CS361: Introduction to Computer Security

Policies and Channels II

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin

Last updated: February 26, 2020 at 11:31

Meaning of Computer Security

Typically today, computer security is described as encompassing at least:

Confidentiality: (also called secrecy/privacy) who can *read* information;

Integrity: who can *write* or modify information;

Availability: what mechanisms ensure that resourses are available when needed.

Confidentility Ain't Enough

Confidentiality models, like BLP, are useful but obviously limited, even with covert channel checking.

A private overwriting a general's documents does not compromise confidentiality, but obviously violates integrity.

How then do we extend our models to handle integrity concerns?

Commercial Security

Often commercial security controls are discretionary, procedural, and decentralized, rather than mandatory and centralized.

Integrity rather than confidentiality may be of primary concern.

What is Integrity?

Integrity is a fuzzier notion than confidentiality and more context dependent.

A program can damage integrity without interaction with the external world, simply by computing data incorrectly. Threats to integrity may be accidental or malicious.

- Who is authorized to provide or modify information?
- How do you separate and protect assets?
- How do you detect and/or correct erroneous or unauthorized changes to data?

Some Integrity Principles

Integrity has aspects and principles of operation not as relevant to military security:

Separation of Duty: several *different* people must be involved to complete a critical function.

Separation of Function: a single person cannot complete complementary roles within a critical process.

Auditing: recoverability and accountability require maintaining an audit trail.

Give some examples of each of these. Explain the difference between separation of duty and separation of function.

Commercial Concerns

For example, Steve Lipner lists five concerns you might find in commercial data integrity:

1. Users will not write their own programs, but use existing production software.
2. Programmers develop and test applications on a nonproduction system, using dummy data.
3. Moving applications from development to production requires a special process.
4. This process must be controlled and audited.
5. Managers and auditors must have access to system state and system logs.

Integrity Meta-Policy

Recall the BLP confidentiality *meta-policy*: information should only flow upward in the lattice of security levels.

What is the meta-policy that integrity models are trying to enforce? What is integrity really about?

What Is Integrity?

Suppose you're standing in a checkout line at the grocery store and on the adjacent newsrack you notice the headline: "Hillary Clinton Adopts Alien Baby." Do you believe it?

What Is Integrity?

Suppose you're standing in a checkout line at the grocery store and on the adjacent newsrack you notice the headline: "Hillary Clinton Adopts Alien Baby." Do you believe it?

Your reaction might differ depending on whether the publication is:

1. *The New York Times*: Wow! Could there be something to this?
2. *The Wall Street Journal:* The vast right wing conspiracy is after poor Hillary again!
3. *The National Enquirer:* Those idiots! How funny!

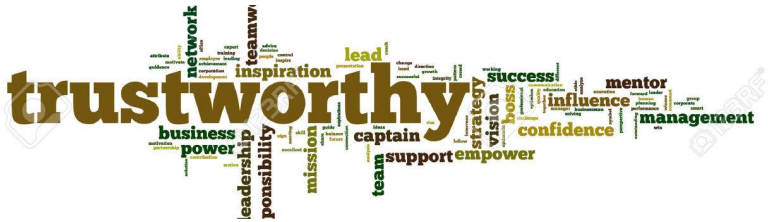
On the Other Hand!

Integrity Levels: Object

Many integrity models presume that we can associate *integrity levels* with subjects and with objects in our system, and define a `dominates` relation between levels.

The integrity level of *an object* describes the degree of "trustworthiness" of the information contained in that object. For example, a "man-on-the-street" report may have lower integrity than a report from a panel of experts.

Integrity Levels: Subject



The integrity level of *a subject* is a measure of the confidence one places in the ability of that subject to produce / handle information. For example, a certified application may have more integrity than freeware downloaded from the Internet.

Structured Levels

Suppose we follow our standard scheme for labels. As usual, we have a hierarchical component and a set of categories.

Then what would labels like the following mean?

(High: {Nuclear}),
(Low: {Crypto}).

Does this scheme even make sense for integrity?

Important Proviso

Note that integrity levels are *not also clearance levels*, though in many cases, they have similar structure and `dominates` is defined exactly analogously.

Integrity concerns are orthogonal to confidentiality concerns and should be treated either separately or in a "mixed policy." *In a system with both confidentiality and integrity constraints, only accesses that pass both tests may be allowed.*

Important Proviso

A general may have read access to very confidential information, but be a very unreliable source of intelligence.

A piece of information may be both highly unreliable and very sensitive, or highly reliable and of little sensitivity.

Biba's Integrity Models

Ken Biba proposed three different models of integrity control. All assume that we can associate *integrity levels* with subjects and objects, analogous to clearance levels in BLP.

1. Low Water Mark Integrity Policy
2. Ring Policy
3. Strict Integrity

Only Strict Integrity had much continuing influence. It is the one typically referred to as "Biba Integrity."

Biba's Low Water Mark Integrity Policy

In general, a *water mark* policy is one where an attribute monotonically floats up (high water mark) or down (low water mark), but may be "reset" at some point.

Biba's Low Water Mark Policy has the following two rules:

1. Subject s can write to object o only if $i(o) \leq i(s)$.
2. If s reads o, then $i'(s) = \min(i(s), i(o))$, where $i'(s)$ is the subject's new integrity level after the read.

What is the underlying assumption/rationale about subjects in this policy? Are they considered at all trustworthy?

Low Water Mark (Cont.)

A potential of the LWM Integrity policy is to monotonically decrease the integrity level of a subject unnecessarily.

This sort of problem is called *label creep* and may result in an overly conservative analysis.

What would happen if you decrease *object* integrity levels instead of subject integrity levels?

Ring Policy

This focuses on direct modification and solves some problems of LWM.

1. Any subject can read any object, regardless of integrity levels.
2. Subject s can write to object o only if $i(o) \leq i(s)$.

Does the Ring policy make some assumption about the subject that the LWM policy does not?

Strict Integrity Policy

This policy is what is typically called the "Biba Model."

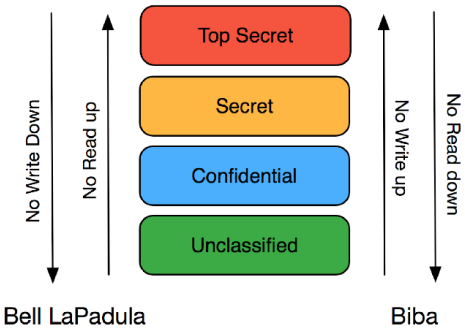
1. **Simple integrity:** Subject s can read object o only if $i(s) \leq i(o)$.
2. **Integrity *-property:** Subject s can write to object o only if $i(o) \leq i(s)$.

This is the *dual* of BLP. What does that mean?

Note that it would be possible to have a system that incorporated both BLP and Strict Integrity, using very similar mechanisms to enforce these two orthogonal aspects of security. *But you need two sets of labels.*

Relation of Biba Models to BLP

Notice that Biba's Strict Integrity Policy is exactly analogous to the Bell and LaPadula model.



What would *confidentiality* policies look like that directly corresponded to Biba's Low Water Mark and Ring Policies?

Would these be reasonable access control policies from the perspective of constraining information flow?

Revisiting Lipner's Concerns

Recall Lipner's integrity concerns for a commercial setting:

1. Users will not write their own programs, but use existing production software.
2. Programmers develop and test applications on a nonproduction system, possibly using contrived data.
3. Moving applications from development to production requires a special process.
4. This process must be controlled and audited.
5. Managers and auditors must have access to system state and system logs.

Lipner's Integrity Matrix Model

Lipner devised his Integrity Matrix Model to handle those concerns via a combination of BLP and Biba Integrity. Determine whether he succeeded.

There are two confidentiality levels:

Audit Manager (AM): system audit and management.

System Low (SL): all other processes.

In addition there are three confidentiality categories:

Production (SP): production code and data.

Development (SD): programs under development.

System Development (SSD): system programs in development.

Lipner's Model (Cont.)

In addition to the confidentiality constraints, we also impose integrity constraints. There are three integrity classification (highest to lowest):

System Program (ISP): system software

Operational (IO): production programs and development software

System Low (ISL): user level behavior

and two integrity categories:

Development (ID)

Production (IP)

Subject Levels

Security levels (both confidentiality and integrity) are assigned to subjects based on their roles in the organization and their need to know.

User Role	Confidentiality	Integrity
Ordinary users	$(SL, \{SP\})$	$(ISL, \{IP\})$
Application developers	$(SL, \{SD\})$	$\{ISL, \{ID\})$
System programmers	$(SL, \{SSD\})$	$\{ISL, \{ID\})$
System managers/auditors	$(AM, \{SP, SD, SSD\})$	$\{ISL, \{IP, ID\})$
System controllers	$(SL, \{SP, SD\})$ and downgrade	$\{ISP, \{IP, ID\})$

Here downgrade means the ability to move from development to production. What is the tranquility property for this system?

Object Levels

Security levels (both confidentiality and interity) are assigned to objects based on who should access them.

Object type	Confidentiality	Integrity
Development code/test data	$(SL, \{SD\})$	$\{ISL, \{ID\})$
Production code	$(SL, \{SP\})$	$\{IO, \{IP\})$
Production data	$(SL, \{SP\})$	$\{ISL, \{IP\})$
Software tools	$(SL, \emptyset)$	$\{IO, \{ID\})$
System programs	$(SL, \emptyset)$	$\{ISP, \{IP, ID\})$
System programs in modification	$(SL, \{SSD\})$	$\{ISL, \{ID\})$
System and application logs	$(AM, \{categories\})$	$\{ISL, \emptyset)$

Make sure you review these and convince yourself that these are reasonable decisions. Build an access control matrix to illustrate who can read / write what in this system.

Lipner's Model

Consider the following questions:

1. Can an ordinary user utilize a system program? Can he modify it?
2. Can a system programmer use production software? Can he modify it?
3. Why is that special downgrade permission required? Could it be done with BLP and Biba alone?

Lipner's Model

Consider the following questions:

1. Can an ordinary user utilize a system program? Can he modify it?
2. Can a system programmer use production software? Can he modify it?
3. Why is that special downgrade permission required? Could it be done with BLP and Biba alone?

The answers:

1. That depends on what "utilize" means. If "utilize" means "read" then he can read, but not modify.
2. Neither.
3. Moving objects from the development to production world means changing their types. There's no way to do that in BLP or Biba.

Integrity Meta-Policy Revisited

Earlier we asked: What is the meta-policy that integrity models are trying to enforce?

Integrity Meta-Policy Revisited

Earlier we asked: What is the meta-policy that integrity models are trying to enforce?

Possible answer: don't allow unreliable information to corrupt more reliable information.

Low integrity information shouldn't flow into high integrity domains.

Clark-Wilson Commercial Security

Biba integrity is a clone of BLP. Clark and Wilson proposed a model that they argued better reflects the integrity requirements of real commercial enterprises, particularly separation of duty.

A major integrity concern is *consistency* among the various components of the system state. For example, for a bank, the funds at the beginning of the day plus the funds deposited minus the funds withdrawn should equal funds on hand.

A *well-formed transaction* is a procedure that moves from one consistent state to another.

Four Basic Criteria

1. **Authentication:** identity of all users must be properly authenticated.
2. **Audit:** modifications should be logged to record every program executed and by whom, in a way that cannot be undone.
3. **Well-formed transactions:** users manipulate data only in constrained ways. Only legitimate accesses are allowed.
4. **Separation of duty:** the system associates with each user a valid set of programs they can run. Prevents unauthorized modifications, thus preserving integrity and consistency with the real world.

Basic Concepts

- *Constrained Data Items:* CDIs are the objects whose integrity is protected
- *Unconstrained Data Items:* UDIs are objects not covered by the integrity policy
- *Integrity Verification Procedures:* IVPs are procedures meant to verify maintainance of integrity of CDIs.
- *Transformation Procedures:* TPs are the only procedures allowed to modify CDIs, or take arbitrary user input and create new CDIs. Designed to take the system from one valid state to another.

Clark-Wilson (Cont.)

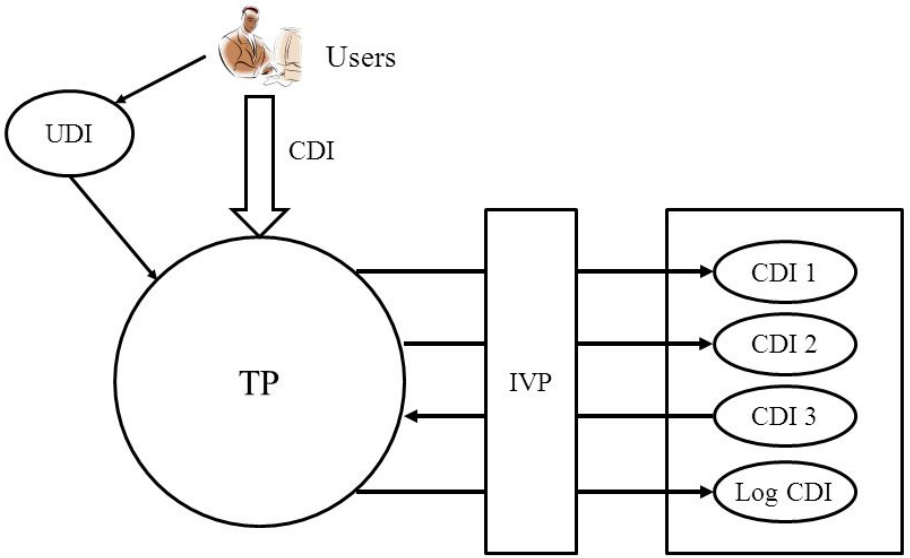
The model is defined in terms of a set of triples of the form:

$$(user, TP, \{CDI\ set\})$$

where *user* is authorized to perform a *transaction procedure* TP, on a given set of *constrained data items* (CDIs).

Some rules govern the application of transformations, the certification of the state, and the enforcement of the constraints.

Basic Concepts

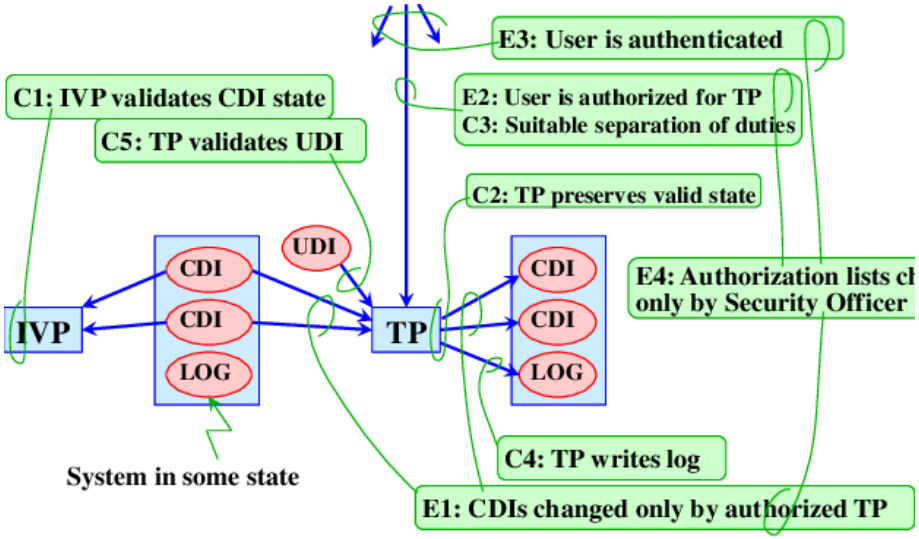


Integrity Rules

There are two kinds of rules—Certification and Enforcement.

- C1: All IVPs must ensure that CDIs are in a valid state when the IVP is run.
- C2: All TPs must be certified valid.
- C3: Assignment of TPs to users must satisfy separation of duty.
- C4: The operation of TPs must be logged.
- C5: TPs executing on UDIs must result in valid CDIs.
- E1: Only certified TPs can manipulate CDIs.
- E2: Users must only access CDIs by means of TPs for which they are authorized.
- E3: The identify of each user attempting to execute a TP must be authenticated.
- E4: Only the agent permitted to certify entities can change the list of entities associated with other entities.

Clark Wilson Rules



Chinese Wall Policy

Brewer and Nash proposed a policy called the **Chinese Wall Policy** that addresses a very specific commercial need: the potential for conflicts of interest and inadvertant disclosure of information.

Strictly speaking, this is not an integrity policy, but an access control confidentiality policy.

Suppose a lawyer specializes in product liability. At one time, she might consult for and see sensitive corporate data from American Airlines or United Airlines, but not both. However, a simultaneous contract with McDonalds would not be a conflict.

Levels of Abstraction

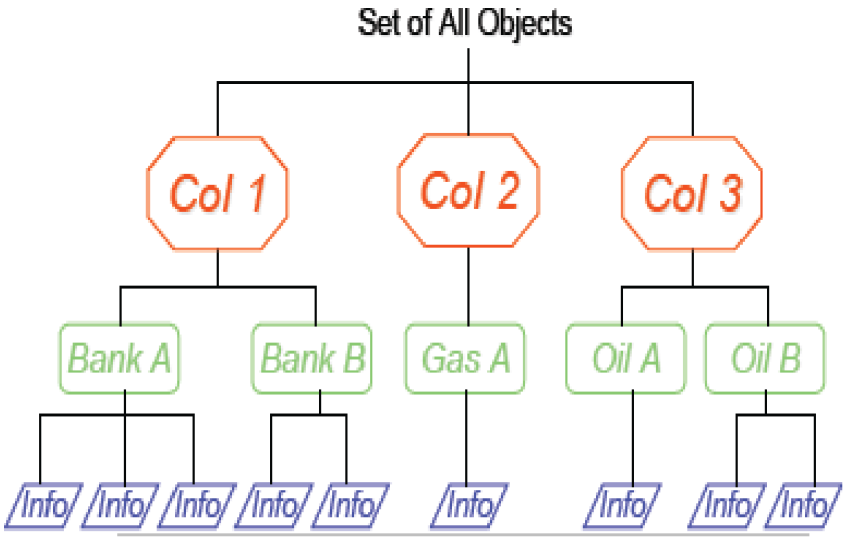
The security policy builds on three levels of abstraction.

- Objects such as files. Objects contain information about only one company.
- Company groups collect all objects concerning a particular company.
- Conflict classes cluster the groups of objects for competing companies.

For example, consider the following conflict classes:

- { Ford, Chrysler, GM }
- { Citicorp, Credit Lyonnais, Deutsche Bank }
- { Microsoft }

Chinese Wall



Chinese Wall Policy

We have a simple access control policy: A person can access information from any company as long as that person has never accessed information from a different company in the same conflict class.

For example, if you access a file from GM, you will subsequently be blocked from accessing any files from Ford or Chrysler. You are free to access files from companies in any other conflict class.

Notice that permissions change dynamically. The access rights that any subject enjoys *depends on the history of past accesses*.

Chinese Wall

The policy restricts access according to the following two properties:

(Chinese Wall) Simple Security Rule: A subject s can be granted access to an object o only if the object:

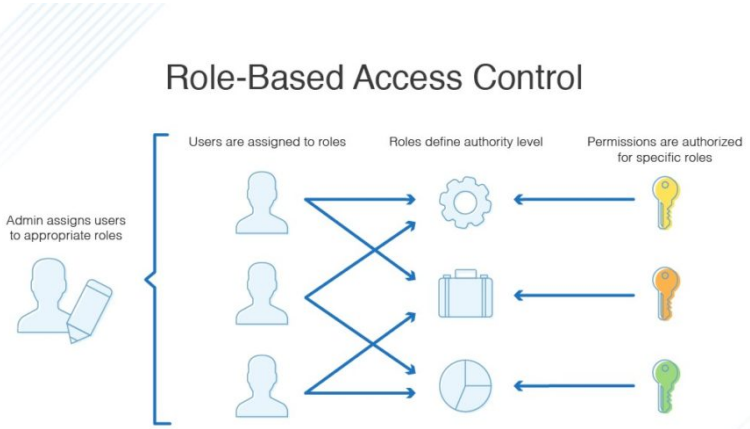
- is in the same company datasets as the objects already accessed by s, that is, "within the Wall," or
- belongs to an entirely different conflict of interest class.

(Chinese Wall) *-property: Write access is only permitted if:

- access is permitted by the simple security rule, and
- no object can be read which is:
  - in a different company dataset than the one for which write access is requested, and
  - contains unsanitized information.

Role-Based Access Control

A *role-based access control* (RBAC) system is another way of enforcing access. It is based on the job functions that a subject performs. A subject may fill several roles at once. Roles may change over time.



Role-Based Access Control

A *role* is a collection of job functions. Each role r is has an associated set of authorized transactions, $trans(r)$.

The *active roles* of a subject s, $actr(s)$, is the set of roles the subject *currently* assumes. The *authorized roles*, $authr(s)$, is the set of roles the subject *may* assume.

The predicate $canexec(s, t)$ is true if subject s can execute transaction t at the current time.

RBAC (Cont.)

Several rules control what subjects can perform which accesses:

Rule of Role Assignment: Let S be the set of subjects and T the set of transactions.

$$\forall s \in S, \forall t \in T, [canexec\,(s, t) \rightarrow actr\,(s) \neq \emptyset]$$

Rule of Role Authorization: Let S be the set of subjects.

$$\forall s \in S, [actr\,(s) \subseteq authr\,(s)]$$

Rule of Transaction Authorization: Let S be the set of subjects and T the set of transactions.

$$\forall s \in S, \forall t \in T, [canexec\,(s, t) \rightarrow t \in trans\,(actr\,(s))]$$

RBAC (Cont.)

Various other notions can be captured in an RBAC system. For example, some roles may *subsume* others, meaning that anyone having role r_j can do at least the functions of r_i. E.g., a trainer can perform all of the actions of a trainee, as well as some others.

We say that role r_j subsumes or contains role r_i ($r_j > r_i$), if:

$$\forall s \in S, [(r_j \in authr(s) \wedge r_j > r_i) \rightarrow r_i \in authr(s)]$$

RBAC can also model *separation of function* (one individual cannot assume both roles r_1 and r_2). This is expressed as:

$$\forall s \in S, [r_1 \in authr(s) \rightarrow r_2 \notin authr(s)]$$

RBAC Advantages

RBAC associates access permissions with a job/function/role rather than with an individual or subject.

RBAC recognizes that a subject can have various functions within the organization.

RBAC allows the subject to transition between roles without having to change identities.

Access Control Matrix

The point of many security models—including Bell and LaPadula, Biba Integrity, and others—is to control the access of subjects to objects according to some criteria. The most general representation of this is an **access control matrix** (ACM).

	$\textbf{object}_1$	...	$\textbf{object}_k$
$\textbf{subject}_1$	A_i, A_j		$\emptyset$
...			
$\textbf{subject}_n$	A_l		A_i, A_m

The ACM gives an explicit representation of every access permitted by every subject to every object.

Policies as ACMs

Any access control policy can be defined using an ACM rather than a set of rules.

- What would such a model look like for a Bell and LaPadula system?
- What would one look like for Biba's Strict Integrity Model?
- How might you combine the two?

Problems with ACMs

Though the ACM is a completely general model of any access control policy, it is difficult and expensive to store a large and sparse matrix of this sort. In a dynamic system, in which subjects and objects come and go, it requires the ability to add and remove rows and columns.

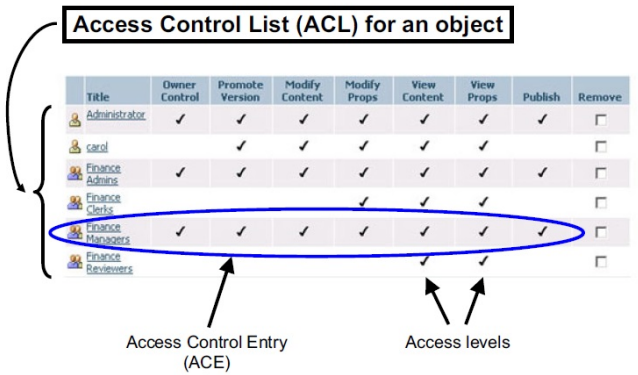
Three common alternatives exists:

1. Maintain a set of rules to compute access permissions based on attributes of subjects and objects. (Think BLP and Biba Integrity.)
2. Associate the permissions with objects. This is called an *access control list* (ACL).
3. Associate the permissions with subjects. This is called a *capability-based system*.

Access Control Lists

An *access control list* (ACL) is a set of permissions that is stored with each object in the system. This represents essentially a column in the ACM.

Each component of the ACL is a pair, $< S, P >$, that lists the set P of permissions that subject S currently holds to the object.



Access Control Lists

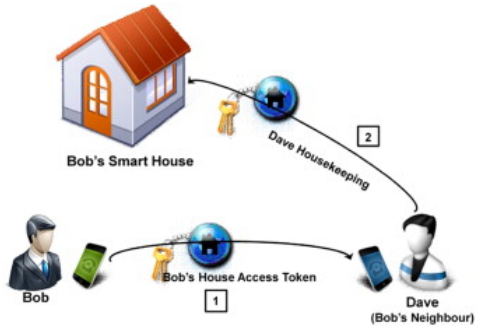
Both Unix/Linux and Windows use ACLs to store access permissions.

Any request by subject S for access A to object O, means checking whether $A \in P$ for the pair $< S, P >$ on O's access control list.

```
-rw-r----- 1 byoung prof  988902 Feb 19 08:18 slides3-integrity-4up.pdf
-rw-r----- 1 byoung prof  970224 Feb 19 08:18 slides3-integrity.pdf
-rw-r----- 1 byoung prof   43344 Feb 19 08:18 slides3-integrity.tex
-rw-r----- 1 byoung prof     481 Feb 19 08:18 slides3-integrity.vrb
```


Capabilities

Some systems store permissions with subjects rather than objects. These are called *capabilities*, and a set of capabilities associated with a subject represents a row in the ACM.



Capabilities

Each subject S maintains a set C of pairs $< O, A >$, meaning that S has current permission to perform access A to object O. To obtain access, the subject must present an appropriate capability. Thus a capability is a type of "ticket."

To maintain security, it is necessary to ensure that subjects cannot manufacture or modify capabilities. That is, they must be *unforgeable* and *unalterable*. There may also be permissions to pass capabilities from one subject to another.

Access Control Matrix

	$\textbf{object}_1$	...	$\textbf{object}_k$
$\textbf{subject}_1$	A_i, A_j		$\emptyset$
...			
$\textbf{subject}_n$	A_l		A_i, A_m

- An ACL effectively stores a column of the ACM with the subject.
- A capability-based system stores a row of the ACM with the object.

What types of security questions would be easy to answer in each approach?