CS361: Introduction to Computer Security

Policies and Channels

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

Last updated: January 29, 2020 at 13:48

How Do We Define Security?

We said that in the most general terms, *security* seems to mean something like "protection of assets against attack."

But this question *is very specific to the context*. Security for a wireless phone system may be very different from security for a military database system or an on-line banking system.

Policies

Often, security for a given system is defined in terms of a *security policy*, also sometimes called a *security model*.

The policy is the system *specification* wrt security. It's a *contract* between the designer/implementor and the customer. Must be both achievable and adequate for the intended uses.

Policies

The policy defines what "security" means for a given system or family of systems.

A policy may be characterized informally, semi-formally, or formally.

It may be very abstract or very concrete.

Thought Experiment #1

Your academic records are stored on computers at the university. Design a security policy to protect them.

Thought Experiment #1

Start by asking: What does it mean "to protect them"?

- What are you protecting and what are the potential threats?
- Who are the stakeholders, i.e., whose interests are at risk?
- Do the interests of various stakeholders conflict?
- Which of the following should you care about: confidentiality, integrity, availability?
- What else?

`https://policies.utexas.edu/policies/student-rights-under-family-educational-rights-and-privacy-act-ferpa` outlines some of the rules for UT Austin.

Metapolicy vs. Policy

A useful distinction is between the system *metapolicy* and the system *policy*.

metapolicy: The security goals in the most abstract sense.

policy: System-specific constraints intended to enforce the metapolicy.

Often the policy doesn't make sense unless you understand the metapolicy.

Metapolicy vs. Policy: Example

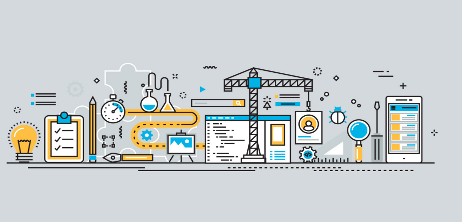
Policy:

1. faculty/staff may not use student social security numbers in documents/files/postings;
2. all older docs containing SSNs must be destroyed unless deemed necessary;
3. documents deemed necessary to retain must be kept in secure storage;
4. blah, blah, blah

MetaPolicy

Metapolicy: social security numbers of students should be protected from disclosure.

Security Design Process



How do you design a secure system? Here are some questions to ask:

1. What are you protecting and what are the potential threats? (risk assessment)
2. What is the intuitive notion of security for such a system? (metapolicy)
3. What are appropriate security rules that attempt to capture this notion for this system? (policy)

Security Design Process

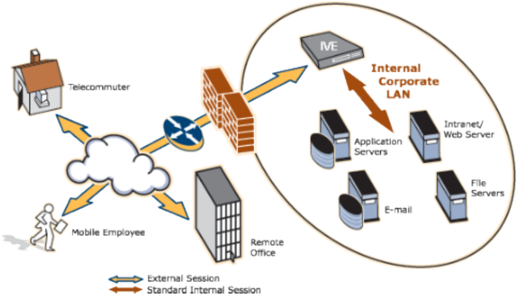
One the policy is defined, move on to system design, implementation, and evaluation.

4. What is an overall system architecture that supports our security goals? (system design)
5. By what specific mechanisms might the security goals be accomplished? (detailed design)
6. Does the system implementation accomplish the goal?
7. How certain can we be of our assessment?
8. Are there intuitively insecure behaviors that fall outside the range of the policy?

Of course, there are lots of other questions that need to be addressed as well in any development.

Suppose you have several secure LAN's that are geographically distributed, and must communicate securely over an insecure backbone network (the Internet).

Try to address the Security Design Process questions for this problem.



Thought Experiment #2

Caution: Don't jump too quickly to an implementation without considering what you're trying to accomplish.

If you think cryptography will solve your problem, then you don't understand cryptography ... and you don't understand your problem. –Bruce Schneier

A common tendency is to dive right to implementation. That makes the policy a set of rules with limited applicability beyond the particular type of application at hand.

Extended Thought Exeriment: MLS

Early security problem: protect the *confidentiality* of military secrets.

Given *information* at various levels of sensitivity and *individuals* having various degrees of trustworthiness, how do you control access to information within the system?

This problem is called *multi-level security* (MLS) or *military security*.

Multi-Level Security

The initial formalization we'll be considering was developed in 1973 by David Bell and Len LaPadula and is called the Bell and LaPadula model (BLP). It's still probably the most influential effort in the history of computer security.

Aside: MAC vs. DAC

Security systems should distinguish between:

Mandatory Access Controls (MAC): security rules that are enforced on every attempted access and not at the discretion of any system user;

Discretionary Access Controls (DAC): security rules that are enforced by the system at the discretion and behest of some users.

Example: the Unix file protection system implements DAC since the protections can be modified by the file owner.

For MLS, we'll focus on mandatory controls.

Aside: Military Multi-User Systems

In military systems, four models of operation are often defined for computers handling classified information:

dedicated: all users cleared for all information on machine; no need for access control (MILS);

system-high: all users cleared, but must obey need-to-know compartments (discretionary access control).

compartmented: all users cleared, but must be need-to-know compartments (mandatory access control). System must handle requests across classifications.

multi-level: not all users cleared for all information; system enforces access control (MLS).

MLS is the most difficult so not widely deployed.

(RAND Report R-609-1, "Security Controls for Computer Systems," (1970) summarizes best practices.)

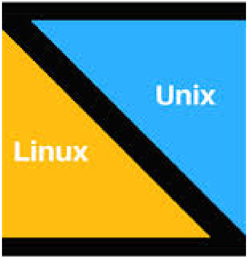
Mechanisms for Building a Solution

Often a security solution/policy (access control) is phrased in terms of the following three categories:

- **Objects:** the items being protected by the system (documents, files, directories, databases, transactions, etc.)
- **Subjects:** entities (users, processes, etc.) that execute activities and request access to objects.
- **Actions:** operations, primitive or complex, that can operate on objects and must be controlled.

For example, in the Unix operating system, processes (subjects) may have permission to perform read, write or execute (actions) on files (objects).

In addition, processes can create other processes, create and delete files, etc. Certain processes (running with `root` permission) can do almost anything.



That is one approach to the security problem.

Multi-Level Security (MLS)

Picture General Eisenhower's office in 1943 Europe:

The problem: Assume an environment in which there are various pieces of *information* at different sensitivity levels: the war plan, the defense budget, the base softball schedule, the general's laundry list, etc. Also, there are a variety of *individuals* with access to selected pieces of information: Eisenhower, Patton, privates, colonels, secretaries, janitors, spies, etc.

Note that this has nothing to do with computers!

Multi-Level Security (MLS)

The goal: Understand what "security" might mean in this context and define some rules to implement it.

Important proviso: For this thought experiment we are only concerned with *confidentiality*, not integrity or availability. This will lead to some counterintuitive results.

Risk Assessment

What are we protecting? Against what threats?

Notice: it's very important that we're only considering confidentiality in this thought experiment. Someone burning down the office and destroying the war plan might be a significant threat, but *it's not a threat to confidentiality.*

Confidentiality Questions

Recall the questions we asked about ensuring confidentiality:

1. How do you group and categorize information?
2. How do you characterize who is authorized to see what?
3. How are the permissions administered and checked? According to what rules?
4. How can authorizations change over time?
5. How do you control the flow of "permissions" in the system? Can I authorize others to view data that I am authorized to view?

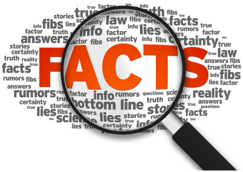
For simplicity, let's assume an environment of *static* permissions. That means we'll ignore questions 4 and 5.

Let's see if we can figure out some possible answers *for this specific setting* to the other questions.

Dividing Information

Back to our thought experiment: Gen. Eisenhower's office in 1943. The relevant "space" of information contains lots of individual atoms or factoids:

1. The base softball team has a game tomorrow at 3pm.
2. The Normandy invasion is scheduled for June 6.
3. The cafeteria is serving chopped beef on toast today.
4. Col. Jones just got a raise.
5. Col. Smith didn't get a raise.
6. The British have broken the German Enigma codes.
7. and so on.



Not all information is created equal. How do we group and categorize information rationally?

Object Sensitivity Levels

Information is parcelled out into documents/folders/objects/files. Documents (objects) are *labeled* according to some authority's assessment of their sensitivity level.

We'll assume a certain form for labels; they might be done differently.

Object Sensitivity Levels

One part of the label is taken from a linearly ordered set. One common scheme has levels: **Unclassified**, **Confidential**, **Secret**, **Top Secret**.

Object Sensitivity Levels

There are also "need-to-know" *categories*, from an unordered set, expressing membership within one or more interest groups, e.g., **Crypto**, **Nuclear**, **Janitorial**, **Embarrassing**, etc.

Some labels are special, but can be treated as need-to-know categories, e.g., **FOUO**, **No Foreign**, **Eyes Only**.

Object Sensitivity Levels

Ideally, the label on any document reflects the sensitivity of the information contained in that document. The label contains both a hierarchical component *and* a set of categories.

For example, two documents might have levels:

(Secret: {Nuclear}),
(Top Secret: {Crypto}).

One can expect that the first contains rather sensitive information related to the category Nuclear. This second contains highly sensitive information in category Crypto.

Some entity/agency/officer makes these labeling decisions. How they are made is outside the scope of our concern.

Object Sensitivity Levels

How do you label a document that contains "mixed information"?

- Suppose the document contains both sensitive and non-sensitive information?
- Suppose it contains information relating to both the Crypto and Nuclear domains?

Sometimes a decision is made that a document classification should be changed. This is called *downgrading* (or *upgrading*).

Authorization Levels

Individuals (subjects) have *clearances* or *authorization levels* that are typically of the same form as document sensitivity levels.

That is, each individual has:

- a hierarchical security level indicating the degree of trustworthiness to which he or she has been vetted;
- a *set* of "need-to-know categories" indicating groups to which he or she belongs or areas of interest in which he or she is authorized to operate.

Authorization Levels

The lowest security level in the system is called *system low*. For our MLS-type system it is **(Unclassified: $\{\ \}$).**

Higher clearances are assigned by some organization or government entity according to their assessment of the individual's trustworthiness and need for the information.

The highest (most permissive) level in the system, if it exists, is called *system high*. What would be system high for our MLS system?

Some levels may be unpopulated, i.e., no individual is cleared at that level.

Least Privilege: An Aside

The need-to-know categories are a reflection that even within a given security level (such as **Top Secret**) there is plenty of information to which not everyone cleared to that level should have access. This is an instance of:

Principle of Least Privilege:
Any subject should have access to the *minimum* amount of information needed to do its job.

This is as close to an axiom as anything in security. Why does it make sense?

Now What?

Given that we have labels for objects and clearances for subjects, how do we decide which subjects are permitted access to which objects?

Surely it's some relationship between the subject level and the object level. But what?

Now What?

For example, should a subject with clearance **(Secret: {Crypto})** be able to read a document labeled **(Confidential: {Crypto})**?

Should a subject with clearance **(Top Secret: {Crypto, Nuclear})** be able to modify a document labeled **(Confidential: {Crypto})**?

The Dominates Relation

Given a set of security labels (L, S), comprising hierarchical levels and categories, we can define a *partial order* among them.

Definition: (L_1, S_1) *dominates* (L_2, S_2) iff

1. $L_1 \geq L_2$ in the ordering on levels, and
2. $S_2 \subseteq S_1$.

We usually write $(L_1, S_1) \geq (L_2, S_2)$.

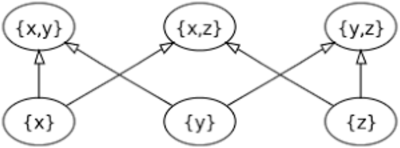
Note that this is *not* a total order. There are security labels A and B, such that neither $A \geq B$ nor $B \geq A$.

A partial order is a relation that is reflexive, transitive, and antisymmetric.

Reflexive: $x \geq x$

Transitive: $[x \geq y \wedge y \geq z] \rightarrow x \geq z$

Anti-symmetric: $[x \geq y \wedge y \geq x] \rightarrow x = y$

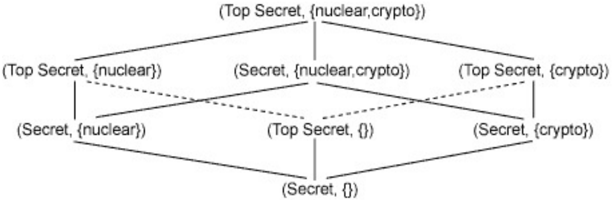


Exercise: Prove that dominates is a partial order.

Lattices

Algebraically, the (full) set of labels with their ordering would form a *lattice*. This is sometimes called "lattice-based security."

In mathematics, a lattice is a partially ordered set (or poset), in which all nonempty finite subsets have both a supremum (join or lub) and an infimum (meet or glb). Lattices can also be characterized as algebraic structures that satisfy certain identities.



Exercise: Suppose you have two hierarchical levels H and L such that $L < H$, and two categories A and B. Using the dominates relation as the partial order, draw the lattice of levels in this system.

How many levels are possible?

Some Answers

Remember our questions about confidentiality! Given our mechanisms for classifying objects (data / files) according to security labels, and personnel according to clearances, what are the answers?

1. How do you group and categorize information? The grouping is done as documents (files) and categorized according to labels.

What does that mean? Who assigns the labels? What about documents that contain "mixed" information?

Some Answers: Continued

2. How do you characterize who is authorized to see what?

The answer seems to be a relationship between the sensitivity level of a document (file) and the authorization level of the individual.

- What is the appropriate relationship?
- How do we codify it as a system of rules for access within this system?
- Does permission depend on the type of access requested? For example, are read and write access interchangeable?

Secure Reading

Suppose subject S with authorization (L_S, C_S) asks to *read* an object O with classification (L_O, C_O). Under what conditions should the request be granted by the system?

For example, suppose a subject has clearance **(Secret: {Crypto})**. Which of the following should he be able to read?

- document labeled **(Confidential: {Crypto})**
- document labeled **(Top Secret: {Crypto})**
- document labeled **(Secret: {Nuclear})**
- document labeled **(Secret: {Crypto, Nuclear})**

So what is the formal rule?

Simple-Security Property

According to the Bell-LaPadula Model (BLP), one of the earliest formal security policies, the first rule governing access is:

> **The Simple-Security Property:** *Subject S with clearance (L_S, C_S) may be granted read access to object O with classification (L_O, C_O) only if (L_S, C_S) dominates (L_O, C_O).*

We will often write "(L_S, C_S) dominates (L_O, C_O)" as "$(L_S, C_S) \geq (L_O, C_O)$," but recall that it involves both hierarchical levels and need-to-know categories.

Simple-Security Property

The Simple Security Property models read access in the world of military documents *and* attempts to codify it for the world of electronic information storage.

The Simple-Security Property: Subject S with clearance (L_S, C_S) may be granted *read* access to object O with classification (L_O, C_O) only if $(L_S, C_S) \geq (L_O, C_O)$.

- Why is it "only if" and not "if and only if"?
- Does this work in an electronic context?
- Is it all that is needed? Why or why not?

Secure Writing

The Simple-Security property codifies restrictions on *read* access to documents. What about *write* access?

Is the problem different with respect to writing in the electronic context than it is in the world of military documents? Why or why not?

More generally, what assumptions can be made about *persons* in the world of military paper documents that cannot be made about *subjects* (processes) in the context of computers?

Read Permission vs. Write Permission: The Berger Case

"Sandy" Berger served as the National Security Advisor under President Bill Clinton from 1997 to 2001.

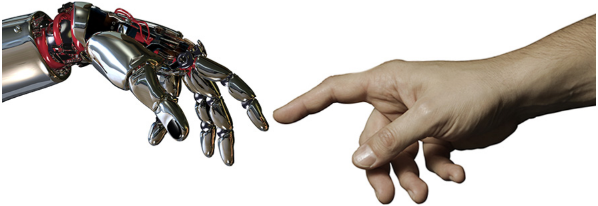
U.S. DoJ prosecuted Berger for unauthorized removal and destruction of classified documents in October 2003 from a National Archives reading room, by stuffing them down his pants.

In April 2005, Berger pled guilty to a misdemeanor charge of unauthorized removal and retention of classified material.

Berger obviously had permission to read the documents. Why was he prosecuted?

Secure Writing

Subjects in the world of military documents are assumed to be *persons* trusted not to disclose (write) to unauthorized parties information to which they have legitimate access.



Subjects in the world of computing are often *programs* operating on behalf of a trusted user (and with his or her clearance). The program may have embedded malicious logic (e.g., a "trojan horse") that "leaks" information without the knowledge of the authorized user.

Mandatory controls on the write accesses of (computer) subjects is needed that might not be necessary for persons. This is sometimes called the *confinement problem*.

What is the appropriate restriction on writing?

Earlier we asked: Should a subject with clearance **(Top Secret: {Crypto, Nuclear})** be able to modify a document labeled **(Confidential: {Crypto})**? What do you think now?

The *-Property

In the Bell-LaPadula Model of security, the following rule is enforced to restrict *write* access:

> **The *-Property:** *Subject S with clearance (L_S, C_S) may be granted write access to object O with classification (L_O, C_O) only if $(L_S, C_S) \leq (L_O, C_O)$.*

This is pronounced "the star property." Why do you think it's called that?

The *-Property

Does this rule make sense? Is it too restrictive? Is it too lax?

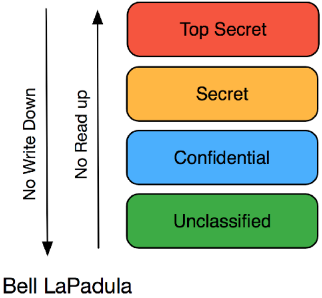
According to the *-property, can a commanding general with a top secret clearance email marching orders to a foot soldier? No!

According to the *-property, can a corporal with no clearance overwrite the war plan? Yes, *but that's an integrity problem*!

The BLP Properties

The simple-security and *-property are sometimes characterized as "read down" and "write up," respectively.

Alternatively, they're characterized as "no read up" and "no write down."



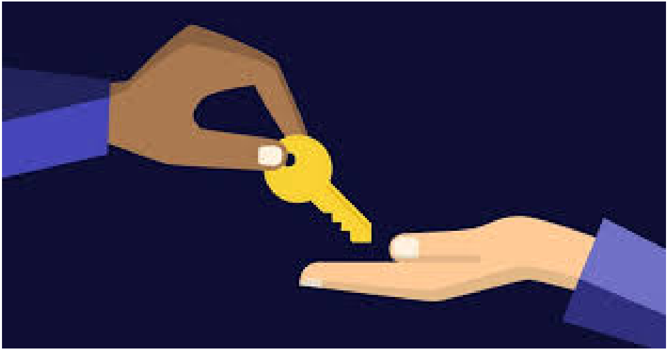
What do you think is the metapolicy of this security system? Prove that subject S can both read and write object O only if the levels are equal.

Trusted Subjects

Often, to get around the more onerous restrictions of a mandatory policy, an implementation may add *trusted subjects*, specialized subjects permitted to operate "outside the rules of the policy" in very constrained ways.

Example: The *-property implies that the general can never send an email to the private. We add a special *downgrader* subject to the system and extend the *-property with the proviso that an object's level can be reduced in specific ways only if the object's contents are reviewed by the downgrader subject *including visual inspection by a trained human being*.

Trusted Subjects



Notice: this technically violates the *-property, but prevents any malicious *program* from leaking information (unless it is clever enough to fool the downgrader). See Steganography.

Information Flow

Notice that Simple Security and the *-Property control two ways in which information can flow from A to B.

1. A can "push" information to B by writing objects in B's space. The *-Property fixes that.
2. B can "pull" information from A by reading objects in A's space. Simple Security fixes that.

Are there additional ways that information can flow from A to B that don't involve either of those mechanisms?

Other Permissions

Our discussion of the Bell and LaPadula model explicitly included **Read** and **Write** access, but not **Create**, **Destroy**, **Execute**, **Append**, others. How might we add these operations to our BLP framework?

In particular, is **Execute** effectively a modify (write) operation? A reference (read) operation? Neither? Both?

Other Permissions

Our discussion of the Bell and LaPadula model explicitly included **Read** and **Write** access, but not **Create**, **Destroy**, **Execute**, **Append**, others. How might we add these operations to our BLP framework?

In particular, is **Execute** effectively a modify (write) operation? A reference (read) operation? Neither? Both?

Maybe that's the wrong way to think about execute. Maybe it *creates a subject* with the creator's permission levels. Then, aren't Simple Security and the *-Property adequate?

Basic Security Theorem

According to BLP, security is essentially defined as follows:
Definition: A system is *secure* if it always satisfies Simple Security and the *-Property.

Bell and LaPadula proved the following theorem:

The Basic Security Theorem: Let Σ be a system with a secure initial state σ_0, and let T be a set of state transitions. If every element of T preserves the simple security condition and the *-property, then every $\sigma_i, i \geq 0$, is secure.

Proof: Simple induction over i

System Z

John McLean (NRL) pointed out that the Basic Security Theorem says something about *states* of the system, but nothing about *what transitions* may occur.

Consider System Z: any attempt to read a file causes all objects and subjects in the system to be downgraded to security level system-low.

The Basic Security Theorem can still be proved for this system but it is obviously insecure.

What's the Point

McLean argued that reasoning merely about *states* isn't adequate. It is also necessary to reason about *transitions*.

Bell responded that the BLP model only provides a framework for reasoning about secure systems, not a definition of security.

One possible Lesson: Formal definitions and theorems don't guarantee anything unless they are validated against reality. Any *interpretation* of the formalism is as valid as any other.

This controversy raised questions about the "foundations" of computer security research.

Tranquility

To deal with transitions, one could add either:

The Strong Tranquility Property: No subjects or objects may change levels during the lifetime of the system.

The Weak Tranquility Property: Subjects and objects may not change levels *in a way that violates Simple Security or the *-Property.*

Is this useful? Is it overly restrictive? What if a user needs to operate at different levels during the course of the day?

Weak Tranquility

The Weak Tranquility Property: (more general form) Subjects and objects do not change levels *in a way that violates the "spirit" of the security policy.*

What does this mean?

- Suppose your system includes a command allowing any subject to *lower* the level of any other subject/object. Does that violate the goals of simple security or the *-property?
- Suppose your system includes a command to *raise* the level of a subject/object. Does that violate the goals of simple security or the *-property?

BLP in a Nutshell

The Bell and LaPadula Model in its original incarnation was somewhat more complex, but a thumbnail version is simply:

- Simple Security Property
- the *-Property
- some version of the Tranquility Property.

Other Concerns

Are simple-security, the *-property, and the tranquility property adequate to ensure confidentiality within the system?
What about the following issues:

- What about other types of operations? Can every operation be thought of as a *read* or *write*? Can some be both?
- What useful operations can you imagine that might subvert the protections offered?
- Our restrictions control access by subjects to objects. Are there ways in which information might be compromised without explicit read or write operations?