Some useful Information

Chinese Wall Rules

**CW-Simple Security Condition**: S can read O if and only if any of the following holds.
- There is an object O' such that S has accessed O' and CD(O') = CD(O).
- For all objects O', O' ∈ PR(S) ⇒ COI(O') ≠ COI(O).
- O is a sanitized object.

(O' ∈ PR(S) indicates O' has been previously read by s)

**CW-Property**: A subject S may write to an object O if and only if both of the following conditions hold.
- The CW-simple security condition permits S to read O.
- For all unsanitized objects O', S can read O' ⇒ CD(O') = CD(O).

Clark-Wilson Certification and Enforcement Rules

**Certification rule 1 (CR1)**: When any IVP is run, it must ensure that all CDIs are in a valid state.

**Certification rule 2 (CR2)**: For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state.

**Enforcement rule 1 (ER1)**: The system must maintain the certification rules, and must ensure that only TPs certified to run on a CDI manipulate that CDI.

**Enforcement rule 2 (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. If the user is not associated with a particular TP and CDI, then the TP cannot access that CDI on behalf of that user.

**Certification rule 3 (CR3)**: The allowed relations must meet the requirements imposed by the principle of separation of duty.

**Enforcement rule 3 (ER3)**: The system must authenticate each user attempting to execute a TP.

**Certification rule 4 (CR4)**: All TPs must append enough information to reconstruct the operation to an append-only CDI.

**Certification rule 5 (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.

**Enforcement rule 4 (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.

Core RBAC

Permissions = 2Operations x Objects

UA ⊆ Users x Roles

PA ⊆ Permissions x Roles

assigned_users: Roles → 2Users

assigned_permissions: Roles → 2Permissions

Op(p): set of operations associated with permission p

Ob(p): set of objects associated with permission p

user_sessions: Users → 2Sessions

session_user: Sessions → Users

session_roles: Sessions → 2Roles

avail_sessions_perms: Sessions → 2Permissions

RBAC with general Role hierarchy

authorized_users: Roles→ 2Users

- authorized_users(r) = \{u | r ≥ r' & (r', u) ∈ UA\}

(Notice that for any role r ≥ r – so all role assigned to r are also authorized to r)

authorized_permissions: Roles→ 2Permissions

- authorized_permissions(r) = \{p | r ≥ r' & (p, r') ∈ PA\}

RH ⊆ Roles x Roles is a partial order, called the inheritance relation & written as ≥.

(r1 ≥ r2) → authorized_users(r1) ⊆ authorized_users(r2) &

authorized_permissions(r2) ⊆ authorized_permissions(r1)

Static SoD

SSD ⊆ 2Roles x N

In absence of hierarchy

Collection of pairs (RS, n) where RS is a role set, n ≥ 2;

for all (RS, n) ∈ SSD, for all t ∈ RS: |t| ≥ n → \(\cap_{r \in t} \text{assigned_users}(r) = ∅\)

In presence of hierarchy

Collection of pairs (RS, n) where RS is a role set, n ≥ 2;

for all (RS, n) ∈ SSD, for all t ∈ RS: |t| ≥ n → \(\cap_{r \in t} \text{authorized_users}(r) = ∅\)