



Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

CSE543 - Introduction to Computer and Network Security Module: Access Control

Professor Patrick McDaniel
Fall 2008

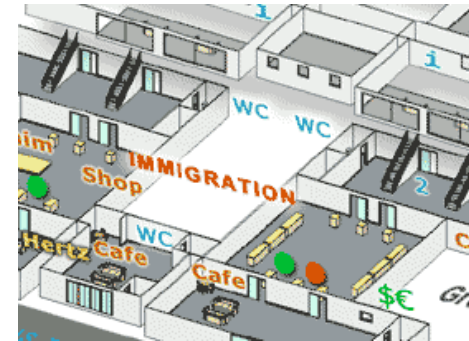
Trusted Computing Base

- The trusted computing base is the infrastructure that you assume will behave correctly
 - ▶ Hardware (keyboard, monitor, ...)
 - ▶ Operating Systems
 - ▶ Implementations
 - ▶ Local networks
 - ▶ Administrators
 - ▶ Other users on the same system
- Axiom: the larger the TCB, the more assumptions you must make (and hence, the more opportunity to have your assumptions violated).



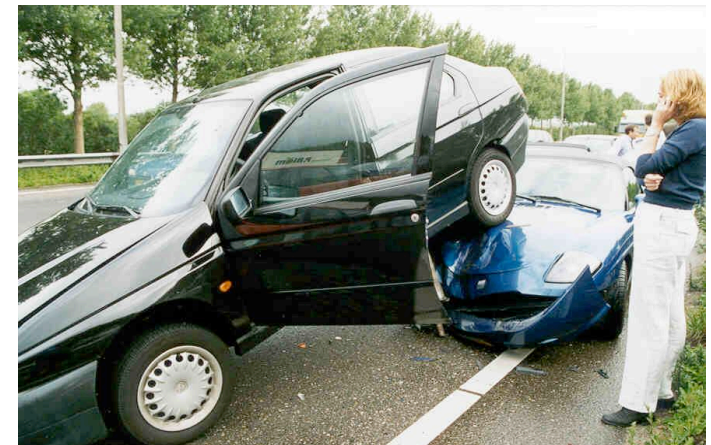
Blindly Following Policy

- First, what is a policy?
 - ▶ Some statement of secure procedure or configuration that parameterizes the operation of a system
 - ▶ Example: Airport Policy
 - ▶ Take off your shoes
 - ▶ No bottles that could contain > 3 ozs
 - ▶ Empty bottles are OK?
 - ▶ You need to put your things through X-ray machine
 - ▶ Laptops by themselves, coat off
 - ▶ Metal detector
- Purpose: prevent on-airplane (metal) weapon ...



... when policy goes wrong

- Driving license test: take until you pass
 - ▶ Mrs. Miriam Hargrave of Yorkshire, UK failed her driving test **39** times between 1962 and 1970!!!!
 - ▶ ... she had 212 driving lessons
 - ▶ She finally got it on the 40th try.
 - ▶ Some years later, she was quoted as saying, “sometimes I still have trouble *turning right*”



Access Control/Authorization

- An ***access control*** system determines what ***rights*** a particular ***entity*** has for a set of ***objects***
- It answers the question
 - ▶ E.g., do ***you*** have the right to ***read /etc/passwd***
 - ▶ Does ***Alice*** have the right to ***view*** the ***EECS website?***
 - ▶ Do ***students*** have the right to ***share project data?***
 - ▶ Does ***Dr. McDaniel*** have the right to ***change*** your ***grades?***

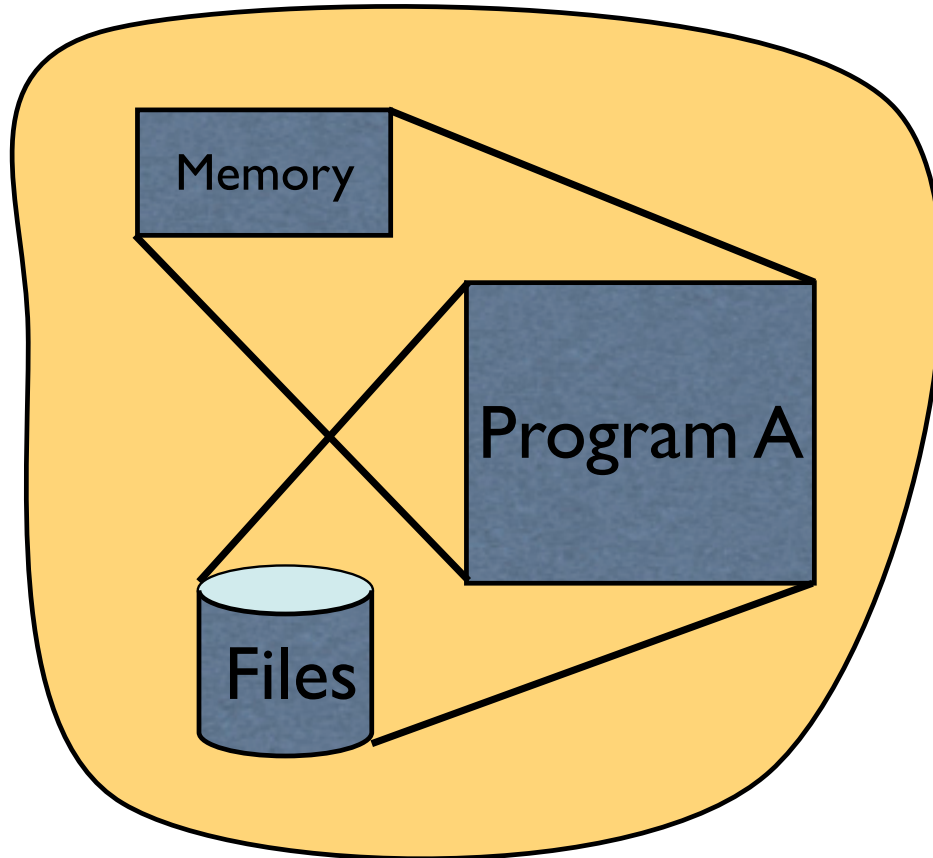
- An ***Access Control Policy*** answers these questions

Simplified Access Control

- **Subjects** are the active entities that do things
 - ▶ E.g., **you, Alice, students, Dr. McDaniel**
- **Objects** are passive things that things are done to
 - ▶ E.g., **/etc/passwd, CSE website, project data, grades**
- **Rights** are actions that are taken
 - ▶ E.g., **read, view, share, change**

Protection Domains

Protection domain



- The *protection domain* restricts access of external parties to our computing system's resources
- How is this done today?
 - ▶ Memory protection
 - ▶ E.g., UNIX protected memory, file-system permissions (rwx...)
- A *protection state* describes access of all programs

Access Control Policy

- “A policy is a set of acceptable behaviors.”
- F. Schneider

- An access control *policy* is a function:

$$P(S,O,R) \rightarrow \{ \text{accept, deny} \}$$

- ▶ Where, set S=subjects, O=objects, R=rights
- The policy is a lot of these tuples, whether explicitly represented that way or not.
- There are many, many ways to represent these.

The Access Matrix

- An access matrix is one way to represent policy.
 - Frequently used mechanism for describing policy
- Columns are objects, subjects are rows.
- To determine if S_i has right to access object O_j , find the appropriate entry.
- Succinct descriptor for $O(|S|*|O|)$ entries
- There is a matrix for each right.

	O_1	O_2	O_3
S_1	Y	Y	N
S_2	N	Y	N
S_3	N	Y	Y

Designing an access control system



- Separation of policy from mechanism
 - ▶ We enforce policy via mechanism, e.g., the filesystem, etc.
 - ▶ Policy is that which specifies rights
- **Idea:** separation gives us the ability to change the meaning of policy or the enforcement of it quickly

“Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one’s subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects.

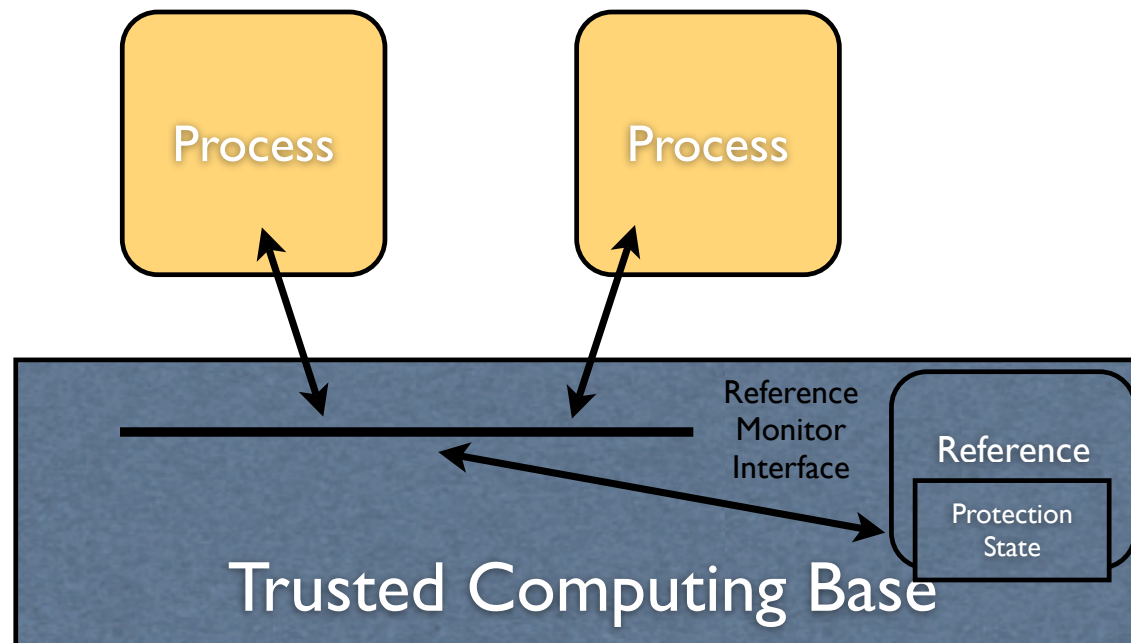
We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day. But nothing is gained on the contrary by tackling these various aspects simultaneously. It is what I sometimes have called **the separation of concerns.**” (Dijkstra)

Access Policy Enforcement

- A *protection state* defines what each subject can do
 - ▶ E.g., in an access matrix
- A *reference monitor* enforces the protection state
 - ▶ A service that responds to the query...
- A correct reference monitor implementation meets the following guarantees
 - ▶ Tamperproof
 - ▶ Complete Mediation
 - ▶ Simple enough to verify
- A protection system consists of a protection state, operations to modify that state, and a reference monitor to enforce that state

Access Control

- Reference Monitor is Central to Authorization



- Consider the Trust and Threat Models in Authorization

Access Control

- Suppose the private key file for J is object O_1
 - ▶ Only J can read
- Suppose the public key file for J is object O_2
 - ▶ All can read, only J can modify
- Suppose all can read and write from object O_3
- What's the access matrix?

	O_1	O_2	O_3
J	?	?	?
S_2	?	?	?
S_3	?	?	?

Trusted Processes

- Does it matter if we do not trust some of J's processes?

	O_1	O_2	O_3
J	R	RW	RW
S_2	N	R	RW
S_3	N	R	RW

Secrecy

- Does the following protection state ensure the secrecy of J's private key in O_1 ?

	O_1	O_2	O_3
J	R	RW	RW
S_2	N	R	RW
S_3	N	R	RW

Integrity

- Does the following access matrix protect the integrity of J's public key file O_2 ?

	O_1	O_2	O_3
J	R	RW	RW
S_2	N	R	RW
S_3	N	R	RW

Protection vs Security

- Protection
 - ▶ Security goals met under *trusted* processes
 - ▶ Protects against an error by a non-malicious entity
- Security
 - ▶ Security goals met under *potentially malicious* processes
 - ▶ Protects against any malicious entity
 - ▶ Hence, For J:
 - Non-malicious process shouldn't leak the private key by writing it to O_3
 - A potentially malicious process may contain a Trojan horse that can write the private key to O_3

Least Privilege

- Limit permissions to those required and no more
- Consider three processes for user J
 - ▶ Restrict privilege of the process J_1 to prevent leaks

	O_1	O_2	O_3
J_1	R	R	N
J_2	N	RW	N
J_3	N	R	RW

Access Control Administration



There are two central ways to specify a policy

1. Discretionary - object “owners” define policy

- ▶ Users have discretion over who has access to what objects and when (trusted users)
- ▶ Canonical example, the UNIX filesystem
 - RWX assigned by file owners

2. Mandatory - Environment enforces static policy

- ▶ Access control policy defined by environment, user has no control over access control (untrusted users)
- ▶ Canonical example, process labeling
 - System assigns labels for processes, objects, and a dominance calculus is used to evaluate rights

DAC vs. MAC

- Discretionary Access Control
 - ▶ User defines the access policy
 - ▶ Can pass rights onto other subjects (discretion)
 - ▶ Their programs can pass their rights
 - Consider a Trojan horse
- Mandatory Access Control
 - ▶ System defines access policy
 - ▶ Subjects cannot pass rights
 - ▶ Subjects' programs cannot pass rights
 - Consider a Trojan horse here



Administrative Operations

- An access matrix defines a protection state
- A protection system also includes a set of operations for modifying that state
- Examples
 - ▶ **Add right (UNIX)**: If the user is the owner of the object, then the user can add an operation to set of operations of another user
 - ▶ **Add right (Lampson)**: If domain has the **copy flag** set for that right in its access matrix row, then it can add that right to any other domain's access row

DAC vs. MAC in Access Matrix

- Subjects:
 - ▶ DAC: users
 - ▶ MAC: labels
- Objects:
 - ▶ DAC: files, sockets, etc.
 - ▶ MAC: labels
- Operations:
 - ▶ Same
- Administration:
 - ▶ DAC: owner, copy flag, ...
 - ▶ MAC: external
- MAC: largely static matrix; DAC: all can change

	O ₁	O ₂	O ₃
S ₁	Y	Y	N
S ₂	N	Y	N
S ₃	N	Y	Y

Saltzer-Schroeder Design Goals

- Classic paper on security describing goals
 - *The protection of information in computer systems*, Saltzer, J.H. and Schroeder, M.D., Proceedings of the IEEE, Publication Date: Sept. 1975, 63(9):1278- 1308.
- Goals (principles)
 - ▶ Economy of Mechanism
 - ▶ Fail-Safe Defaults
 - ▶ Complete Mediation
 - ▶ Separation of Privilege
 - ▶ Least Privilege
 - ▶ Open Design
 - ▶ Least Common Mechanism
 - ▶ Psychological Acceptability



Conflicting Goals

- Challenges of building a secure system
 - ▶ What are the *users'* goals?
 - ▶ What do *application developers* want?
 - ▶ What about the *data owners* (corporations/governments)?
 - ▶ What is the purpose of *system administrators*?
 - ▶ What about the requirements of *operating system designers*?

- Need a *satisfying* balance among these goals?

Access Policy Goals

- Rights assignment is the process of describing a security goal
- “*Principle of least privilege*”
 - ▶ You should provide the minimal set or rights necessary to perform the needed function
 - ▶ **Implication 1:** you want to reduce the protection domain to the smallest possible set of objects
 - ▶ **Implication 2:** you want to assign the minimal set of rights to each subject
 - ▶ **Caveat:** of course, you need to provide enough rights and a large enough protection domain to get the job done.
- What other kinds of policy goals are there?

Policy Goals

- **Secrecy**
 - ▶ Don't allow reading by unauthorized subjects
 - ▶ Control where data can be written by authorized subjects
 - Why is this important?
- **Integrity**
 - ▶ Don't permit dependence on lower integrity data/code
 - Why is this important?
 - ▶ What is "dependence"?
- **Availability**
 - ▶ The necessary function must run
 - ▶ Doesn't this conflict with above?