Lecture 6
October 9, 2023

Principles of Declassification

- Principle of Semantic Consistency
  - As long as semantics of components that do not do declassification do not change, the components can be altered without affecting security
- Principle of Occlusion
  - A declassification operation cannot conceal an *improper* declassification
- Principle of Conservativity
  - Absent any declassification, the system is secure
- Principle of Monotonicity of Release
  - When declassification is performed in an authorized manner by authorized subjects, the system remains secure

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 *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*

Cryptosystem

- Quintuple ($\mathcal{E}$, $\mathcal{D}$, $\mathcal{M}$, $\mathcal{K}$, $\mathcal{C}$)
  - $\mathcal{M}$ set of plaintexts
  - $\mathcal{K}$ set of keys
  - $\mathcal{C}$ set of ciphertexts
  - $\mathcal{E}$ set of encryption functions *e*: $\mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$
  - $\mathcal{D}$ set of decryption functions *d*: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$

Example

- Example: Cæsar cipher
  - $\mathcal{M}$ = { sequences of letters }
  - $\mathcal{K}$ = { i | i is an integer and $0 \le i \le 25$ }
  - $\mathcal{E}$ = { E_k | $k \in \mathcal{K}$ and for all letters m, $E_k(m) = (m + k) \bmod 26$ }
  - $\mathcal{D}$ = { D_k | $k \in \mathcal{K}$ and for all letters c, $D_k(c) = (26 + c - k) \bmod 26$ }
  - $\mathcal{C} = \mathcal{M}$

Attacks

- Opponent whose goal is to break cryptosystem is the *adversary*
  - Assume adversary knows algorithm used, but not key
- Three types of attacks:
  - *ciphertext only*: adversary has only ciphertext; goal is to find plaintext, possibly key
  - *known plaintext*: adversary has ciphertext, corresponding plaintext; goal is to find key
  - *chosen plaintext*: adversary may supply plaintexts and obtain corresponding ciphertext; goal is to find key

Basis for Attacks

- Mathematical attacks
  - Based on analysis of underlying mathematics
- Statistical attacks
  - Make assumptions about the distribution of letters, pairs of letters (digrams), triplets of letters (trigrams), *etc.*
    - Called *models of the language*
  - Examine ciphertext, correlate properties with the assumptions.

Symmetric Cryptography

- Sender, receiver share common key
  - Keys may be the same, or trivial to derive from one another
  - Sometimes called *secret key cryptography*
- Two basic types
  - Transposition ciphers
  - Substitution ciphers
  - Combinations are called *product ciphers*

Transposition Cipher

- Rearrange letters in plaintext to produce ciphertext
- Example (Rail-Fence Cipher)
  - Plaintext is `HELLO WORLD`
  - Rearrange as

```
HLOOL
ELWRD
```

  - Ciphertext is `HLOOL ELWRD`

Attacking the Cipher

- Anagramming
  - If 1-gram frequencies match English frequencies, but other *n*-gram frequencies do not, probably transposition
  - Rearrange letters to form *n*-grams with highest frequencies

Example

- Ciphertext: `HLOOLELWRD`
- Frequencies of 2-grams beginning with `H`
  - `HE` 0.0305
  - `HO` 0.0043
  - `HL`, `HW`, `HR`, `HD` < 0.0010
- Frequencies of 2-grams ending in `H`
  - `WH` 0.0026
  - `EH`, `LH`, `OH`, `RH`, `DH` ≤ 0.0002
- Implies `E` follows `H`

Example

- Arrange so the H and E are adjacent

```
HE
LL
OW
OR
LD
```

- Read across, then down, to get original plaintext

Substitution Ciphers

- Change characters in plaintext to produce ciphertext
- Example (Caesar cipher)
  - Plaintext is `HELLO WORLD`
  - Change each letter to the third letter following it (`X` goes to `A`, `Y` to `B`, `Z` to `C`)
    - Key is 3, usually written as letter '`D`'
  - Ciphertext is `KHOOR ZRUOG`