

CS590U

Access Control: Theory and Practice

Lecture 5 (September 9th)

The Proposed NIST RBAC
Standard and Constraints



Announcements

- Class mailing list
- David Ferraiolo visit on Sep 16/17
 - sign-up sheet to meet him after the class
- Proposal due Sep 30 (20% of grade)
- Student lectures (From Oct 16 to Nov 25)



Topics Covered Previously

- Why a course on access control
- Principles of access control
- Access Matrix Model
 - Graham-Denning
 - HRU (adequacy of the simple safety problem)
- DAC, MAC
- RBAC: RBAC96 model

3

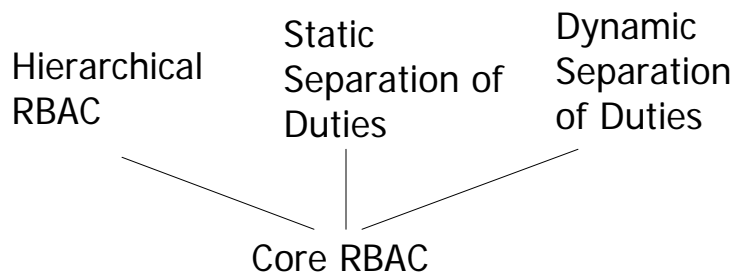


Topics to Be Covered

- Trust management
 - PolicyMaker, KeyNote, SPKI/SDSI, RT
 - background on logic and Datalog
- Access control in databases
- Mandatory Access Control
- Guest lectures
- Student lectures

4

Overview of the Proposal NIST Standard for RBAC



5

Core RBAC (1)

- *USERS*
- *ROLES*
- *OBS*
- *OPS*
- $PRMS = 2^{(OPS \times OBS)}$
 - $Op : (p: PRMS) \rightarrow 2^{OPS}$
 - $Ob : (p: PRMS) \rightarrow 2^{OBS}$

6



Core RBAC (2)

- $UA \subseteq USERS \times ROLES$
 - $assigned_users : (r : ROLES) \rightarrow 2^{USERS}$
- $PA \subseteq PRMS \times ROLES$
 - $assigned_permissions : (r : ROLES) \rightarrow 2^{PRMS}$

7



Core RBAC (3)

- $SESSIONS$
- $session_users : (s : SESSIONS) \rightarrow USERS$
 - $user_sessions : (u : USERS) \rightarrow 2^{SESSIONS}$
- $session_roles : (s : SESSIONS) \rightarrow 2^{ROLES}$
 - $avail_session_perms : (s : SESSIONS) \rightarrow 2^{PRMS}$

8



Hierarchical RBAC: Generalized Role Hierarchies

- $RH \subseteq ROLES \times ROLES$
 - user inheritance & permission inheritance
 - we say r_1 inherits r_2 if $r_1 \geq r_2$
- $authorized_users : (r : Roles) \rightarrow 2^{USERS}$
- $authorized_permissions :$
 $(r : Roles) \rightarrow 2^{PRMS}$

9



Hierarchical RBAC: Limited Role Hierarchies

- Role Hierarchies with the limitation that each role has at most one immediate senior
 - Role hierarchies form a forest

10



Constrained RBAC: Motivations

- Example of SoD
 - The following duties shall be performed by different individuals:
 1. Check request reviewer
 2. Check preparer
 3. Check issuer
 4. Check deliverer
 5. Ledger reviewer

11



Constrained RBAC: Static SoD

- $SSD \subseteq (2^{ROLES} \times \mathbb{N})$ is a collection of pairs (rs, n)
 - rs : a role set
 - n : $n \geq 2$ is a natural number
- For each (rs, n) , no user is **authorized** for n or more roles in rs

12



SoD with Role Hierarchies

- Two roles can be mutually exclusive only if neither one inherits the other
- If two roles are mutually exclusive, no role can inherit from both
- If two roles are mutually exclusive, there can be no “root” or “super user”.

13



Constrained RBAC: Dynamic SoD

- $DSD \subseteq (2^{ROLES} \times \mathbb{N})$ is a collection of pairs (rs, n)
 - rs : a role set
 - n : $n \geq 2$ is a natural number
- For each (rs, n) , no user is allowed to activate n or more roles in rs in one session

14



Functional Specifications

- Administrative functions
- Supporting system functions
- Review functions

15



Functional Specification for Core RBAC (1)

- Administrative functions
 - AddUser, DeleteUser, AddRole, DeleteRole
 - AssignUser, DeassignUser
 - GrantPermission, RevokePermission
- Supporting system functions
 - CreateSession, AddActiveRole, DropActiveRole, CheckAccess

16



Functional Specification for Core RBAC (2)

- Review functions (mandatory)
 - AssignedUsers(r), AssignRoles(u)
- Review functions (optional)
 - RolePermission(r), UserPermission(u)
 - SessionRoles(s), SessionPermissions(s)
 - RoleOperationsOnObject(o)
 - UserOperationsOnObject(o)

17



Functional Specification for Hierarchical RBAC (1)

- Administrative functions
 - all the administrative functions for core
 - semantics for some functions may need to be redefined
 - AddInheritance, DeleteInheritance
 - AddAscendant, AddDescendant
- Supporting System Functions
 - Same as core (issue of activation hierarchy)

18



Functional Specification for Hierarchical RBAC (2)

- Review functions (mandatory)
 - AssignedUsers(r), AssignendRoles(u)
 - AuthorizedUsers(r), AuthorizedRoles(u)
- Review functions (optional)
 - All the optional review functions for core

19



Functional Specification for SSD (1)

- Administrative functions
 - all the administrative functions for core
 - CreateSSDSet, DeleteSSDSet
 - AddSSDRoleMember
 - DeleteSSDRoleMember
 - SetSSDCardinality
- Supporting system functions
 - same as core

20



Functional Specification for SSD (2)

- Review functions
 - all the review functions for core
 - SSDRoleSets
 - SSDRoleSetRoles
 - SSDRoleSetCardinality

21



Functional Specification for DSD (1)

- Administrative functions
 - all the administrative functions for core
 - CreatedSDSet, DeleteSDSet
 - AddSDRoleMember
 - DeleteSDRoleMember
 - SetSDCardinality
- Supporting system functions
 - same as core

22



Functional Specification for DSD (2)

- Review functions
 - all the review functions for core
 - DSDRoleSets
 - DSDRoleSetRoles
 - DSDRoleSetCardinality

23



SoD and Permission Assignments (1)

- Mutually exclusive roles is a means rather than an end
- SoD is the goal:
 - no single user possesses all the permissions needed to accomplish a sensitive task

24



SoD and Permission Assignments (2)

- A permission assignment problem
 - Giving a set of tasks where each task requires a set of permissions, assign permissions to roles such that no single role has access to all permissions required by any task
 - Graph coloring problem

25



A Project Topic (1)

- How do we know SoD goals has been achieved by constraints?
 - sensitive tasks and the permissions they require need to be identified
- SoD may be more complicated
 - a sensitive task may be completed by a user having some property

26



A Project Topic (2)

- Tasks:
 - Design a language to specify SoD objectives.
 - Given SoD objectives and permission assignments, verify that constraints satisfy the objectives.
 - Assume a fixed permission assignments, generate mutually exclusive constraints to satisfy the SoD objectives.

27



Temporal constraints

- Why temporal constraints
 - limit resource use
 - may be required for controlling time-sensitive activities
- TRBAC [Bertino, Bonatti, and Ferrari]
- GTRBAC [Joshi, Bertino, and Ghafoor]

28



Representation of time-related concepts

- Periodicity ($[Begin, End], P$)
 - P is the periodic expression denoting a set of periodic time instants
 - e.g., ($[1/1/2002, 12/31/2002], Mondays$), ($[1-Nov-2002, 31-March-2003], 22-06$)
- Duration D
 - specifies a length of time

29



General form of temporal constraints

- (X, E)
 - X is either a periodic time or a duration
 - E is an event expression

30



Temporal Constraints

- Role-enabling and -disabling constraints
 - Periodicity constraints, e.g.,
((`[1-Nov-2002,31-March-2003]`, `22-06`),
enable doctor-on-call)
 - Duration constraints
(2 Hours, enable NurseInTraining)

31



Temporal Constraints

- Role-activation and –deactivation constraints
 - only duration is allowed
 - e.g., (`45 Min`, active download-SHR)
- Temporal constraints on user-role assignments
 - e.g., (`[1-Nov-2002,31-Dec-2002]`, assign Dr.Ken to consulting-physician)

32



Temporal constraints and Role Hierarchies

- Permission inheritance becomes more complicated
 - a junior role may not be enabled