Authorization

UT LAW396V

LECTURE NOTES

Authentication/Authorization

Validating Identity

Permissions Assigned to a Validated Identity

A Framework

Policy

Mechanism

Assurance

Incentives

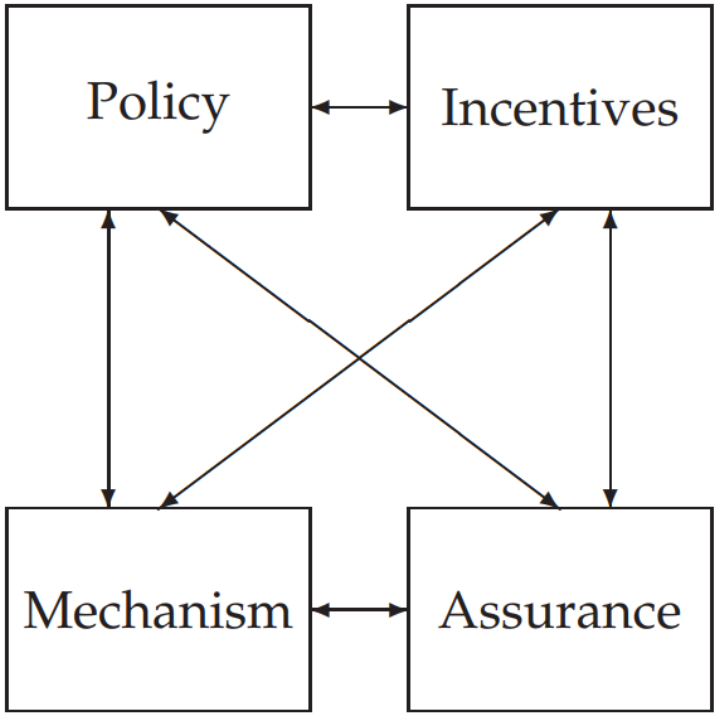


Figure 1.1: Security Engineering Analysis Framework

What a Security Policy is ***NOT***

- Generic platitude statements
- Butt-covering for legal/regulation
- Aspirational, motivational, etc

Megacorp Inc security policy

1. This policy is approved by Management.
2. All staff shall obey this security policy.
3. Data shall be available only to those with a 'need-to-know'.
4. All breaches of this policy shall be reported at once to Security.

Figure 8.1: A typical corporate information security policy

From, *Security Engineering*, 2nd Ed., Ross Anderson, 2008

Ross Anderson's Observations

1. It dodges the central issue, namely 'Who determines ''need-to-know'' and how?'

2. How are breaches to be detected and who has a specific duty to report them?

From, *Security Engineering*, 2nd Ed., Ross Anderson, 2008

What a Security Policy *IS*

- Specific, testable properties
- A strategy for security
- Example:

"All checks over $10,000 must be signed by two managers."

Policy vs Policy Model

- I prefer “policy” for each statement
- I prefer “policy model” for the combination
- Anderson uses them interchangeably
- There are other formulations

Policy and Authorization

- Policy (model) is the security strategy
- Testable security statements
- ***Often an authorization model***
- (Policy defines what is authorized)

What Policy Does/Does Not

- ***Defines*** a way of measuring/testing security
- See previous “signed by 2 managers” example
- Does ***not*** prevent “something bad happening”
- Is ***not*** guaranteed to be “right”
- If policy is right, bad things ***can*** happen
- If policy is wrong, bad things ***will*** happen

Conceptual Building Blocks

- Nothing here is ***implementation***
- There are still conceptual components:
  - ***Permission Models***
  - ***User Models***
  - ***Data Models***
  - ***Enforcement Models***
  - ***Objectives***

Access Controls

The mechanism by which authorization permissions are managed

Within most information systems, the most common controls:

- (C)reate
- (R)ead
- (U)pdate
- (D)elete

Most other controls can be thought of as a form of one of these

Every-day Approaches

ACCESS CONTROL LISTS

CAPABILITIES

One View of ACL/Capabilities

User	Accounting Data
Sam	rw
Alice	rw
Bob	r

Figure 4.4: Access control list (ACL)

User	Operating System	Accounts Program	Accounting Data	Audit Trail
Bob	rx	r	r	r

Figure 4.5: A capability

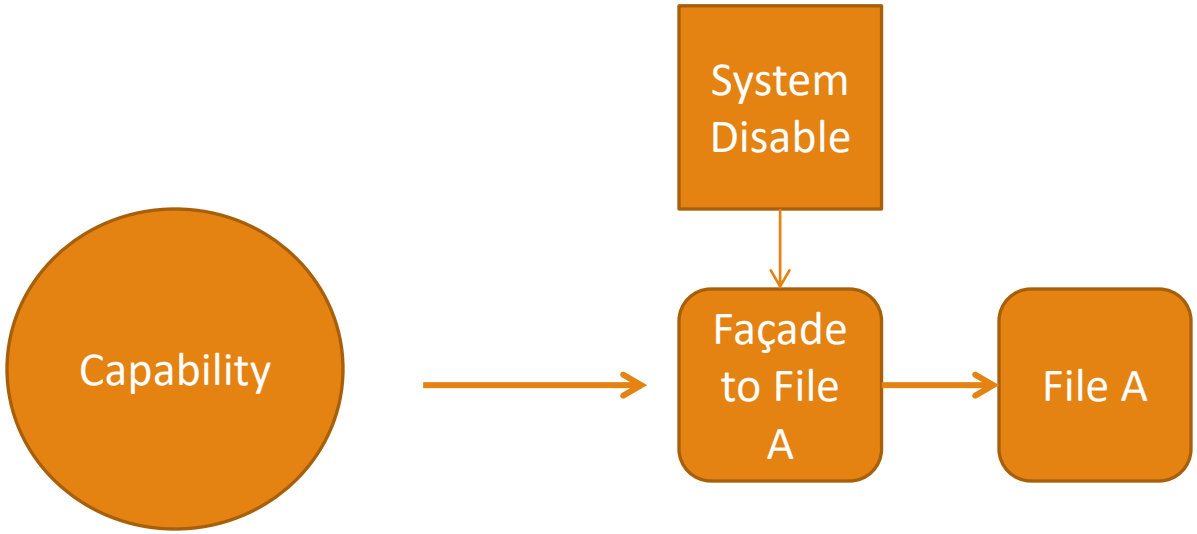
Broader Concept

A ***capability*** is an *enabling* technology for access

An ***access control list*** is a *filtering* technology for access

Opponents of capabilities argue that you cannot change a file's status

They just don't understand capabilities



MAC vs DAC

Mandatory Access Controls – what is permitted is determined by policy

Discretionary Access Controls – what is permitted is determined by user

Multi-Level Security (MLS)

Users and data are assigned classifications

What users are permitted to do with data depends on both labels

Relationship to MAC:

- Some use it interchangeably
- Some define parts of such a system as MAC (see next slide)
- Anderson does not say the policy is MAC, but the controls that enforce it are
- ***I prefer Anderson's formulation***

Bell Lapadula Model

Design emerged from military document classification

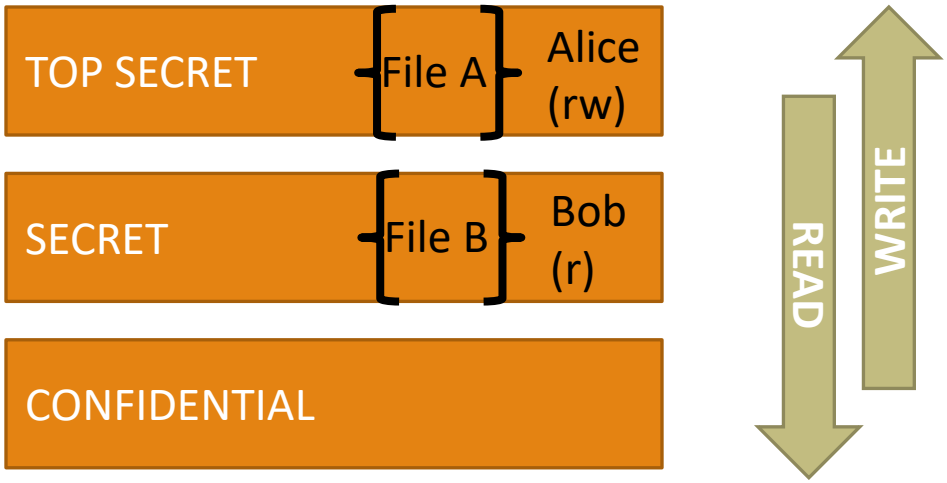
Three protection properties

- *Simple Security Property:* No Read Up (NRU)
- **-Property:* No Write Down (NWD)
- Discretionary Access Controls ***within the label***

No Read Up/No Write Down

The *-property was the big innovation of BLP. It *assumed* trojans and buggy code!

Discretionary Access



BLP as a Model Security Policy

- BLP is a well defined security policy
- So… is a system with BLP ***secure?***
- BLP itself is relatively easy to understand and enforce, ***BUT:***
  - **Is it the right security policy?**
  - **Is it going to work at the edges?**

BLP Edge #1: Declassifying Data

- What if we ***do*** want to write data down?
- Original BLP "gets around" this by having trusted subjects
- The NWD policy only applies to ***untrusted*** subjects
- But there is no definition for trusted/untrusted
- Trusted subjects introduce ***two*** risks:
  - Risks for any trusted subject
  - Risks that designers will make too many subjects trusted
- Other solutions: security officer, additional policy, etc

BLP Edge #2: Creation of labels

- Model does not say how to create data, subjects, or labels
- Described by the creators of BLP, but not part of the model
- Common solutions are data created by subject at same level
- But how do subjects get their level?

Example of Additional Policy

- **Strong tranquility**: security labels never change during operation
  - Example: put system into offline state to make changes
- **Weak tranquility**: labels never change in a way that violates security policy
  - As subject accesses info that is higher, their level increases
  - At any given time, the NWD policy is enforced

BLP Edge #3: Data Doesn't Flow

- The model can "work too well."
- Data becomes compartmentalized
- Data flows upward, duplication, etc., etc., etc.
- In other words, sometimes even working "right" is "wrong"

Biba model

Upside-down BLP

- You can only read up and write down
- The goal is *integrity* not *confidentiality*

Partially used in Vista. Uses the NoWriteUp.

- Most files are “medium” or higher. IE is “low”
- So, things downloaded can read most files, *but not write to them!*

Why BLP or Biba?

- BLP ***primarily enforces confidentiality***
- Biba ***primarily enforces integrity***
- Obviously, picking the ***right*** model for a system is crucial
- Remember: many systems fail because the designers ***protect the wrong things*** or protect the right things but ***in the wrong way***.

Domain and Type Enforcement

- DTE assigns a “type” to data objects (e.g., files)
- DTE assigns a “domain” to subjects (e.g., user processes)
- Rules for domain-to-domain and domain-to-type
- Used in SE-Linux and Android
- Powerful, but can be complicated/hard to use
- Perhaps not a real model, but a ***model framework***

DTE Policy #1

This policy gives all permissions to every file on the system.

```
type unrestricted_t;

domain unrestricted_d = (/sbin/init),
                        (cdrwx->unrestricted_t);

initial_domain =  unrestricted_d;

assign -r unrestricted_t /;
```

From “DOMAIN TYPE ENFORCEMENT”
https://acv.cs.mtu.edu/dteTutorial.htm

DTE Policy #2

This policy gives restricts “simpleDaemon” to only rw etc/simpleDaemon file.

```
01    type unrestricted_t, simpleDaemon_t;
02
03    domain unrestricted_d = (/sbin/init),
04                            (cdrwx->unrestricted_t),
05                            (drwx->simpleDaemon_t),
06                            (auto->simpleDaemon_d);
07
08    domain simpleDaemon_d = (/bin/simpleDaemon),
09                            (rw->simpleDaemon_t);
10
11    initial_domain = unrestricted_d;
12
13    assign -r unrestricted_t /;
14    assign    simpleDaemon_t /etc/simpleDaemon;
```

From “DOMAIN TYPE ENFORCEMENT”
https://acv.cs.mtu.edu/dteTutorial.htm

Role-Based Access Controls

- RBAC is widely used commercially
- Each user of the system has one or more roles
- Each role has various permissions (can be MAC or DAC)
- Each role's permissions should be specific/limited
- User may switch roles as needed
- Problems include role-creep, data rot, etc.

RBAC Example #1

Permission/Role	Writer	Reader
Edit	Yes	No
Delete	Yes	No
Read	Yes	Yes

From “What is Role-Based Access Control (RBAC)? Examples, Benefits, and More” https://www.upguard.com/blog/rbac

RBAC Example #2

Instead, you implement RBAC, creating some permissions that users of your gift shop POS module would need:

- `read:catalog-item`
- `read:customer-profile`
- `create:invoice`

And to make these easier to manage, you create a role called `Gift Shop Manager` and add these permissions to that role.

Similarly, you create permissions for users of your marketing module, which include:

- `create:newsletter`
- `edit:newsletter`
- `delete:newsletter`
- `send:newsletter`
- `edit:distribution-list`

From “Sample Use Cases: Role-Based Access Control” https://auth0.com/docs/manage-users/access-control/sample-use-cases-role-based-access-control

Attribute-Based Access Controls

- ABAC includes all of RBAC but adds additional information
- ABAC also includes attributes: time of day, device, etc.
- ABAC is seen as being exceptionally expressive
- Like DTE, can be very complicated and hard to get right

Access Control Principles

- Least privilege
- Separation of duties/concerns
- Accountability/Auditability
- “Conditional” Access

Why Access Controls are Hard

- Hard to model all usage
- For example “Side Channels”
- Another example “Inference Controls”

Inference

Information sharing often involves some kind of “scrubbing”

In MLS, a report is redacted before moving down a security layer

In privacy-preserving systems, data is often *anonymized*

The problem, of course, is inference

- People can often be identified by their medical records even with names removed
- And, of course, we’ve seen this with AOL and Google

Inference Control

Characteristic formula – the query instructions to get some set

Query set – the set produced by a characteristic formula

Sensitive Statistics – stats that deanonymize information:

- For example, if the set is too small, than we've identified an individual by attributes

Query Size

You can limit how small a result is from a query

But you also have to worry about returning N-1!!

Also, you have to deal with using multiple queries to get a smaller than N intersection

Anderson's Example Policy

1. Access control: each identifiable clinical record shall be marked with an access control list naming the people who may read it and append data to it.
2. Record opening: a clinician may open a record with herself and the patient on the access control list. Where a patient has been referred, she may open a record with herself, the patient and the referring clinician(s) on the access control list.
3. Control: One of the clinicians on the access control list must be marked as being responsible. Only she may alter the access control list, and she may only add other health care professionals to it.
4. Consent and notification: the responsible clinician must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever responsibility is transferred. His consent must also be obtained, except in emergency or in the case of statutory exemptions.

Anderson's Example Policy

statutory exemptions.

5. Persistence: no-one shall have the ability to delete clinical information until the appropriate time period has expired.
6. Attribution: all accesses to clinical records shall be marked on the record with the subject's name, as well as the date and time. An audit trail must also be kept of all deletions.
7. Information flow: Information derived from record A may be appended to record B if and only if B's access control list is contained in A's.
8. Aggregation control: there shall be effective measures to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add to their access control list already has access to personal health information on a large number of people.
9. Trusted computing base: computer systems that handle personal health information shall have a subsystem that enforces the above principles in an effective way. Its effectiveness shall be subject to evaluation by independent experts.