ECS 235B Module 23
Clark-Wilson Model
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: $D + YB = W$

• *Well-formed transaction* move 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 \textit{not} required before use of the system, but \textit{is} required before manipulation of CDIs (requires using TPs)
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

\((user, TP, \{ CDI \text{ set } \})\)

• 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*