Lecture 2

The Access Control Matrix Model and its Implementations

Knowledge is Structured in Consciousness

Wholeness Statement

The confidentiality and integrity of data depend on access control. An Access Control Matrix specifies an access control policy which can be implemented in many ways. Access to pure consciousness is available to anyone and everyone who knows the technique of effortlessly turning the attention inwards.

Overview

1. Eight design principles for implementing security mechanisms
2. The Access Control Matrix Model
3. Methods for implementing an Access Control Matrix
4. Six consumer technologies that are destroying traditional IT (security)

Subjects and Objects

- *Object*: the set of all protected entities
  - I.e., entities relevant to the protection state of the system
  - *Protection state*: the subset of the state of the system that deals with protection
  - Objects are files, devices, and processes, but could be other entities such as messages
- *Subject*: the set of all active objects such as processes and users

Eight Fundamental Design Principles

(Chapter 12 , pp. 199-208)

Eight Design Principles (Chapter 12)

- These principles underlie the design and implementation of the mechanisms for enforcing security policies
- We will revisit these over and over again throughout the course

Design Principles

1. *Principle of Least Privilege*: A subject should only be given those privileges needed to do its task
   (SCI: disallow the birth of an enemy)
   Example: Faculty don't get administrator privileges

2. *Principle of Fail-Safe Defaults*: Unless a subject is given explicit access to an object, access should be denied
   (SCI: spontaneous right action)
   Example: Default settings maximize security. To open something up, user has to do it.

Design Principles

3. *Principle of Economy of Mechanism*: security mechanisms should be as simple as possible
   (SCI: do less and accomplish more)
   Example: Simple programs have fewer bugs for attackers to exploit
   - Exception: an OS that doesn't check passwords would be simpler but less secure,
   - Exception: the JVM, which checks array bounds, is more complex but more secure than native code

4. *Principle of Complete Mediation*: all accesses to objects must be checked to ensure that they are allowed
   (SCI: infinity at a point)
   Example: Assume nothing, always check security

Design Principles

5. *Principle of Open Design*: the security of a mechanism should NOT depend on the secrecy of its design or implementation
   (SCI: all this is That)
   Example: Don't assume that attacker can't reverse engineer a program.

6. *Principle of Separation of Privilege*: a system should not grant permission based on a single condition
   (SCI: sequential unfoldment)
   Example: Packet has to pass through two firewalls to get to internal network

Design Principles

7. *Principle of Least Common Mechanism*: mechanisms used to access resources should not be shared
   (SCI: cultural integrity)
   Example: Reduce sharing; keep important data and programs off the network if possible.

8. *Principle of Psychological Acceptability*: security mechanisms may add some extra burden, but that burden must be both minimal and reasonable
   (SCI: the nature of life is the expansion of happiness)
   Example: Configuring and executing a program should be as easy and as intuitive as possible, e.g., understandable messages etc., otherwise system administrators may not set the software up with the proper security. On the other hand, messages must not impart unnecessary information

The Access Control Matrix Model and its Implementations

Chapter 2 (pp 27-30) and
Chapter 14

Access Control Matrix (ACM)

- *Objects* are the set of protected entities
- *Subjects* are the set of active objects in the system
- An ACM is a way of specifying policy,
  - namely, what are the access permissions of the subjects of the system
- Suppose the set of rights is
  {read, write, execute, append, own}
- So what would reading from a process mean?
  - Could allow P1 to accept messages from P2
  - Or could allow P1 to look at the state of P2

Access Control Matrix (ACM)

Consider the following system and ACM with two processes and two files

	file1	file2	process1	process2
process1	read, write, own	read	read, write, execute, own	write
process2	append	read, own	read	read, write, execute, own

ACM Example

- The set of host rights on an LAN are
  {ftp, mail, nfs, own}

Host name	telegraph	nob	toadflax
telegraph	own	ftp	ftp
nob		ftp, nfs, mail, own	ftp, nfs, mail
toadflax		ftp, mail	ftp, nfs, mail, own

ACM

- Only the allowed permissions are in the ACM
  - A blank square in the ACM means that the associated process has no permissions on the associated object
  - This follows the *principle of fail-safe defaults* (why?)
- An ACM is very abstract and is used to specify policy
- Next we will look at how an ACM is implemented
  - If the ACM specifies security policy, then what is its implementation called?

ACM Implementation

- An ACM policy can be implemented (mechanism) in various ways.
  a. Access control list (focuses on the columns of the ACM)
  b. Capability list (focuses on the rows of the ACM)
- Groups and roles can make administration easier.

Propagated Access Control List (PACL)

- So far we have only focused on controlling access to a specific object
- However, the initial creator of an object may also want to control all future accesses to copies of the information in the object
- This kind of control can be achieved using a PACL

Who assigns permissions?

- ACM policy can also specify who assigns permissions.
- Three possibilities:
  - discretionary,
  - mandatory, or
  - originator-controlled

Access Control Matrix

Protection State

- The *state of a system* is the collection of the current values of
  1. all memory locations
  2. all secondary storage
  3. all registers
  4. all other components
- The subset of the state of the system that deals with protection is called the *protection state*
- An *access control matrix* describes the current protection state

Authorized States

- States in which a system is allowed to reside
- For example:
  - Let P be the set of all possible protection states
  - Let Q be the set of authorized states
    - Q is clearly a subset of P
  - Whenever the system is in Q, the system is *secure*
  - Whenever the system is in P-Q, it is *non-secure*
- A *security policy* characterizes the states in Q
- A *security mechanism* prevents a system from entering a state in P-Q

Access Control Matrix Model

- Describes a protection system by characterizing the rights of each subject (active entity) with respect to every other entity.
- ACM is policy.
- The actual implementation (mechanism) can take many forms
  - (e.g., ACL, capability lists, roles, etc.)
- Later, we look at four ways that an ACM can actually be implemented.

ACM and the OS

- Conceptually an OS consults the ACM to determine if an attempt to transition to a new state is secure
- For example, a process run by Bob wants to change its state from not having read a file to having read a file
  - If Bob is not allowed to read the file, this transition must be disallowed by the OS

ACM Examples

Unix file system

S = {process1, process2}
O = {file1, file2, process1, process2}
R = {own, read, write, append, execute}

	file1	file2	process1	process2
process1	read, write, own	read	own, read, write, execute	write
process2	append	read, own	read	own, read, write, execute

Local Area Network

S = {telegraph, nob, toadflax}
O = {telegraph, nob, toadflax}
R = {ftp, mail, nfs, own}

Host name	telegraph	nob	toadflax
telegraph	own	ftp	ftp
nob		ftp, nfs, mail, own	ftp, nfs, mail
toadflax		ftp, mail	ftp, nfs, mail, own

Programs

subjects are procedures
objects also include variables

S = { inc_ctr, dec_ctr, manager }
O = { counter, inc_ctr, dec_ctr, manager }
R = { +, -, call }

	counter	inc_ctr	dec_ctr	manager
inc_ctr	+			
dec_ctr	-			
manager		call	call	call

User Based ACM

(processes run on behalf of a user)

S = { bob, jill, jack }
O = { file1, file2, file3 }
R = { read, write, execute, append, own }

	file1	file2	file3
bob	own, read, write	read	read, write
jill	append	own, read	read, write
jack			own, read, write

Main Point

1. ACM is the simplest way to represent access control. This simple notation gives rise to all access control schemes. Pure consciousness is the simplest form of awareness that gives rise to all of manifest creation.

Access Control Mechanisms

(Chapter 14)

Access Control List (ACL)

- Each object is associated with pairs that describe the subjects that can access the object and with what permissions.
- An implementation of each column of the ACM is associated with the object.

User Based ACM

(processes run on behalf of a user)

S = { bob, jill, jack }

O = { file1, file2, file3 }

R = { read, write, execute, append, own }

	file1	file2	file3
bob	read, write, own	read	read, write
jill	append	read, own	read, write
jack			read, write, own

Example ACL

File1

<bob, {read, write, own}>

<jill, {append}>

- Subjects with no permissions are not listed

ACL with duplicate premissions

It is often the case that a group of users all have the same permissions on an object, e.g.,

File1

<bob, {read, write, own}>

<jill, {read}>

<jack, {read}>

<joan, {read}>

- To save space and administrative effort, a group can be created and the group added to the ACL

ACL with a Group

students = {jill, jack, joan}

File1

<bob, {read, write, own}>

<students, {read}>

Roles

- If students have access to many objects, then the administrator would have to add the *students* group to each object with the associated permissions.
- Easier to define *student* as a role rather than a group.
- A role is a collection of objects and the operations allowed on those objects, e.g.,

  student

  <file1, {read}>,

  <file2, {read, write}>

  <file3, {append}>

  ...

Roles and the OS

- When a student is logged into the system, he/she would be assigned the role of student.
- Any access to an object that doesn't explicitly mention the student (e.g., jill) would require the OS to check the roles assigned to the student to see if the access is allowed.

Main Point

2. A role includes an object and the allowable operations on that object. Spontaneous right action is action that promotes maximum evolution for everybody.

Capabilities

- A capability list represents a row in the ACM
- Each subject is associated with the objects it can access and the permissions it has on these objects.

User Based ACM

(processes run on behalf of a user)

S = { bob, jill, jack }

O = { file1, file2, file3 }

R = { read, write, execute, append, own }

	file1	file2	file3
bob	read, write, own	read	read, write
jill	append	read, own	read, write
jack			read, write, own

Capability Lists

bob
  <file1, {read, write, own}>
  <file2, {read}>
  <file3, {read, write}>
jill
  <file1, {append}>
  <file2, {read, own}>
  <file3, {read , write}>
jack
  <file3, {read, write, own}>

Protecting Capabilities

- Since a process can pass capabilities to child processes, capabilities must be protected from tampering
- There are three ways to prevent unauthorized changes:
  1. Tagged architecture
  2. Storing the capability list inside the kernel
  3. Encrypting the capability list
- Which service component of computer security are we concerned with here?

1. Tagged Architecture

- Each memory location has a tag bit that specifies whether that location contains a capability
- Only the kernel can modify a memory location that contains a capability

2. Keep Capability List in Kernel

Unix does this with file descriptors

```
int fd = open(...);
write(fd, ...);
```

only the OS has access to what the fd points to

3. Encrypt the Capability List

Keep the capability list in user space but encrypt them so users cannot tamper with them.

Such a capability looks like this:

f(Object, Rights, Check)

- f is a cryptographically secure 1-way function
- Check is a long random number generated by the OS and known only to the kernel

Mobile Code

- Capabilities solve the problem of sandboxing mobile code very elegantly
- When a foreign program is started, it is given a capability list containing only the capabilities that the machine owner wants to give it.
- Revoking access to an object is difficult because all capabilities that mention that object have to be found.
  - Solution: create a global object table
    - Each object has an entry in the table
    - Capabilities reference the entry, not the object
  - This is not a problem for an ACL

Propagated Access Control Lists (PACL)

Propagated Access Control Lists

- So far, anybody who can read an object can copy it and give other people access to it
- The original creator of the object cannot control this
- With PACL, the creator of a file has control over who can access it
- With the file, the OS would store the creator and a list of those people who the creator wishes to give access to the file

PACL

- Associated with data independent of the object that it is currently residing in.
- An ACL is just associated with an object.

PACL associated with Ann

```
<subject name='Ann'>
 <access>
  <pacl creator='Ann'>
   <entry subject='Ann' permissions='r'/>
   <entry subject='Betty' permissions='r'/>
   <entry subject='Dorothy' permissions='rw'/>
   <entry subject='Elisabeth' permissions='r'/>
  </pacl>
 </access>
</subject>
```

Ann creates a file named "dates"

So "dates" gets Ann's PACL:

```
<object name='dates'>
 <pacl creator='Ann'>
  <entry subject='Ann' permissions='r'/>
  <entry subject='Betty' permissions='r'/>
  <entry subject='Dorothy' permissions='rw'/>
  <entry subject='Elisabeth' permissions='r'/>
 </pacl>
</object>
```

Betty's PACL

Before reading the file "dates"

```
<subject name='Betty'>
 <access>
  <pacl creator='Betty'>
   <entry subject='Betty' permissions='r'/>
   <entry subject='Cherisse' permissions='r'/>
   <entry subject='Dorothy' permissions='r'/>
  </pacl>
 </access>
</subject>
```

After Betty reads the file "dates" her PACL changes to

```
<subject name='Betty'>
 <access>
  <intersection>
   <pacl creator='Betty'>
    <entry subject='Betty' permissions='r'/>
    <entry subject='Cherisse' permissions='r'/>
    <entry subject='Dorothy' permissions='r'/>
   </pacl>

   <pacl creator='Ann'>  <!-- This is PACL of file dates -->
    <entry subject='Ann' permissions='r'/>
    <entry subject='Betty' permissions='r'/>
    <entry subject='Dorothy' permissions='rw'/>
    <entry subject='Elisabeth' permissions='r'/>
   </pacl>
  </intersection>
 </access>
</subject>
```

When Betty writes the file "datescopy", its PACL becomes:

```
<object name='datescopy'>
 <pacl creator='Betty'>
      <!-- this is intersection of Betty and Ann's PACLs -->
  <entry subject='Betty' permissions='r'/>
  <entry subject='Dorothy' permissions='r'/>
      <!-- Note that Dorothy lost write permission-->
 </pacl>
</object>
```

Types of Access Control (pp 53-54)

- Discretionary access control (DAC):
  - an individual user can set an access control mechanism to allow or deny access to an object.
- Mandatory access control (MAC):
  - a system mechanism controls access to an object and an individual user cannot alter that access.
- Originator controlled access control (ORCON):
  - access based on the creator of an object (or on information it contains)

Main Point

3. There are many ways to implement an ACM. All of relative creation is a manifestation of the field of pure consciousness.

Six consumer technologies that are destroying traditional IT (security)

1. USB flash drives
2. Rogue wireless access points
3. Web mail with GB of storage
4. P2P Software
5. Personal smartphones
6. Instant messaging software

1. USB flash drives

- Can hold large amounts of data
- A user could walk out with an unencrypted copy of a large chunk of a file server
- Possible Solutions:
  - Disable USB mass storage transfer
  - Do not allow files to be saved to local hard drives

2. Rogue wireless access points

- Employees can buy a $100 (or less) wireless access point and plug it into their laptop
  - If near a window that is near a street or parking lot, then anyone driving by has a free Internet connection and access to the corporate network
- Possible Solutions:
  - Require authentication before allowing access to the network
  - Snoop for wireless access points (if necessary, do not allow them as part of security policy)

3. Web mail with GB of storage

- Large company files can be transferred to and stored in Web mail accounts, such as gmail and yahoo, which are much less secure than corporate email
  - Most corporate accounts do not allow large attachments
- Possible Solutions:
  - Do not allow any traffic out of the network to the internet directly from clients - proxy everything

4. P2P Software

- Only has peer nodes
  - No notion of client and server nodes
  - Peer nodes can function simultaneously as both client and server
- Simplifies file sharing
  - Used for transferring large files
- Circumvents the firewall
- No need to do the usual FTP configuration and faster

5. Personal smartphones

- Employees may forward their business email to their smartphones which creates privacy, regulatory, and security issues
- Possible Solutions:
  - Do not allow any traffic out of the network to the internet directly from users, including email
  - Training of employees in the consequences

6. Instant messaging software

- IM file transfers can introduce files that have not been scanned by antivirus programs
- IM software is often installed by the user (against security policy)
- Most users send messages unencrypted so company secrets may be sent out (without realizing the consequences) on the internet for hackers to sniff
- Possible Solutions:
  - Do not allow any traffic out of the network to the internet directly from users
  - Training of employees

Conclusions

- The associated security issues are arising because of new technology and the Principle of Psychological Acceptability
  - i.e., employees usually want to be more productive rather than thwart security policy
  - IT needs to find out how to accommodate this desire to be productive without allowing security to be compromised
  - Some of the "solutions" above may only cause employees to become even more creative in their attempts to get around security restrictions
  - Solution possibility:
    - if employees need to work at home, employers could provide flash drives that automatically encrypt data,
    - Automatically encrypt email so only the intended receiver can read it
    - I.e., make it easier for employees to get the job done efficiently without compromising security

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

1. You can only write to directories for which you have write permission.
2. An access control matrix is a simple way to specify the access control mechanisms.

3. Transcendental Consciousness remains a "secret" if the attention is always directed outwards.
4. Wholeness moving within itself: in Unity Consciousness one has total knowledge of the object, namely it is a manifestation of one's own Self. This is the ultimate state of trust.