Access Control

Information Security

Dr Hans Georg Schaathun

University of Surrey

Autumn 2011 – Week 9

Outline

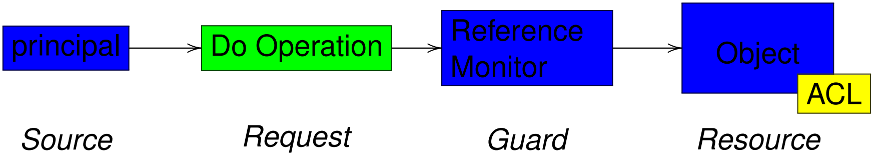
Session objectives

- Introduce fundamental terminology of access control
- Understand principles of privilege management and identity management

Outline

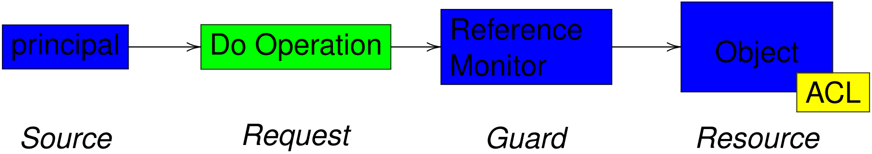
Outline

The request



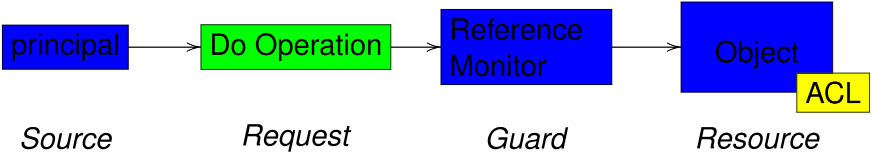
- Authentication
- Authorisation

The request



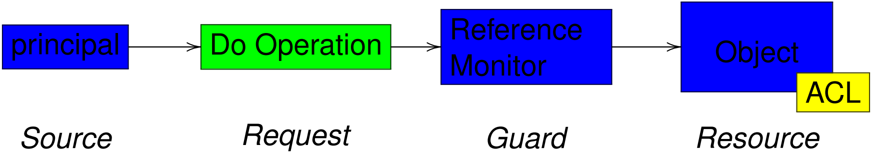
- Authentication
- Authorisation

The request



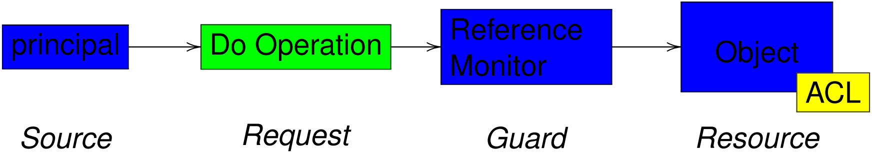
- Authentication
  - Who made the request R?
- Authorisation

The request



- Authentication
  - Who made the request *R*?
- Authorisation

The request



- Authentication
  - Who made the request R?
- Authorisation
  - Who is trusted to access an object o?
  - Who is trusted to have request R granted?

Subjects and objects

A subject is an active entitity within an IT system

- e.g. user, process

An object is a resource that (some) subject may access or use.

- e.g. files, printers, memory

A principal is an entity that can be granted access to objects or can make statements affecting access control decissions.

- distinction subject/principal is not always necessary
- a subject (process) may act on behalf of a subject (user)

What is an object?

- A file — very traditional view (read/write/execute)
- A system — access or no access
- An operation — i.e. an action to take
- A room — access or no access

Authentication and Authorisation

- Authentication
  - Determine identity.
- Authorisation
  - Determine privileges.
- This allows identity based access control.
- Could you do authorisation without authentication?

Authentication and Authorisation

- Authentication
  - Determine identity.
- Authorisation
  - Determine privileges.
- This allows identity based access control.
- Could you do authorisation without authentication?

Authentication and Authorisation

- Authentication
  - Determine identity.
- Authorisation
  - Determine privileges.
- This allows identity based access control.
- Could you do authorisation without authentication?

Outline

Four subproblems

- Identification and Authentication
  - establishing the identity of a subject
- Identity management
  - managing identities and credentials
  - essential data for authentication
- Authorisation
  - granting privileges to an identified subject
- Privilege Management
  - managing mapping of subject to privileges
  - necessary data for authorisation

Problem Domain

Access controll is a general problem ...

- Operating System
- File System
- Web Site
- Locked Doors
- Paper Archive Records
- Database Records
- Documents (PDF, etc.)

Outline

Outline

Access modes

Observe i.e. *read*

- Limited by confidentiality

Alter i.e. *append*

- Limited to ensure integrity

Execute (running a program)

- Can you execute without reading?
  - Sometimes; it may be sufficient that the OS reads it.

- *write* = *read* + *append* (Bell-LaPadula)

Access modes

Observe i.e. *read*

- Limited by confidentiality

Alter i.e. *append*

- Limited to ensure integrity

Execute (running a program)

- Can you execute without reading?
  - Sometimes; it may be sufficient that the OS reads it.

- *write* = *read* + *append* (Bell-LaPadula)

Access modes

Observe i.e. *read*

- Limited by confidentiality

Alter i.e. *append*

- Limited to ensure integrity

Execute (running a program)

- Can you execute without reading?
  - Sometimes; it may be sufficient that the OS reads it.

- *write* = *read* + *append* (Bell-LaPadula)

Discretionary or Mandatory

Discretionary Access Control The owner of each resource determines access permissions.

Mandatory Access Control A central authority defines a security policy defining access rights

- This is 4th Design Decision from Gollmann (Ch 2).
  - Centralised or local security control?

Access Control Structures

- Access Control Matrix: $[A_{s,o}]$
  - $A_{s,o}$ is the permissions of Subject s to Object o.
  - $A_{s,o} \subset \{\text{alter}, \text{observe}\}$
- Subject-wise capabilities
  - For each Subject s, maintain a list of rights.
- Access Control List: object-wise
  - For each Object o, maintain a list of access permissions.
  - suitable for discretionary access control
- Access data takes a lot of space.
- Coarser access control is more common.

Access Control Structures

- Access Control Matrix: $[A_{s,o}]$
  - $A_{s,o}$ is the permissions of Subject s to Object o.
  - $A_{s,o} \subset \{\text{alter}, \text{observe}\}$
- Subject-wise capabilities
  - For each Subject s, maintain a list of rights.
- Access Control List: object-wise
  - For each Object o, maintain a list of access permissions.
  - suitable for discretionary access control
- Access data takes a lot of space.
- Coarser access control is more common.

Access Control Structures

- Access Control Matrix: $[A_{s,o}]$
  - $A_{s,o}$ is the permissions of Subject s to Object o.
  - $A_{s,o} \subset \{\text{alter}, \text{observe}\}$
- Subject-wise capabilities
  - For each Subject s, maintain a list of rights.
- Access Control List: object-wise
  - For each Object o, maintain a list of access permissions.
  - suitable for discretionary access control
- Access data takes a lot of space.
- Coarser access control is more common.

Access Control Structures

- Access Control Matrix: $[A_{s,o}]$
  - $A_{s,o}$ is the permissions of Subject s to Object o.
  - $A_{s,o} \subset \{\text{alter}, \text{observe}\}$
- Subject-wise capabilities
  - For each Subject s, maintain a list of rights.
- Access Control List: object-wise
  - For each Object o, maintain a list of access permissions.
  - suitable for discretionary access control
- Access data takes a lot of space.
- Coarser access control is more common.

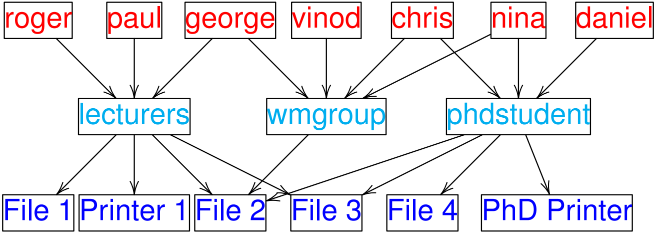
Access Control Structures

- Access Control Matrix: $[A_{s,o}]$
  - $A_{s,o}$ is the permissions of Subject s to Object o.
  - $A_{s,o} \subset \{\text{alter}, \text{observe}\}$
- Subject-wise capabilities
  - For each Subject s, maintain a list of rights.
- Access Control List: object-wise
  - For each Object o, maintain a list of access permissions.
  - suitable for discretionary access control
- Access data takes a lot of space.
- Coarser access control is more common.

Outline

Group-based access control

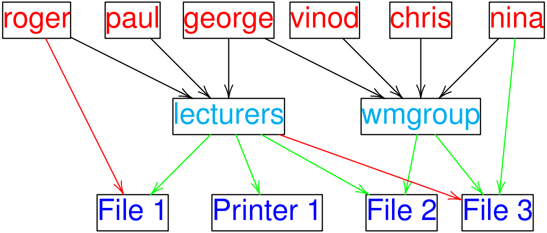
- Access can be organised in groups.



- Save the effort of considering access for individual users.

Policy Conflicts

- Permission can be positive or negative
  - Access denied or Access permitted or



- How do you resolve conflicts?
  - *roger* has red and green path to `File 1`.
  - *george* has red and green path to `File 3`.

A general rule

- Any security policy has to define precedence.
- How do you resolve conflicting policy rules?

Example

- User rights takes precedence over group rights.
- Negative group rights takes precedence over positive group rights.
  - Or the other way around...

Abstract Data Types and Procedures

Datatype (or class)

- access restricted to certain methods
  - general programming practice
  - prevents some errors
  - allows access control
  - distinguishes between public and private

Procedure is a method accessing a datatype.

- More fine-grained than alter and observe
- An ADT can only be accessed via well-defined procedures
- Use of each procedure can be restricted

Roles

- A role is a collection of procedures
- A user can hold several roles
- Many user can hold the same role
- Roles typically map the organisation structure
    - Research Assistant on Watermarking
    - Research Assistant on Artificial Intelligence
    - Team leader on Watermarking
    - Team leader on Artificial Intelligence

Role-Based Access Control (RBAC)

- Hierarchical:
    - The team leader may appoint research assistants
    - The lecturer may appoint (enroll) students
- Hierarchical means semi-centralised
    - Policy can be made at every level.
    - The central chief can make organisation-wide policies.
    - Team leaders can define mandatory access control for small teams.
- RBAC is common in database management systems

Security levels

- Classic classification
- Linear ordering of security levels
- Sounds rather military...

Top Secret

$\cap$

Secret

$\cap$

Confidential

$\cap$

Unclassified

Protection Rings

- Security level are used in hardware
  - called Protection Rings
- E.g. for Intel 80x86
  1. Operating system kernel
  2. Operating system
  3. Utilities
  4. User processes
- Protection rings had to be implemented to run Multics.
- Unix uses only ring 0 (root) and ring 3 (user).

Hardware Security Policy

- The following Security Policy is implemented:
    - Procedures can only access objects in their own ring and outer rings.
    - Procedures can invoke subroutines in their own ring only.
- Question for you:
    - Why is a procedure not allowed to invoke subroutines in an outer ring?

- Cf. Bell-LaPadula model

Hardware Security Policy

- The following Security Policy is implemented:
  - Procedures can only access objects in their own ring and outer rings.
  - Procedures can invoke subroutines in their own ring only.
- Question for you:
  - Why is a procedure not allowed to invoke subroutines in an outer ring?

- Cf. Bell-LaPadula model

Hardware Security Policy

- The following Security Policy is implemented:
  - Procedures can only access objects in their own ring and outer rings.
  - Procedures can invoke subroutines in their own ring only.
- Question for you:
  - Why is a procedure not allowed to invoke subroutines in an outer ring?

- Cf. Bell-LaPadula model

Hardware Security Policy

- The following Security Policy is implemented:
  - Procedures can only access objects in their own ring and outer rings.
  - Procedures can invoke subroutines in their own ring only.
- Question for you:
  - Why is a procedure not allowed to invoke subroutines in an outer ring?
- Subroutines in outer rings can be modified by procedures in outer rings.
- If such a modified subroutine were invoked in an inner ring, it would run with more privileges.
- The modifying procedure could then make code to be executed with privileges it should not have.
- Cf. Bell-LaPadula model

Hardware Security Policy

- The following Security Policy is implemented:
  - Procedures can only access objects in their own ring and outer rings.
  - Procedures can invoke subroutines in their own ring only.
- Question for you:
  - Why is a procedure not allowed to invoke subroutines in an outer ring?
- Subroutines in outer rings can be modified by procedures in outer rings.
- If such a modified subroutine were invoked in an inner ring, it would run with more privileges.
- The modifying procedure could then make code to be executed with privileges it should not have.
- Cf. Bell-LaPadula model

Hardware Security Policy

- The following Security Policy is implemented:
  - Procedures can only access objects in their own ring and outer rings.
  - Procedures can invoke subroutines in their own ring only.
- Question for you:
  - Why is a procedure not allowed to invoke subroutines in an outer ring?
- Subroutines in outer rings can be modified by procedures in outer rings.
- If such a modified subroutine were invoked in an inner ring, it would run with more privileges.
- The modifying procedure could then make code to be executed with privileges it should not have.
- Cf. Bell-LaPadula model

Hardware Security Policy

- The following Security Policy is implemented:
  - Procedures can only access objects in their own ring and outer rings.
  - Procedures can invoke subroutines in their own ring only.
- Question for you:
  - Why is a procedure not allowed to invoke subroutines in an outer ring?
- Subroutines in outer rings can be modified by procedures in outer rings.
- If such a modified subroutine were invoked in an inner ring, it would run with more privileges.
- The modifying procedure could then make code to be executed with privileges it should not have.
- Cf. Bell-LaPadula model

Discussion Exercise

- [Gollmann 4.3] Discuss: What are the differences between groups and roles, if there are any differences at all?

Multilevel Security

- One set of classifications with a linear (hierarchical) ordering $\leq_H$
  - public $\leq_H$ confidential $\leq_H$ secret
- One set of categories
  - E.g. $\{\mathrm{EE}, \mathrm{Comp}, \mathrm{Math}\}$
- A Compartment is a set of categories
  - Subset ordering $\subset$
- A security level is a pair (category,classification)
  - $(h_1, c_1) \leq (h_2, c_2) \Leftrightarrow (h_1 \leq_H h_2 \wedge c_1 \subset c_2)$
- Access is granted if $(h_{\text{object}}, c_{\text{object}}) \leq (h_{\text{subject}}, c_{\text{subject}})$
  - We say that $(h_{\text{subject}}, c_{\text{subject}})$ dominates $(h_{\text{object}}, c_{\text{object}})$

Need-to-know policy

- Multilevel security can
  - restrict access to members of a project or department
  - while maintaining mandatory access control
- Computing staff with highest clearing (secret,{comp})
  - has no rights to objects from EE or Maths

$$(\text{public}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}\}) \quad (1)$$

$$(\text{secret}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}, \text{EE}\}) \quad (2)$$

$$(\text{public}, \{\text{EE}\}) \not\leq (\text{secret}, \{\text{comp}\}) \quad (3)$$

- Staff do not need to know about other departments
- No need ⇒ No access

Need-to-know policy

- Multilevel security can
  - restrict access to members of a project or department
  - while maintaining mandatory access control
- Computing staff with highest clearing (secret,{comp})
  - has no rights to objects from EE or Maths

$$(\text{public}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}\}) \quad (1)$$

$$(\text{secret}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}, \text{EE}\}) \quad (2)$$

$$(\text{public}, \{\text{EE}\}) \not\leq (\text{secret}, \{\text{comp}\}) \quad (3)$$

- Staff do not need to know about other departments
- No need ⇒ No access

Need-to-know policy

- Multilevel security can
  - restrict access to members of a project or department
  - while maintaining mandatory access control
- Computing staff with highest clearing (secret,{comp})
  - has no rights to objects from EE or Maths

$$(\text{public}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}\}) \tag{1}$$

$$(\text{secret}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}, \text{EE}\}) \tag{2}$$

$$(\text{public}, \{\text{EE}\}) \not\leq (\text{secret}, \{\text{comp}\}) \tag{3}$$

- Staff do not need to know about other departments
- No need ⇒ No access

Need-to-know policy

- Multilevel security can
  - restrict access to members of a project or department
  - while maintaining mandatory access control
- Computing staff with highest clearing (secret,{comp})
  - has no rights to objects from EE or Maths

$$(\text{public}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}\}) \tag{1}$$

$$(\text{secret}, \{\text{comp}\}) \leq (\text{secret}, \{\text{comp}, \text{EE}\}) \tag{2}$$

$$(\text{public}, \{\text{EE}\}) \not\leq (\text{secret}, \{\text{comp}\}) \tag{3}$$

- Staff do not need to know about other departments
- No need ⇒ No access

Outline

What is Identity Management?

- Someone, somewhere needs to store
  - identity (personal information)
  - credentials (to allow authentication)
    - e.g. picture, password, biometric data, etc.

The user problem

How can you manage all your credentials?

- One user name per service
- One password per service
- One smart card per service

The user problem

How can you manage all your credentials?

- One user name per service
- One password per service
- One smart card per service

The server problem

How do you establish identities the first time?
How do you collect credentials?

- Boot-strap problem
  - initial identification and authentication to create account
- Storage of identity information
  - Security problems and the lot

Third-Party Identity Management

- Identity Management external to Access Control
- Service Provider prompts an Identity Server
  - authorisation based on identification
  - but identification is completely out-sourced
- The Identity Server does
  - identification and authorisation
  - issues a certificate of identity for the access control mechanism
- For example: OpenID

The Identity Server

- Same credentials for many services
- Configurability
  - personal information managed on a per service basis
- For example, commenting on `http://www.bt.no`
  - identitification required
  - different identity providers accepted
  - facebook, OpenID, etc.

Client-Side Identity Information

Could the user store all his identity information and credentials?

- Smart-Cards or small hardware devices
  - storage for identity
  - trusted device for the service provider
- The device issues a certificate
  - public key cryptography

Open access web sites

Why do you require identification for open access (free of charge) web sites?

Outline

Methods of identification

- Something you know (password)
- Something you carry (smartcard)
- Something you are (fingerprint)
- Something you do (signature)

Outline

Criteria

Universal	everybody has it
Particular	one-to-one mapping for individual
Lasting	not subject to change
Important	natural characteristic of individual
Readable	anyone can read it
Storable	we can store it
Sufficient	no need for other identifiers
Precise	significant difference between individuals
Simple	reliable identification – few errors
Cheap	cost-efficient for the task
Convenient	no nuisance to the user
Acceptable	to society and most individuals

Storing biometric data

Storage of biometric data is a privacy concern

- Different options
  - complete data to reproduce the biometric object
  - hashed storage, allowing validation and not reproduction
  - smart-card storage — only the user has access

Outline

Conclusion

- Two separate management problems
  - Privilege Management
  - Identity Management
- Must be handled separately
- Two operational problems
  - Identification and Authentication
  - Authorisation
- May or may not be handled separately