Lecture 7, April 13

ECS 235B, Foundations of Computer and Information Security

Spring Quarter 2026

Requirements of Integrity Policies

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a non-production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

Principles of Operation

- *Separation of duty*: if two or more steps are required to perform a critical function, at least two different people should perform the steps
- *Separation of function*: different entities should perform different functions
- *Auditing*: recording enough information to ensure the abilities to both recover and determine accountability

Biba Integrity Model

Basis for all 3 models:

- Set of subjects S, objects O, integrity levels I, relation $\leq \subseteq I \times I$ holding when second dominates first
- $min: I \times I \rightarrow I$ returns lesser of integrity levels
- $i: S \cup O \rightarrow I$ gives integrity level of entity
- $\underline{r}: S \times O$ means $s \in S$ can read $o \in O$
- $\underline{w}$, $\underline{x}$ defined similarly

Intuition for Integrity Levels

- The higher the level, the more confidence
  - That a program will execute correctly
  - That data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important point: *integrity levels are **not** security levels*

Information Transfer Path

- An *information transfer path* is a sequence of objects o_1, ..., o_{n+1} and corresponding sequence of subjects s_1, ..., s_n such that s_i r o_i and s_i w o_{i+1} for all i, $1 \leq i \leq n$.
- Idea: information can flow from o_1 to o_{n+1} along this path by successive reads and writes

Low-Water-Mark Policy

- Idea: when s reads o, $i(s) = min(i(s), i(o))$; s can only write objects at lower levels
- Rules
  1. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
  2. If $s \in S$ reads $o \in O$, then $i'(s) = min(i(s), i(o))$, where $i'(s)$ is the subject's integrity level after the read.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

Information Flow and Model

- If information transfer path from $o_1 \in O$ to $o_{n+1} \in O$, enforcement of low-water-mark policy requires $i(o_{n+1}) \leq i(o_1)$ for all $n > 1$.
  - Idea of proof: Assume information transfer path exists between o_1 and o_{n+1}. Assume that each read and write was performed in the order of the indices of the vertices. By induction, the integrity level for each subject is the minimum of the integrity levels for all objects preceding it in path, so $i(s_n) \leq i(o_1)$. As nth write succeeds, $i(o_{n+1}) \leq i(s_n)$. Hence $i(o_{n+1}) \leq i(o_1)$.

Problems

- Subjects' integrity levels do not increase as system runs
  - Soon no subject will be able to access objects at high integrity levels
- Alternative: change object levels rather than subject levels
  - Soon all objects will be at the lowest integrity level
- Crux of problem is model prevents indirect modification
  - Because subject levels lowered when subject reads from low-integrity object

Ring Policy

- Idea: subject integrity levels static
- Rules
  1. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
  2. Any subject can read any object.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.
- Difference with low-water-mark policy is any subject can read any object
- Eliminates indirect modification problem
- Same information flow result holds

Strict Integrity Policy

- Dual of Bell-LaPadula model
  1. $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
  2. $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
  3. $s_1 \in S$ can execute $s_2 \in S$ iff $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model
- Information flow result holds
  - Different proof, though
- Term "Biba Model" refers to this

LOCUS and Biba

- Goal: prevent untrusted software from altering data or other software
- Approach: make levels of trust explicit
  - *credibility rating* based on estimate of software's trustworthiness (0 untrusted, *n* highly trusted)
  - *trusted file systems* contain software with a single credibility level
  - Process has *risk level* or highest credibility level at which process can execute
  - Must use *run-untrusted* command to run software at lower credibility level

Clark-Wilson Integrity Model

- Integrity defined by a set of constraints
  - Data in a *consistent* or valid state when it satisfies these
- Example: Bank
  - *D* today's deposits, *W* withdrawals, *YB* yesterday's balance, *TB* today's balance
  - Integrity constraint: $TB = D + YB - W$
- *Well-formed transaction* moves system from one consistent state to another
- Issue: who examines, certifies transactions done correctly?

Entities

- CDIs: constrained data items
  - Data subject to integrity controls
- UDIs: unconstrained data items
  - Data not subject to integrity controls
- IVPs: integrity verification procedures
  - Procedures that test the CDIs conform to the integrity constraints
- TPs: transaction procedures
  - Procedures that take the system from one valid state to another

Certification Rules 1 and 2

CR1 When any IVP is run, it must ensure all CDIs are in a valid state

CR2 For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state

- Defines relation *certified* that associates a set of CDIs with a particular TP
- Example: TP balance, CDIs accounts, in bank example

Enforcement Rules 1 and 2

ER1 The system must maintain the certified relations and must ensure that only TPs certified to run on a CDI manipulate that CDI.

ER2 The system must associate a user with each TP and set of CDIs. The TP may access those CDIs on behalf of the associated user. The TP cannot access that CDI on behalf of a user not associated with that TP and CDI.

- System must maintain, enforce certified relation
- System must also restrict access based on user ID (*allowed* relation)

Users and Rules

CR3 The allowed relations must meet the requirements imposed by the principle of separation of duty.

ER3 The system must authenticate each user attempting to execute a TP

- Type of authentication undefined, and depends on the instantiation
- Authentication *not* required before use of the system, but *is* required before manipulation of CDIs (requires using TPs)

Logging

CR4 All TPs must append enough information to reconstruct the operation to an append-only CDI.

- This CDI is the log
- Auditor needs to be able to determine what happened during reviews of transactions

Handling Untrusted Input

CR5 Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.

- In bank, numbers entered at keyboard are UDIs, so cannot be input to TPs. TPs must validate numbers (to make them a CDI) before using them; if validation fails, TP rejects UDI

Separation of Duty In Model

ER4 Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

- Enforces separation of duty with respect to certified and allowed relations

Comparison With Requirements

1. Users can't certify TPs, so CR5 and ER4 enforce this
2. Procedural, so model doesn't directly cover it; but special process corresponds to using TP
   - No technical controls can prevent programmer from developing program on production system; usual control is to delete software tools
3. TP does the installation, trusted personnel do certification

Comparison With Requirements

4. CR4 provides logging; ER3 authenticates trusted personnel doing installation; CR5, ER4 control installation procedure
   - New program UDI before certification, CDI (and TP) after
5. Log is CDI, so appropriate TP can provide managers, auditors access
   - Access to state handled similarly

Comparison to Biba

- Biba
  - No notion of certification rules; trusted subjects ensure actions obey rules
  - Untrusted data examined before being made trusted
- Clark-Wilson
  - Explicit requirements that *actions* must meet
  - Trusted entity must certify *method* to upgrade untrusted data (and not certify the data itself)

UNIX Implementation

- Considered “allowed” relation

  (*user*, *TP*, { *CDI set* })

- Each TP is owned by a different user
  - These “users” are actually locked accounts, so no real users can log into them; but this provides each TP a unique UID for controlling access rights
  - TP is setuid to that user
- Each TP’s group contains set of users authorized to execute TP
- Each TP is executable by group, not by world

CDI Arrangement

- CDIs owned by *root* or some other unique user
  - Again, no logins to that user's account allowed
- CDI's group contains users of TPs allowed to manipulate CDI
- Now each TP can manipulate CDIs for single user

Examples

- Access to CDI constrained by user
  - In “allowed” triple, *TP* can be any TP
  - Put CDIs in a group containing all users authorized to modify CDI
- Access to CDI constrained by TP
  - In “allowed” triple, *user* can be any user
  - CDIs allow access to the owner, the user owning the TP
  - Make the TP world executable

Problems

- 2 different users cannot use same copy of TP to access 2 different CDIs
  - Need 2 separate copies of TP (one for each user and CDI set)
- TPs are setuid programs
  - As these change privileges, want to minimize their number
- *root* can assume identity of users owning TPs, and so cannot be separated from certifiers
  - No way to overcome this without changing nature of *root*

Trust Models

- Integrity models state conditions under which changes preserve a set of properties
  - So deal with the *preservation* of trustworthiness
- Trust models deal with confidence one can have in the initial values or settings
  - So deal with the *initial* evaluation of whether data can be trusted

Definition of Trust

Anna *trusts* Bill if Anna believes, with a level of subjective probability, that Bill will perform a particular action, both before the action can be monitored (or independently of the capacity of being able to monitor it) and in a context in which it affects Anna's own action.

- Includes subjective nature of trust
- Captures idea that trust comes from a belief in what we do not monitor
- Leads to transitivity of trust

Transitivity of Trust

Transitivity of trust: if Anna trusts Bill and Bill trusts Carole, then Anna trusts Carole

- Not always; depends on Anna's assessment of B's judgment
- *Conditional transitivity of trust*: A trusts C when
  - B recommends C to A;
  - A trusts B's recommendations;
  - A can make judgments about B's recommendations; and
  - Based on B's recommendation, A may trust C less than B does
- *Direct trust*: A trusts C because of A's observations and interactions
- *Indirect trust*: A trusts C because A accepts B's recommendation

Types of Beliefs Underlying Trust

- *Competence*: A believes B competent to aid A in reaching goal
- *Disposition*: A believes B will actually do what A needs to reach goal
- *Dependence*: A believes she needs what B will do, depends on what B will do, or it's better to rely on B than not
- *Fulfillment*: A believes goal will be reached
- *Willingness*: A believes B has decided to do what A wants

Types of Beliefs Underlying Trust

- *Persistence*: A believes B will not change B's mind before doing what A wants
- *Self-confidence*: A believes that B knows B can take the action A wants
- *Majority behavior*: A's belief that most people from B's community are trustworthy
- *Prudence*: Not trusting B poses unacceptable risk to A
- *Pragmatism*: A's current interests best served by trusting B

Trust Management

- Use a language to express relationships about trust, allowing us to reason about trust
  - Evaluation mechanisms take data, trust relationships and provide a measure of trust about the entity or whether an action should or should not be taken
- Two basic forms
  - Policy-based trust management
  - Reputation-based trust management

Policy-Based Trust Management

- Credentials instantiate policy rules
  - Credentials are data, so they too may be input to the rules
  - Trusted third parties often vouch for credentials
- Policy rules expressed in a policy language
  - Different languages for different goals
  - Expressiveness of language determines the policies it can express

Example: Keynote

- Basic units
  - Assertions: describe actions allowed to possessors of credentials
    - Policy: statements about policy
    - Credential: statements about credentials
  - Action environment: attributes describing action associated with credentials
- Evaluator: takes set of policy assertions, set of credentials, action environment and determines if proposed action is consistent with policy

Example

- Consider email domain: policy assertion authorizes holder of mastercred for all actions:

```
Authorizer: "POLICY"
Licensees: "mastercred"
```

- Credential assertion:

```
KeyNote-Version: 2
Local-Constants: Alice="cred1234", Bob="credABCD"
Authorizer: "authcred"
Licensees: Alice || Bob
Conditions: (app_domain == "RFC822-EMAIL") &&
              (address ~= "^.*@keynote\\.ucdavis\\.edu$")
Signature: "signed"
```

- Compliance Value Set: `{ "_MIN_TRUST", "_MAX_TRUST" }`

Example: Results

- Evaluator given action environment:

```
_ACTION_AUTHORIZERS = "Alice"
app_domain = "RFC822-EMAIL"
address = "snoopy@keynote.ucdavis.edu"
```

  it satisfies policy, so returns `_MAX_TRUST`

- Evaluator given action environment:

```
_ACTION_AUTHORIZERS = "Bob"
app_domain = "RFC822-EMAIL"
address = "opus@admin.ucdavis.edu"
```

  it does not satisfy policy, so returns `_MIN_TRUST`

Example 2

- Consider separation of duty: policy assertion delegates authority to pay invoices to entity with credential “fundmgrcred”:

```
Authorizer: "POLICY"
Licensee: "fundmgtcred"
Conditions: (app_domain == "INVOICE" && @dollars < 10000)
```

- Credential assertion (requires 2 signatures on any expenditure):

```
KeyNote-Version: 2
Comment: This credential specifies a spending policy
Authorizer: "authcred"
Licensees: 2-of("cred1", "cred2", "cred3", "cred4", "cred5")
Conditions: (app_domain=="INVOICE")       # note nested clauses
                -> { (@dollars) < 2500) -> "Approve";
                     (@dollars < 7500) -> "ApproveAndLog"; };
Signature: "signed"
```

- Compliance Value Set: `{"Reject", "ApproveAndLog", "Approve" }`

Example 2: Results

- Evaluator given action environment:

```
_ACTION_AUTHORIZERS = "cred1,cred4"
app_domain = "INVOICE"
dollars = "1000"
```

  it satisfies first clause of condition, and so policy, so returns `Approve`

- Evaluator given action environment:

```
_ACTION_AUTHORIZERS = "cred1"
app_domain = "INVOICE"
dollars = "1500"
```

  it does not satisfy policy as too few Licensees, so returns `Reject`

Example 2: Results

- Evaluator given action environment:

```
_ACTION_AUTHORIZERS = "cred1,cred2"
app_domain = "INVOICE"
dollars = "3541"
```

  it satisfies second clause of condition, and so policy, so returns `ApproveAndLog`

- Evaluator given action environment:

```
_ACTION_AUTHORIZERS = "cred1,cred5"
app_domain = "INVOICE"
dollars = "8000"
```

  it does not satisfy policy as amount too large, so returns `Reject`

Reputation-Based Trust Management

- Use past behavior, information from other sources, to determine whether to trust an entity
- Some models distinguish between direct, indirect trust
- Trust category, trust values, agent's identification form *reputation*
- *Recommendation* is trust information containing at least 1 reputation
- Systems use many different types of metrics
  - Statistical models
  - Belief models (probabilities may not sum to 1, due to uncertainty in belief)
  - Fuzzy models (reasoning involves degrees of trustworthiness)

Example 1

- Direct trust: –1 (untrustworthy), 1 to 4 (degrees of trust, increasing), 0 (cannot make trust judgment)
- Indirect trust: –1, 0 (same as for direct trust), 1 to 4 (how close the judgment of recommender is to the entity being recommended to)
- Formula:

$$t(T, P) = tv(T) \prod_{i=1}^{n} \frac{tv(R_i)}{4}$$

where T is entity of concern, P trust path, $tv(x)$ trust value of x, $t(T,P)$ overall trust in T based on trust path P

Example 1

- Amy wants Boris' recommendation about Danny so she asks him
  - Amy trusts Boris' recommendations with trust value 2 as his judgment is somewhat close to hers
- Boris doesn't know Danny, so he asks Carole
  - He trusts her recommendations with trust value 3
- Carole believes Danny is above average programmer, so she replies with a recommendation of 3
- Boris adds this to the end of the recommendation
- *P* is Amy—Boris—Carole—Danny, so R_1 = Boris, R_2 = Carole, *T* = Danny, so

$$T(\text{“Danny”}, P) = \frac{2}{4} \times \frac{3}{4} \times 3 = 1.125$$