CS590U
Access Control: Theory and Practice

Lecture 14 (March 2)
RBAC: Constraint Generation
SSoD Policies

- The SoD principle: the collaboration of multiple users is needed to perform some sensitive tasks
- Static enforcement of SoD: multiple users together have all permissions to perform these tasks
- SSoD policies
  - \( \text{ssod}\{\{p1,p2,p3,p4\}, 3\} \) means that 3 users are required to cover all permissions in \( \{p1,p2,p3,p4\} \), i.e., no 2 users have all permissions in \( \{p1,p2,p3,p4\} \)
- Checking whether an RBAC state is safe wrt. an SSoD policy is coNP-complete.
SMER Constraints

- \( \text{smer} \{ r_1, \ldots, r_m \}, t \)
  - means that no user can be authorized for \( t \) or more roles from \( \{ r_1, \ldots, r_m \} \)

Examples
- \( \text{smer} \{ r_1, r_2 \}, 2 \) means that \( r_1 \) and \( r_2 \) are mutually exclusive, i.e., no user can be authorized for both roles
- \( \text{smer} \{ r_1, r_2, r_3 \}, 2 \) is equivalent to
  \[ \{ \text{smer} \{ r_1, r_2 \}, 2 \}, \text{smer} \{ r_2, r_3 \}, 2 \}, \text{smer} \{ r_1, r_3 \}, 2 \} \]
- \( \text{smer} \{ r_1, r_2, r_3 \}, 3 \) means that no user can be authorized for all three roles
Generation of SMER

- How did SMER constraints get there in the first place (for us to consider EV)?
- Alternate approach: start with set E of SSoD policies, then generate SMER constraints.
- The generation problem
  - Input: PA,RH,E
  - Output: C
  - Goal: C should implement \langle PA,RH,E \rangle as precisely as possible
First Step: From SSoD to RSSoD

- SSoD policies are about permissions
- SMER constraints are about role memberships
- Need to translate requirements on permissions to those on roles
  - \( \text{ssod}\{\{p_1, \ldots, p_n\}, k\} \) no \( k-1 \) users have all permissions
  - \( \text{rssod}\{\{r_1, \ldots, r_n\}, k\} \) no \( k-1 \) users have all roles
  - \( \text{smer}\{\{r_1, \ldots, r_m\}, t\} \) no single user has \( t \) or more roles
Example

- Example:
  - $E = \{ \text{ssod}(\{p_1,p_2,p_3,p_4,p_5\}, 3) \}$
  - $PA = \{(r_1,p_1), (r_2,p_2), (r_3,p_3), (r_4,p_4), (r_4,p_5)\}$
  is equivalent to
  - $D = \{ \text{rssod}(\{r_1,r_2,r_3,r_4\}, 3) \}$
  under every RH
The Generation Problem Restated

- Given a set D of RSSoD requirements and a role hierarchy RH, generate a set C of SMER constraints that implements D under RH

- Compatibility between C and RH
  - SMER constraints may render some roles unusable, e.g., given C={smer({r1,r2},2)} and RH={r3≥r1, r3≥r2}, no user can ever be authorized for r3
Implements

**Definition:** C implements D under RH iff.
- C is compatible with RH
  - every role in RH can be made nonempty without violating C
- C enforces D under RH
  - for every UA such that (UA,RH) satisfies C, (UA,RH) is safe wrt D
Example

- \( D = \{ \text{rssod}\{r_1, r_2, r_3, r_4\}, 3\} \)
- \( RH = \{ r_5 \geq r_1, r_5 \geq r_2 \} \)

Then

- \( C_1 = \{ \text{smer}\{r_1, r_2, r_3\}, 2\} \) enforces \( D, RH \), but is incompatible with \( RH \)
- \( C_2 = \{ \text{smer}\{r_1, r_3, r_4\}, 2\} \) implements \( D, RH \)
- \( C_3 = \{ \text{smer}\{r_1, r_3\}, 2\}, \text{smer}\{r_2, r_4\}, 2\), \text{smer}\{r_3, r_4\}, 2\} \) also implements \( D, RH \)
Precise Implementation

- C is necessary to enforce D under RH
  - if for every UA, (UA,RH) is safe wrt D and every role in D has at least one authorized user implies that (UA,RH) satisfies C
- C precisely enforces D under RH, iff
  - C enforces D under RH, and
  - C is necessary to enforce D under RH
- C precisely implements D under RH iff
  - C implements D under RH, and
  - C is necessary to enforce D under RH
Expressive Power Questions

- Do we need SMER constraints other than 2-2? Answer: yes
  - ex1: \( D = \{ \text{rssod}(\{r_1,r_2,r_3\}, 2) \} \), \( RH=\{r_4\geq r_1, r_4\geq r_2, r_5\geq r_1, r_5\geq r_3, r_6\geq r_2, r_6\geq r_3\} \), \( C=\{\text{smer}(\{r_1,r_2,r_3\}, 3)\} \) implements \( D \), but no set of 2-2 SMER constraints would be compatible with \( RH \)
  - do we have such examples showing the need for \( k-k \) SMER constraints for arbitrary \( k \)? Yes.
  - ex2: when \( RH=\emptyset \), to precisely enforce \( D = \{ \text{rssod}(\{r_1,r_2,r_3\}, 2) \} \), one still need 3-3 SMER
Expressive Power Questions

- Can we do without 2-2 SMER (or 2-n SMER)?
  Answer: No.
Restrictiveness of Constraints

- Goal: “least restrictive” set of constraints that implements D under RH
- $C_1$ is less restrictive than $C_2$ under RH if the UA’s allowed by $C_1$ is a strict superset of the UA’s allowed by $C_2$.
- C is minimal if C implements D and no other constraint that implements D is less restrictive.
- If C is precise, then C is minimal.
Precise Implementation is not always Possible

- $D = \{ \text{rssod}(\{r_1, r_2, r_3, r_4\}, 3) \}$
- $RH = \{ r_5 \geq r_1, r_5 \geq r_2 \}$
- $C_2 = \{ \text{smer}(\{r_1, r_3, r_4\}, 2) \}$ implements $D, RH$
- $C_3 = \{ \text{smer}(\{r_1, r_3\}, 2), \text{smer}(\{r_2, r_4\}, 2), \text{smer}(\{r_3, r_4\}, 2) \}$ also implements $D, RH$

- Both $C_2$ and $C_3$ minimally enforce $D$ under $RH$
A Generation Algorithm That Works for RH=∅

Input: rssod(R, k)
Output: SMER constraints
1  Let n = |R|, S = emptyset
2  If k = 2 output smer(R, n)
3  Else
4     for all j from 2 to floor(((n-1)/(k-1)) + 1)
5       let m = (k-1)(j-1) + 1
6     for each size-m subset R’ of R
7       output smer(R’, j)
Output of the Algorithm

- If $k = 2$, output is $\text{smer}(R, n)$
- If $k = n$, output is $\text{smer}(R, 2)$
- In other cases, we get multiple outputs. Each is sufficient to enforce the RSSoD
  - each constraint that is generated is minimal.
  - every singleton set of constraints that is minimal is generated.
Next Lecture

- UNIX Access Control