
Security Policies

CS461/ECE422

Information Assurance

Fall 2007

Overview

- Natural language policies
- Implementation policies
 - High level
 - Low level

Reading Material

- *Security in Computing*,
 - Organizational Security policies - Chapter 8.3
 - Privacy policies – Chapter 10.1-10.2
 - Security Policies – Chapter 5.2
- SANS policy project
 - <http://www.sans.org/resources/policies/>
- *Introduction to Computer Security and Computer Security: Art and Science*, Chapter 4
 - UC Davis natural language policy example.

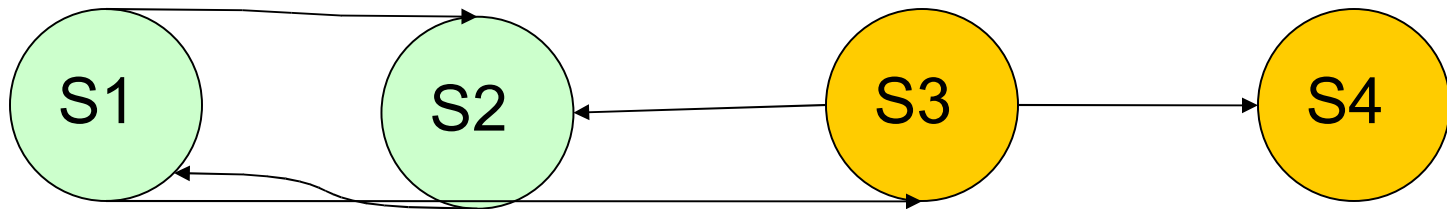
Motivation

- Security Policies guides implementation
 - Reflects what one can assume about an organization
- Policy occurs at multiple levels
 - Policy-driven management

Security Policy

- Policy partitions system states into:
 - Authorized (secure)
 - These are states the system can enter
 - Unauthorized (nonsecure)
 - If the system enters any of these states, it's a security violation
- Secure system
 - Starts in authorized state
 - Never enters unauthorized state

Authorized System States



Components of Security Addressed in Policy

- Basic Security Components
 - Confidentiality
 - Integrity
 - Availability

Policy Models

- Abstract description of a policy or class of policies
- Types of policies
 - Military (governmental) security policy
 - Policy primarily protecting confidentiality
 - Commercial security policy
 - Policy primarily protecting integrity
 - Confidentiality policy
 - Policy protecting only confidentiality
 - Integrity policy
 - Policy protecting only integrity
 - Service Level Agreements
 - Availability agreements

Question

- Policy disallows cheating
 - Includes copying homework, with or without permission
- CS class has students do homework on computer
- Anne forgets to read-protect her homework file
- Bill copies it
- Who cheated?
 - Anne, Bill, or both?

Answer Part 1

- Bill cheated
 - Policy forbids copying homework assignment
 - Bill did it
 - System entered unauthorized state (Bill having a copy of Anne's assignment)
- If not explicit in computer security policy, certainly implicit
 - Not credible that a unit of the university allows something that the university as a whole forbids, unless the unit explicitly says so

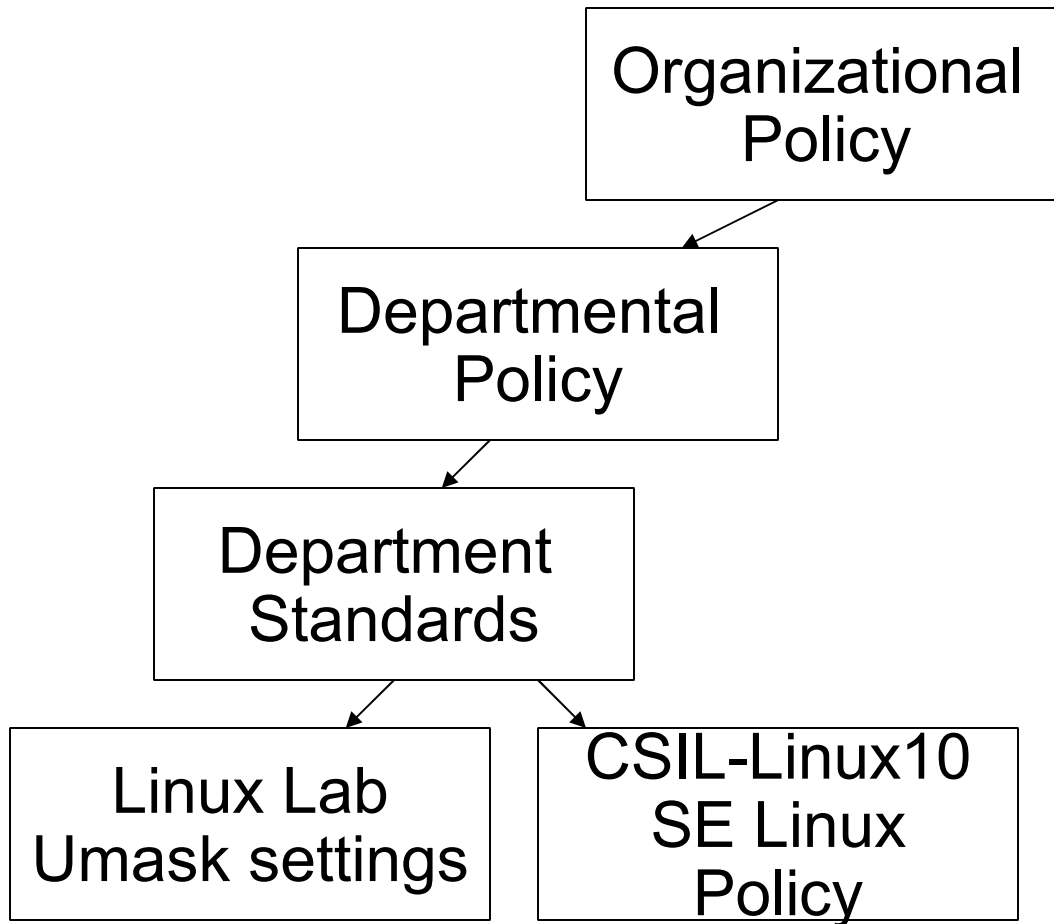
Answer Part #2

- Anne didn't protect her homework
 - Not required by security policy
- She didn't breach security
- If policy said students had to read-protect homework files, then Anne did breach security
 - She didn't do this

Mechanisms

- Entity or procedure that enforces some part of the security policy
 - Access controls (like bits to prevent someone from reading a homework file)
 - Disallowing people from bringing CDs and floppy disks into a computer facility to control what is placed on systems

Hierarchy of Policies



Natural Language Security Policies

- Targeting Humans
 - Written at different levels
 - To inform end users
 - To inform lawyers
 - To inform technicians
- As with all policies, should define purpose not mechanism
 - May have additional documents that define how policy maps to mechanism
- Should be enduring
 - Don't want to update with each change to technology
- Shows due diligence on part of the organization

Security Policy Contents

- Purpose – Why are we trying to secure things
- Identify protected resources
- Who is responsible for protecting
 - What kind of protection? Degree but probably not precise mechanism.
- Cover all cases
- Realistic

Example Privacy policies

- Busey Bank - <http://busey.com/>
 - Financial Privacy Policy
 - Targets handling of personal non-public data
 - Clarifies what data is protected
 - Who the data is shared with
 - Web Site Privacy Policy
 - Outlines how data is handled on the web site
 - Has a link to another document more security mechanism details

Example Acceptable Use Policy

- IEEE Email Acceptable Use Policy
 - <http://eleccomm.ieee.org/email-aup.shtml>
 - Inform user of what he can do with IEEE email
 - Inform user of what IEEE will provide
 - Does not accept responsibility of actions resulting from user email
 - Does not guarantee privacy of IEEE computers and networks
 - Examples of acceptable and unacceptable use

University policies

- Link to general policies and through the FERPA link the information security policy
 - http://www.cites.uiuc.edu/edtech/policies_guideli

Policy Languages

- Express security policies in a precise way
- A continuum of policy languages
 - English Policies
 - May be legally precise. Used as basis for legal action.
 - May be written imprecisely just to give real users a sense of the policy
 - High-level languages
 - Policy constraints expressed abstractly
 - Low-level languages
 - Policy constraints expressed in terms of program options, input, or specific characteristics of entities on system

High-Level Policy Languages

- Constraints expressed independent of enforcement mechanism
- Constraints restrict entities, actions
- Constraints expressed unambiguously
 - Requires a precise language, usually a mathematical, logical, or programming-like language
- Examples
 - Java constraint language – described in CS:A&S
 - DTEL type enforcement language
 - SAML <http://en.wikipedia.org/wiki/SAML>
 - IETF Policy models <ftp://ftp.rfc-editor.org/in-notes/rfc3585.txt>

DTEL – Domain Type Enforcement Language

- Basis: access can be constrained by types
- Combines elements of low-level, high-level policy languages
 - Implementation-level constructs express constraints in terms of language types
 - Constructs do not express arguments or inputs to specific system commands
- Used in Sidewinder firewalls
- Details of DTEL in <http://citeseer.ist.psu.edu/cache/papers/cs/16179/http:zSzzS>
- Type enforcement policies resurfacing in SE Linux

Example

- Goal: users cannot write to system binaries
- Subjects in administrative domain can
 - User must authenticate to enter that domain
- Subjects belong to domains:
 - *d_user* ordinary users
 - *d_admin* administrative users
 - *d_login* for login
 - *d_daemon* system daemons

Types

- Object types:
 - *t_sysbin* executable system files
 - *t_readable* readable files
 - *t_writable* writable files
 - *t_dte* data used by enforcement mechanisms
 - *t_generic* data generated from user processes
- For example, treat these as partitions
 - In practice, files can be readable and writable; ignore this for the example

Domain Representation

- Sequence
 - First component is list of programs that start in the domain
 - Other components describe rights subject in domain has over objects of a type

`(crwd->t_writable)`

means subject can create, read, write, and list (search) any object of type `t_writable`

d_daemon Domain

```
domain d_daemon = (/sbin/init),  
    (crwd->t_writable),  
    (rd->t_generic, t_readable, t_dte),  
    (rxd->t_sysbin),  
    (auto->d_login);
```

- Compromising subject in *d_daemon* domain does not enable attacker to alter system files
 - Subjects here have no write access
- When `/sbin/init` invokes login program, login program transitions into *d_login* domain

d_admin Domain

```
domain d_admin =
    (/usr/bin/sh, /usr/bin/csh, /usr/bin/ksh),
    (crwxd->t_generic),
    (crwxd->t_readable, t_writable, t_dte,
     t_sysbin),
    (sigtstp->d_daemon);
```

- *sigtstp* allows subjects to suspend processes in *d_daemon* domain
- Admin users use a standard command interpreter

d_user Domain

```
domain d_user =  
    (/usr/bin/sh, /usr/bin/csh, /usr/bin/ksh),  
    (crwd->t_generic),  
    (rxd->t_sysbin),  
    (crwd->t_writable),  
    (rd->t_readable, t_dte);
```

- No auto component as no user commands transition out of it
- Users cannot write to system binaries

d_login Domain

```
domain d_login =  
    (/usr/bin/login),  
    (crwd->t_writable),  
    (rd->t_readable, t_generic, t_dte),  
    setauth,  
    (exec->d_user, d_admin);
```

- Cannot execute anything except the transition
 - Only /usr/bin/login in this domain
- *setauth* enables subject to change UID
- *exec* access to *d_user*, *d_admin* domains

Set Up

```
initial_domain = d_daemon;
```

- System starts in *d_daemon* domain

```
assign -r t_generic /;
```

```
assign -r t_writable /usr/var, /dev, /tmp;
```

```
assign -r t_readable /etc;
```

```
assign -r -s dte_t /dte;
```

```
assign -r -s t_sysbin /sbin, /bin,  
                /usr/bin, /usr/sbin;
```

- These assign initial types to objects
- `-r` recursively assigns type
- `-s` binds type to name of object (delete it, recreate it, still of given type)

Add Log Type

- Goal: users can't modify system logs; only subjects in *d_admin*, new *d_log* domains can

```
type t_readable, t_writable, t_sysbin,  
      t_dte, t_generic, t_log;
```

- New type *t_log*

```
domain d_log =  
  (/usr/sbin/syslogd),  
  (crwd->t_log),  
  (rwd->t_writable),  
  (rd->t_generic, t_readable);
```

- New domain *d_log*

Fix Domain and Set-Up

```
domain d_daemon =  
    (/sbin/init),  
    (crwd->t_writable),  
    (rxd->t_readable),  
    (rd->t_generic, t_dte, t_sysbin),  
    (auto->d_login, d_log);
```

- Subject in *d_daemon* can invoke logging process
- Can log, but not execute anything

```
assign -r t_log /usr/var/log;  
assign t_writable /usr/var/log/wtmp,  
    /usr/var/log/utmp;
```

- Set type of logs

Low-Level Policy Languages

- Set of inputs or arguments to commands
 - Check or set constraints on system
- Low level of abstraction
 - Need details of system, commands
- Can think of as specific configuration languages.
Generally very closely tied to an application.
- Examples:
 - Xhost
 - Unix file system access commands
 - Tripwire integrity databases
 - IOS router CLI – Could argue IOS ACL's are higher level

Example: X Window System

- UNIX X11 Windowing System
- Access to X11 display controlled by list
 - List says what hosts allowed, disallowed access
`xhost +groucho -chico`
- Connections from host groucho allowed
- Connections from host chico not allowed

Example: tripwire

- File scanner that reports changes to file system and file attributes
 - *tw.config* describes what may change
 - `/usr/mab/tripwire +gimnpsu012345678-a`
 - Check everything but time of last access (“-a”)
 - Database holds previous values of attributes

Example Database Record

```
/usr/mab/tripwire/README 0 ..../. 100600 45763 1
917 10 33242 .gtPvf .gtPvY .gtPvY 0
.ZD4cc0Wr8i21ZKaI..LUOr3
.0fwo5:hf4e4.8TAqd0V4ubv ?..... ...9b3
1M4GX01xbGIX0oVuGo1h15z3 ?:
Y9jfa04rdzM1q:egt1APgHk ?.
Eb9yo.2zkEh1XKovX1:d0wF0kfAvC ?
1M4GX01xbGIX2947jdyrior38h15z3 0
```

- file name, version, bitmask for attributes, mode, inode number, number of links, UID, GID, size, times of creation, last modification, last access, cryptographic checksums

Comments

- System administrators not expected to edit database to set attributes properly
- Checking for changes with tripwire is easy
 - Just run once to create the database, run again to check
- Checking for conformance to policy is harder
 - Need to either edit database file, or (better) set system up to conform to policy, then run tripwire to construct database

Key Points

- Policies specify Why
 - Mechanisms specify how
- Range of security policies
 - From High level and imprecise to formal and precise