule-num rule-spec : insert a rule before the num-th rule
- -R chain rule-num rule-spec : replace the num-th rule with a rule

Rule Specs

- A combination of the following (common use)
  - -i ifc -o ifc -p protocol -m pattern
    -s address[/mask][,…] --sport spnum
    -d address[/mask][,…] --dport dpnum
    -j target
- Access control matrix
  - subject
  - object
  - right

Example

- iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
- iptables -A INPUT -p tcp --dport 80 -j ACCEPT
- iptables -A INPUT -i lo -j ACCEPT
- iptables -A INPUT -j REJECT --reject-with icmp-host-prohibited
- iptables -A OUTPUT -p tcp -d www.facebook.com -j DROP
- iptables -A INPUT -s 192.168.1.0/24 -j DROP
- iptables -A INPUT -p icmp -j ACCEPT
- iptables -N f2b-SSH
- iptables -A INPUT -p tcp --dport 22 -j f2b-SSH
- iptables -A f2b-SSH -s 218.65.30.4/32 -j REJECT --reject-with icmp-port-unreachable
- iptables -A f2b-SSH -j RETURN

Capability Lists

- ACL: When Subjects and Objects are Changing
  - Add a subject
  - Add an object
  - Remove a subject
  - Remove an object
- Is ACL good for all systems?
  - A file system: files are objects and change often.
  - A web server: users are subjects and change often.

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) }

Capability Lists

- Define security policy
- Create the access control matrix
- Implement and put the CL in a storage
- When a user (s) requests an operation (p) on an object (o)
    - Retrieve the CL(s)
    - Check if (o,p) matches ($\in$) an entry of CL(s).
    - Yes, proceed; No, deny.

Capability Example

- File descriptor
  - int fd = open(“file”, O_RDONLY);
  - The fd can be transferred from one process to another process.
  - But, no matter which process obtains fd, the process can only read the file.
  - So, fd references to a capability on the file (o) with the read (p) permission.

ACLs vs. Capabilities

- The procedure
  - A subject or an object is created.
  - Rights are added to ACLs or CLs
  - When a subject tries to access an object, rights are checked.
- Assume a system of 100 subjects and 1000 objects.

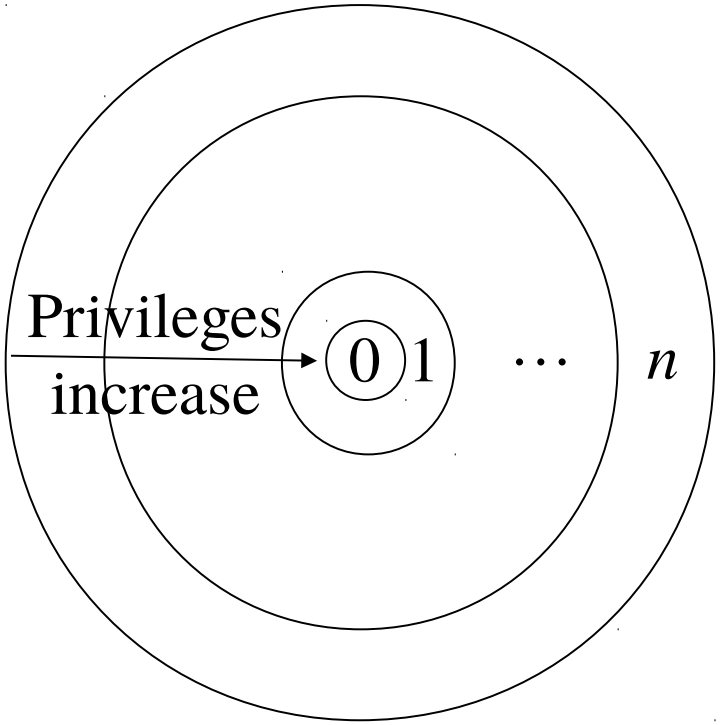
ACLs vs. Capabilities

- ACLs
  - Add a subject
  - Add an object
  - Remove a subject
  - Remove an object
  - Copy an object and its permissions
- CLs

ACLs vs. Capabilities

- Both 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
- Suggested that the second question, which in the past has been of most interest, is the reason ACL-based systems more common than capability-based systems
  - As first question becomes more important (in incident response, for example, that has many unknown subjects), this may change.

Ring-Based Access Control



- Process (segment) accesses another segment
  - Read
  - Execute
- *Gate* is an entry point for calling segment
- Rights:
  - *r* read
  - *w* write
  - *e* execute

Reading/Writing/Appending

- Procedure executing in ring r
- Data segment with *access bracket* (a_1, a_2)
- Mandatory access rule
  - $r \leq a_1$ allow access
  - $a_1 < r \leq a_2$ allow r access; not w access
  - $a_2 < r$ deny all access

Rings and Access Control Matrix

- A process with ring 3 needs to access an object with an access bracket [2, 4]
  - What access does the process have?
  - How to describe this access control in access control matrix?
  - What kind of access control matrix cannot rings implement?
    - S1 and S2; O1 and O2
    - S1 rw O1; S1 r O2
    - S2 r O1; S2 rw O2
    - Not possible

Executing

- Procedure executing in ring r
- Call procedure in segment with *access bracket* (a_1, a_2) and *call bracket* (a_2, a_3)
  - Often written (a_1, a_2 , a_3)
- Mandatory access rule
  - $r < a_1$ allow access; ring-crossing fault
  - $a_1 \leq r \leq a_2$ allow access; no ring-crossing fault
  - $a_2 < r \leq a_3$ allow access if through valid gate
  - $a_3 < r$ deny all access

Versions

- Multics
  - 8 rings (from 0 to 7)
- Digital Equipment's VAX
  - 4 levels of privilege: user, monitor, executive, kernel
- Older systems
  - 2 levels of privilege: user, supervisor

Access Control

- Chapter 2
  - Access Control Matrix Model
- Chapter 14
  - Access Control Mechanisms
- Chapter 4
  - Access Control Models

Models of Access Control

- Involved entities in AC
  - System, subject, object, the owner of the object, the creator of the object
  - System checks AC, but not necessarily sets AC.
- Discretionary AC : owner
- Mandatory AC : system
- Originator Controlled AC : creator

Models of Access Control

- Discretionary Access Control (DAC, IBAC)
  - Individual user sets access control mechanism to allow or deny access to an object
- Example
  - A child controls who can read his/her diary.
  - A user controls the access of the files he/she owns.

Models of Access Control

- Mandatory Access Control (MAC)
  - System mechanism controls access to object, and individual cannot alter that access
- Example
  - The law allows a court to access individual's driving records.
  - The system controls the ownership of the files.

Models of Access Control

- Originator Controlled Access Control (ORCON)
  - Originator (creator) of information controls who can access information
- Example
  - The distribution of a company's software is only allowed by the company.
    - A user owns (purchased) the software.