Lecture 6, April 10, 2026

Trust

Administrator installs patch

1. Trusts patch came from vendor, not tampered with in transit
2. Trusts vendor tested patch thoroughly
3. Trusts vendor's test environment corresponds to local environment
4. Trusts patch is installed correctly

Trust in Formal Verification

- Gives formal mathematical proof that given input *i*, program *P* produces output *o* as specified
- Suppose a security-related program *S* formally verified to work with operating system *O*
- What are the assumptions?

Trust in Formal Methods

1. Proof has no errors
   - Bugs in automated theorem provers
2. Preconditions hold in environment in which *S* is to be used
3. *S* transformed into executable *S′* whose actions follow source code
   - Compiler bugs, linker/loader/library problems
4. Hardware executes *S′* as intended
   - Hardware bugs (Pentium `f00f` bug, for example)

Types of Access Control

- Discretionary Access Control (DAC, IBAC)
  - Individual user sets access control mechanism to allow or deny access to an object
- Mandatory Access Control (MAC)
  - System mechanism controls access to object, and individual cannot alter that access
- Originator Controlled Access Control (ORCON, ORGCON)
  - Originator (creator) of information controls who can access information

Policy Languages

- Express security policies in a precise way
- 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

Example: Ponder

- Security and management policy specification language
- Handles many types of policies
  - Authorization policies
  - Delegation policies
  - Information filtering policies
  - Obligation policies
  - Refrain policies

Entities

- Organized into hierarchical domains
- Network administrators
  - *Domain* is /NetAdmins
  - Subdomain for net admin trainees is
  - /NetAdmins/Trainees
- Routers in LAN
  - Domain is /localnet
  - Subdomain that is a testbed for routers is
  - /localnet/testbed/routers

Authorization Policies

- Allowed actions: netadmins can enable, disable, reconfigure, view configuration of routers

```
inst auth+ switchAdmin {
      subject /NetAdmins;
      target  /localnetwork/routers;
      action  enable(), disable(), reconfig(), dumpconfig();
}
```

Authorization Policies

- Disallowed actions: trainees cannot test performance between 8AM and 5PM

```
inst auth- testOps {
    subject /NetEngineers/trainees;
    target  /localnetwork/routers;
    action  testperformance();
    when    Time.between("0800", "1700");
}
```

Delegation Policies

- Delegated rights: net admins delegate to net engineers the right to enable, disable, reconfigure routers on the router testbed

```
inst deleg+ (switchAdmin) delegSwitchAdmin {
    grantee  /NetEngineers;
    target   /localnetwork/testNetwork/routers;
    action   enable(), disable(), reconfig();
    valid    Time.duration(8);
}
```

Information Filtering Policies

- Control information flow: net admins can dump everything from routers between 8PM and 5AM, and config info anytime

```
inst auth+ switchOpsFilter {
    subject  /NetAdmins;
    target   /localnetwork/routers;
    action   dumpconfig(what)
             { in partial = "config"; }
            if (Time.between("2000", "0500")){
                 in partial = "all"; }
}
```

Refrain Policies

- Like authorization denial policies, but enforced by the *subjects*: net engineers cannot send test results to net developers while testing in progress

```
inst refrain testSwitchOps {
    subject  s=/NetEngineers;
    target   /NetDevelopers;
    action   sendTestResults();
    when    s.teststate="in progress"
}
```

Obligation Policies

- Must take actions when events occur: on 3rd login failure, net security admins will disable account and log event

```
inst oblig loginFailure {
    on        loginfail(userid, 3);
    subject   s=/NetAdmins/SecAdmins;
    target    t=/NetAdmins/users ^ (userid);
    do        t.disable() -> s.log(userid);
}
```

Example

- Policy: separation of duty requires 2 different members of Accounting approve check

```
inst auth+ separationOfDuty {
    subject  s=/Accountants;
    target   t=checks;
    action   approve(), issue();
    when     s.id <> t.issuerid;
}
```

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

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:eqt1APgHk ?.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

Example English Policy

- Computer security policy for academic institution
  - Institution has multiple campuses, administered from central office
  - Each campus has its own administration, and unique aspects and needs
- Deals with electronic communications
  - Policy
  - User Advisories
  - Implementation at University of California Davis

Background

- University of California
  - 10 campuses (including UC Davis), each run by a Chancellor
  - UC Office of the President (UCOP) runs system, and is run by President of University of California
- UCOP issues policies that apply to all campuses
- Campuses implement the policy in a manner consistent with directions from UCOP

Electronic Communications Policy

- Begins with purpose, to whom policy applies
  - Includes email, video, voice, other means
  - Not to printed copies of communications
  - Not to Dept. of Energy labs that UC manages, or to Dept. of Energy employees
- Gives general implementation guidelines

Use of Electronic Communications

- University does *not* want to deal with contents of these!
  - But all communications relating to University administration are public records
  - Others may be too
- Allowable users
  - Faculty, staff, students, others associated with UC
  - Others authorized by the Chancellors or UCOP
  - Others participating in programs UC sponsors

Allowable Uses

- University business
  - Classes, research, *etc.*
- Incidental personal use OK
  - But can’t interfere with other uses
- Anonymous communications OK
  - But can’t use a false identity

Non-Allowable Uses

- Endorsements not OK
- Running personal businesses not OJK
- Illegal activities not OK
  - Must respect intellectual property laws, US DMCA
- Violating University of campus policies or rules not OK
- Users can’t put “excessive strain” on resources
  - No spamming, DoD or DDoS attacks

Privacy, Confidentiality

- General rule: respected the same way as is for paper
- Cannot read or disclose without permission of holder, except in specific circumstances
- To do so requires written permission of:
  - A designated Vice Chancellor (campus)
  - A Senior Vice President, Business and Finance (UCOP)

Privacy, Confidentiality

- Written permission not required for:
  - Subpoena or search warrant
  - Emergency
    - But must obtain approval as soon as possible afterwards
  - In all these cases, must notify those affected by the disclosure that the disclosure occurred, and why

Limits of Privacy

- Electronic communications that are public records will not be confidential
- Electronic communications may be on backups
- Electronic communications may be seen during routine system monitoring, etc.
  - Admins instructed to respect privacy, but *will* report “improper governmental activity”

Security Services, Practices

- Routine monitoring
- Need for authentication
- Need for authorization
- Need for recovery mechanisms
- Need for audit mechanisms
- Other mechanisms to enforce University policy

User Advisories

- These are less formal, give guidelines for the use of electronic communications
  - Show courtesy and consideration as in non-electronic communications
  - Laws about privacy in electronic communications are not as mature as laws about privacy in other areas
  - University provides neither encryption nor authentication
    - Easy to falsify sender

UC Davis Implementation

- Acceptable Use Policy
  - Incorporates the UCD Principles of Community
  - Requires respect of rights of others when using electronic communications
  - Use encouraged for education, university business, university-related activities

UC Davis Implementation

- UC Davis specific details
  - Only Chancellor-approved charitable activities may use these resources
  - Cannot be used to create hostile environment
    - This includes violating obscenity laws
  - Incidental personal use OK under conditions given in Electronic Communications Policy

UC Davis Implementation

- Unacceptable conduct
  - Not protecting passwords for University resources
  - Not respecting copyrights, licenses
  - Violating integrity of these resources
  - Creating malicious logic (worms, viruses, *etc*.)
    - Allowed if done as part of an academic research or instruction program supervised by academic personnel; and
    - It does not compromise the University's electric communication resource

UC Davis Implementation

- Allowed users
  - UCD students, staff, faculty
  - Other UCD academic appointees and affiliated people
    - Such as postdocs and visiting scholars
- People leaving
  - Forwarding email allowed
  - Recipient must agree to return to the University any email about University business

Exceptions Allowing Disclosure

- Required by law;
- Reliable evidence of violation of law, University policies;
- Failure to do so may result in:
  - Significant harm
  - Loss of significant evidence of violations;
  - Significant liability to UC or its community;
- Not doing so hampers University meeting administrative, teaching obligations

Confidentiality Policy

- Goal: prevent the unauthorized disclosure of information
  - Deals with information flow
  - Integrity incidental
- Multi-level security models are best-known examples
  - Bell-LaPadula Model basis for many, or most, of these

Bell-LaPadula Model, Step 1

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist are called *security clearance* $L(s)$ for subjects and *security classification* $L(o)$ for objects

Example

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaley can only read Telephone Lists

Reading Information

- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition (Step 1)
  - Subject *s* can read object *o* iff, $L(o) \leq L(s)$ and *s* has permission to read *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 1)
  - Subject *s* can write object *o* iff $L(s) \leq L(o)$ and *s* has permission to write *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

Basic Security Theorem, Step 1

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 1, and the *-property, step 1, then every state of the system is secure
  - Proof: induct on the number of transitions

Bell-LaPadula Model, Step 2

- Expand notion of security level to include categories
- Security level is (*clearance*, *category set*)
- Examples
  - (Top Secret, { NUC, EUR, ASI })
  - (Confidential, { EUR, ASI })
  - (Secret, { NUC, ASI })

Levels and Lattices

- (A, C) *dom* (A', C') iff $A' \leq A$ and $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) $\neg$*dom* (Confidential, {EUR})
- Let C be set of classifications, K set of categories. Set of security levels $L = C \times K$, *dom* form lattice
  - $lub(L) = (max(A), C)$
  - $glb(L) = (min(A), \varnothing)$

Levels and Ordering

- Security levels partially ordered
  - Any pair of security levels may (or may not) be related by *dom*
- "dominates" serves the role of "greater than" in step 1
  - "greater than" is a total ordering, though

Reading Information

- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition (Step 2)
  - Subject *s* can read object *o* iff *L*(*s*) *dom L*(*o*) and *s* has permission to read *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 2)
  - Subject *s* can write object *o* iff *L*(*o*) *dom L*(*s*) and *s* has permission to write *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

Basic Security Theorem, Step 2

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 2, and the *-property, step 2, then every state of the system is secure
  - Proof: induct on the number of transitions
  - In actual Basic Security Theorem, discretionary access control treated as third property, and simple security property and *-property phrased to eliminate discretionary part of the definitions — but simpler to express the way done here.