RBAC: Role-based Access Control

NIST standards

Introduction

Applications

- database management
- security management
- network operating system products
- workflow management, ERP/CRM et al.

ANSI/NIST Standardization of RBAC

- Reference Model
- Functional Specifications

Reference Model

RBAC elements

- users
- roles
- permissions
- operations
- objects

Reference model

defines set of basic RBAC elements and relations as types and functions

Minimum set of features

- aspect of role hierarchies
- aspect of static constraint relations
- aspect of dynamic constraint relations

Functional Specifications

Administrative operations

- create
- delete
- maintain

Administrative reviews

- query operations on RBAC elements & relations

System level functionality

- creation of user sessions to include role activation/deactivation
- enforcement of constraints on role activation
- for calculation of access decisions

Reference Model: components

1. Core RBAC
2. Hierarchical RBAC
3. Static Separation of Duty (SSD) Relations
4. Dynamic Separation of Duty (DSD) Relations

RBAC components

1. Core RBAC
2. Hierarchical RBAC
   - general role hierarchies
   - limited role hierarchies
3. Static Separation of Duty (SSD) Relations
   - Core RBAC
   - SSD relations with general role hierarcies
   - SSD relations with limited role hierarcies
4. Dynamic Separation of Duty (DSD) Relations
   - Core RBAC
   - DSD relations with general role hierarcies
   - DSD relations with limited role hierarcies

Core RBAC
five basic data elements

1. USERS
2. ROLES
3. OBS - objects
4. OPS - operations
5. PRMS - permissions

Core RBAC
element sets and relations

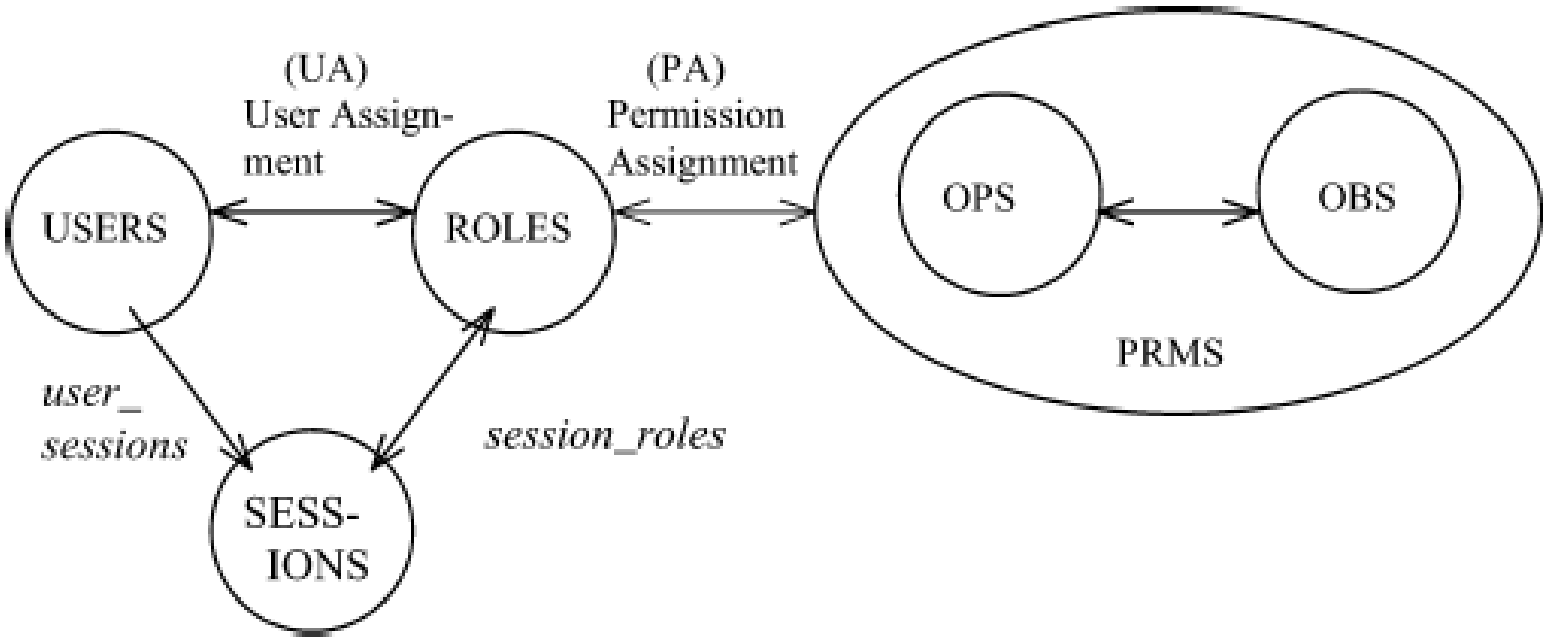

Figure: Core RBAC

Core RBAC

ROLES

means for naming many-to-many relationships among individual users and permissions (semantic construct for formulating policy)

SESSIONS

mapping between a user and an activated subset of roles that are assigned to the user

OBS - objects

information containers - entity that contains/receives information (file/dir, tuples/column/view in databases, printer)

Core RBAC
specification

- $UA \subseteq USERS \times ROLES$

Core RBAC
specification

- $UA \subseteq USERS \times ROLES$
- $assigned\_users(r : ROLES) \rightarrow 2^{USERS}$
  $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$

Core RBAC
specification

- $UA \subseteq USERS \times ROLES$
- $assigned\_users(r : ROLES) \rightarrow 2^{USERS}$
  $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$
- $PRMS = 2^{(OPS \times OBS)}$

Core RBAC
specification

- $UA \subseteq USERS \times ROLES$
- $assigned\_users(r : ROLES) \rightarrow 2^{USERS}$
  $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$
- $PRMS = 2^{(OPS \times OBS)}$
- $PA \subseteq PRMS \times ROLES$

Core RBAC
specification

- $UA \subseteq USERS \times ROLES$
- $assigned\_users(r : ROLES) \rightarrow 2^{USERS}$
  $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$
- $PRMS = 2^{(OPS \times OBS)}$
- $PA \subseteq PRMS \times ROLES$
- $assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}$
  $assigned\_permissions(r) = \{p \in PRMS \mid (p, r) \in PA\}$

Core RBAC
specification

- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$

Core RBAC
specification

- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$

Core RBAC
specification

- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$
- $SESSIONS = the\ set\ of\ sessions$

Core RBAC
specification

- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$
- $SESSIONS = the\ set\ of\ sessions$
- $session\_users(s : SESSIONS) \rightarrow USERS$

Core RBAC
specification

- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$
- $SESSIONS = the\ set\ of\ sessions$
- $session\_users(s : SESSIONS) \rightarrow USERS$
- $session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$
  $session\_roles(s_i) \subseteq \{r \in ROLES \mid (session\_users(s_i), r) \in UA\}$

Core RBAC
specification

- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$
- $SESSIONS = the\ set\ of\ sessions$
- $session\_users(s : SESSIONS) \rightarrow USERS$
- $session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$
  $session\_roles(s_i) \subseteq \{r \in ROLES \mid (session\_users(s_i), r) \in UA\}$
- $avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}$

$$avail\_session\_perms(s) = \bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$$

Hierarchical RBAC

six basic data elements

1. USERS
2. ROLES
3. OBS - objects
4. OPS - operations
5. PRMS - permissions

Hierarchical RBAC

six basic data elements

1. USERS
2. ROLES
3. OBS - objects
4. OPS - operations
5. PRMS - permissions
6. RH - role hierarchies

Hierarchical RBAC

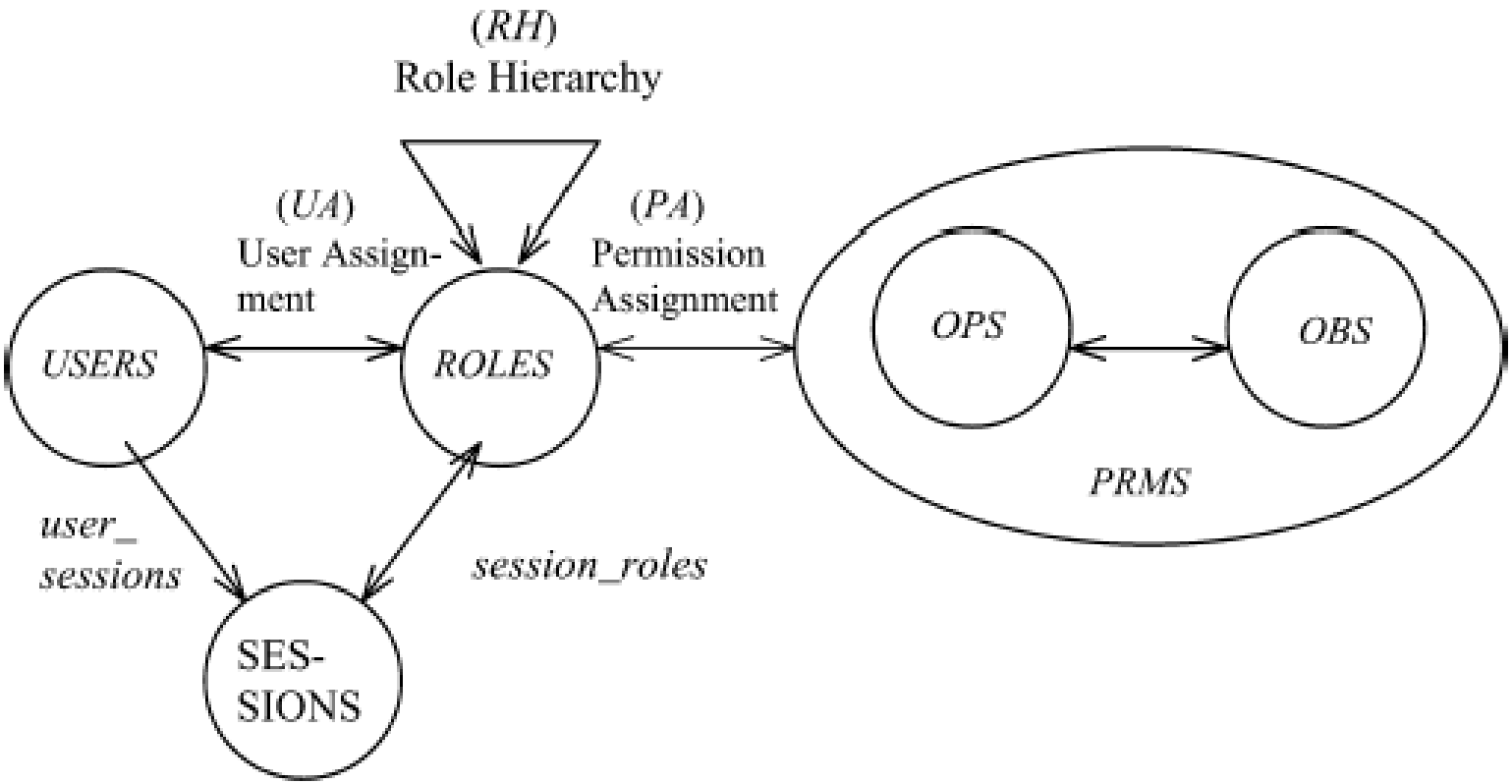

Figure: Hierarchical RBAC

Hierarchical RBAC

RH - role hierarchies

define an inheritance relation among roles

$r_1 \succeq r_2$

r_1 "inherits" role r_2
if all privileges of r_2 are also privileges of r_1

1. general role hierarchies
2. limited role hierarchies

Hierarchical RBAC

specification: general role hierarchies

- $RH \subseteq ROLES \times ROLES$
  $r_1 \succeq r_2 \Rightarrow$
  $authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$

Hierarchical RBAC

specification: general role hierarchies

- $RH \subseteq ROLES \times ROLES$
  $r_1 \succeq r_2 \Rightarrow$
  $authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$
- $authorized\_users(r : ROLES) \rightarrow 2^{USERS}$
  $authorized\_users(r) = \{u \in USERS \mid r' \succeq r, (u, r') \in UA\}$

Hierarchical RBAC

specification: general role hierarchies

- $RH \subseteq ROLES \times ROLES$
  $r_1 \succeq r_2 \Rightarrow$
  $authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$
- $authorized\_users(r : ROLES) \rightarrow 2^{USERS}$
  $authorized\_users(r) = \{u \in USERS \mid r' \succeq r, (u, r') \in UA\}$
- $authorized\_permissions(r : ROLES) \rightarrow 2^{PRMS}$
  $authorized\_permissions(r)$
  $= \{p \in PRMS \mid r' \succeq r, (p, r') \in PA\}$

Hierarchical RBAC

specification: limited role hierarchies

constraint

$\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$

Hierarchical RBAC

specification: limited role hierarchies

constraint

$\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$

- $RH \subseteq ROLES \times ROLES$
  $r_1 \succeq r_2 \Rightarrow$
  $authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$

Hierarchical RBAC

specification: limited role hierarchies

constraint

$\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$

- $RH \subseteq ROLES \times ROLES$
  $r_1 \succeq r_2 \Rightarrow$
  $authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$
- $authorized\_users(r : ROLES) \rightarrow 2^{USERS}$
  $authorized\_users(r) = \{u \in USERS \mid r' \succeq r, (u, r') \in UA\}$

Hierarchical RBAC

specification: limited role hierarchies

constraint

$\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$

- $RH \subseteq ROLES \times ROLES$
  $r_1 \succeq r_2 \Rightarrow$
  $authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$
- $authorized\_users(r : ROLES) \rightarrow 2^{USERS}$
  $authorized\_users(r) = \{u \in USERS \mid r' \succeq r, (u, r') \in UA\}$
- $authorized\_permissions(r : ROLES) \rightarrow 2^{PRMS}$
  $authorized\_permissions(r)$
  $= \{p \in PRMS \mid r' \succeq r, (p, r') \in PA\}$

Static Separation of Duty (SSD) Relations

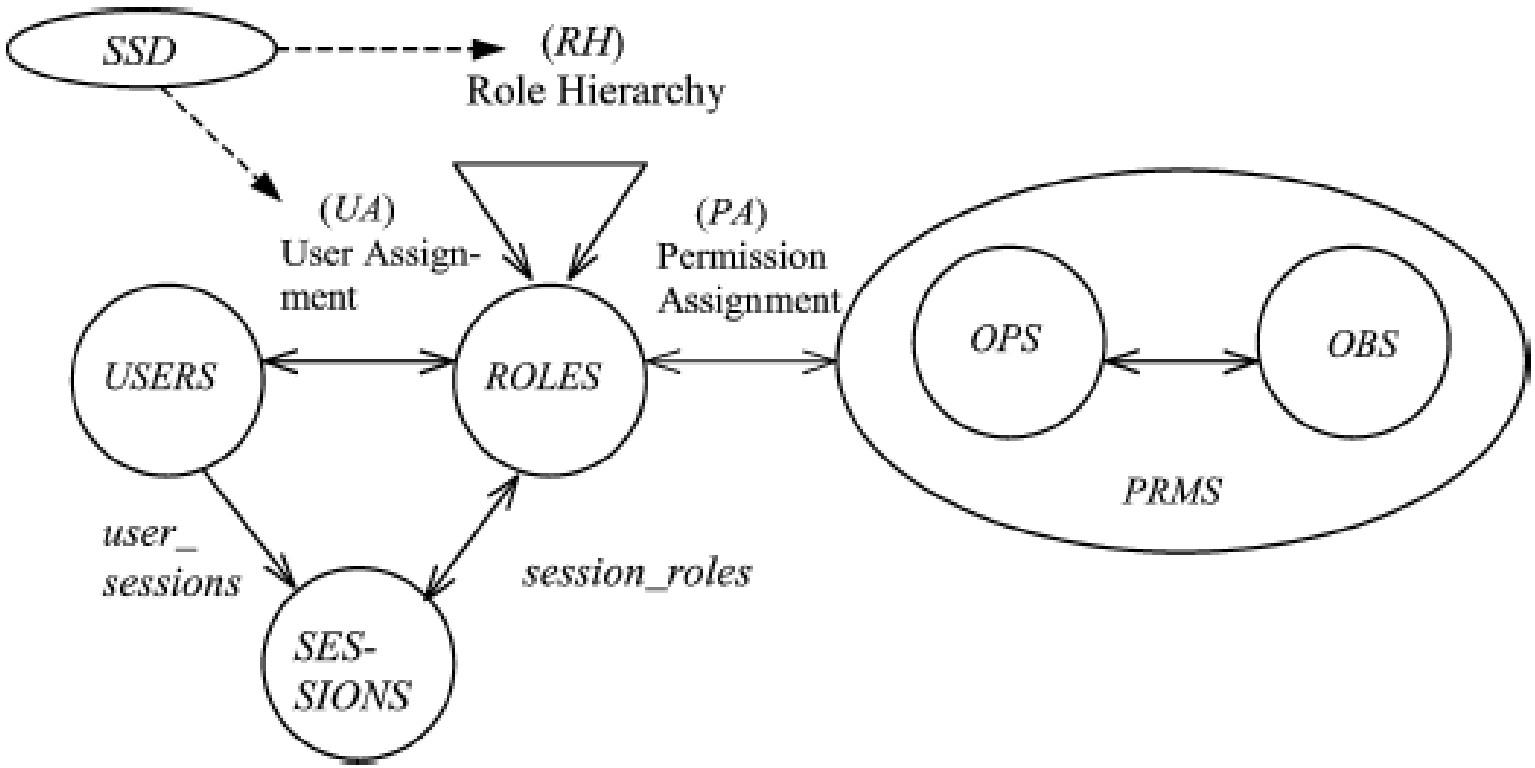

Figure: SSD within Hierarchical RBAC

Static Separation of Duty (SSD) Relations

- $SSD \subseteq (2^{ROLES} \times N)$ is a collection of pairs (rs, n) in SSD
  - each rs is a role set,
  - t a subset of roles in rs, and
  - n is a natural number ≥ 2

  with the property that no user is assigned to n or more roles from the ser rs in each $(rs, n) \in SSD$

Static Separation of Duty (SSD) Relations

- $SSD \subseteq (2^{ROLES} \times N)$ is a collection of pairs (rs, n) in SSD
  - each rs is a role set,
  - t a subset of roles in rs, and
  - n is a natural number ≥ 2

  with the property that no user is assigned to n or more roles from the ser rs in each $(rs, n) \in SSD$
- Formally,

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} assigned\_users(r) = \emptyset$$

Static Separation of Duty (SSD) Relations

- $SSD \subseteq (2^{ROLES} \times N)$ is a collection of pairs (rs, n) in SSD
  - each rs is a role set,
  - t a subset of roles in rs, and
  - n is a natural number ≥ 2

  with the property that no user is assigned to n or more roles from the ser rs in each $(rs, n) \in SSD$
- Formally,

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} assigned\_users(r) = \emptyset$$

- And, in presence of hierarchy

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} authorized\_users(r) = \emptyset$$

Dynamic Separation of Duty (DSD) Relations

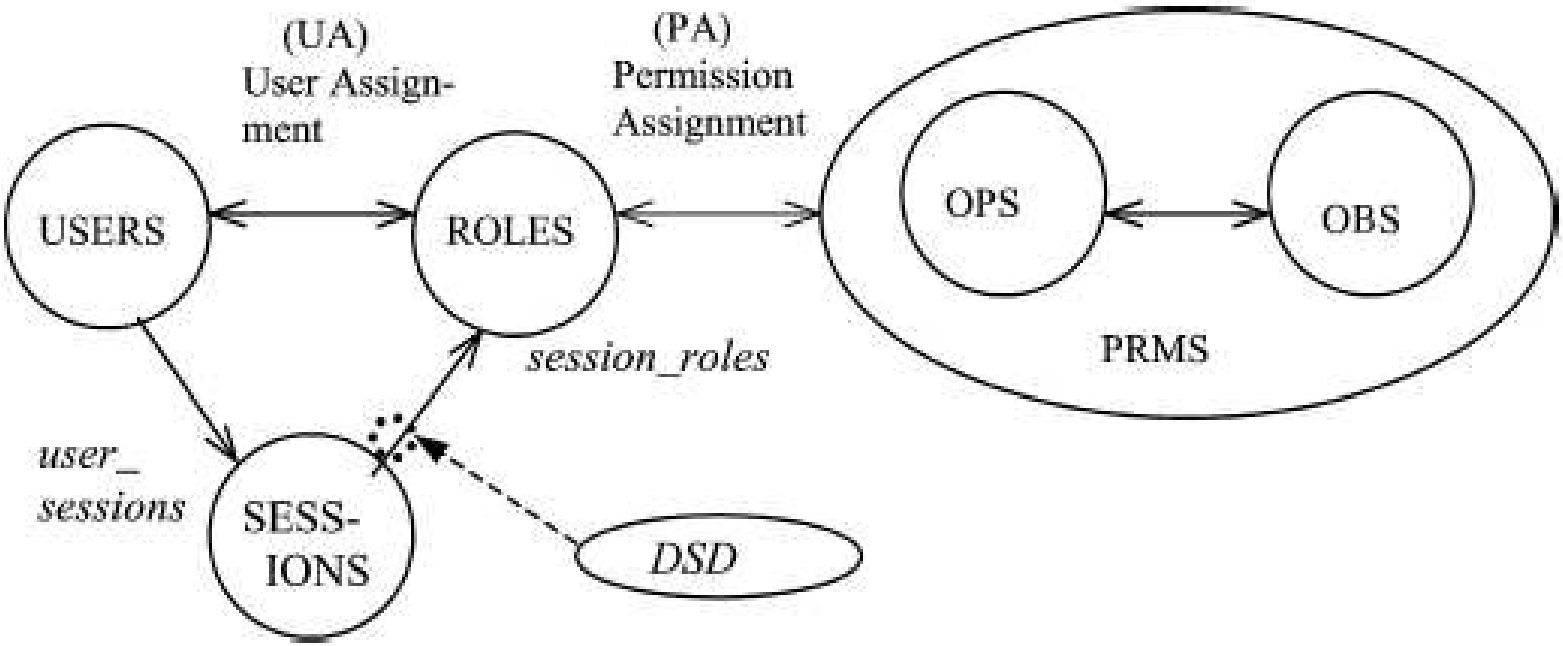

Figure: DSD Relations

Dynamic Separation of Duty (DSD) Relations

- $DSD \subseteq (2^{ROLES} \times N)$ is a collection of pairs (rs, n) in DSD
  - rs is a role set, and
  - n is a natural number ≥ 2

  with property that no subject may activate n or more roles from the set rs in each $dsd \in DSD$

Dynamic Separation of Duty (DSD) Relations

- $DSD \subseteq (2^{ROLES} \times N)$ is a collection of pairs (rs, n) in DSD
  - rs is a role set, and
  - n is a natural number ≥ 2

  with property that no subject may activate n or more roles from the set rs in each $dsd \in DSD$
- Formally,
  $\forall rs \in 2^{ROLES}, n \in N, (rs, n) \in DSD \Rightarrow n \geq 2.|rs| \geq n, and$
  $\forall s \in SESSIONS, \forall rs \in 2^{ROLES}, \forall role\_subset \in 2^{ROLES},$
  $\forall n \in N, (rs, n) \in DSD, role\_subset \subseteq rs, role\_subset \subseteq$
  $session\_roles(s) \Rightarrow |role\_subset| < n$

System and Administrative Functional Specifications

A subset of the Z notation is used for specifications.

Schema-Name(Declaration) $\lhd$ Predicate; ...; Predicate $\rhd$

NAME

an abstract data type whose elements represent identifiers of entities that may or may not be included in the RBAC system (roles, users, sessions, etc.)

Administrative Commands

- AddUser
- DeleteUser
- AddRole
- DeleteRole
- AssignUser
- DeassignUser
- GrantPermission
- RevokePermission

Core RBAC
administrative command - AddUser

$$AddUser(user : NAME) \lhd$$
$$user \notin USERS$$
$$USERS' = USERS \cup \{user\}$$
$$user\_sessions' = user\_sessions \cup \{user \mapsto \emptyset\} \rhd$$

Core RBAC
administrative command - DeleteUser

$$
\begin{aligned}
&DeleteUser(user : NAME) \lhd \\
&\quad user \in USERS \\
&\quad [\forall s \in SESSIONS \bullet s \in user\_sessions(user) \Rightarrow DeleteSession(s)] \\
&\quad UA' = UA \backslash \{r : ROLES \bullet user \mapsto r\} \\
&\quad assigned\_users' = \{r : ROLES \bullet r \mapsto (assigned\_users(r) \backslash \\
&\qquad \{user\})\} \\
&\quad USERS' = USERS \backslash \{user\} \rhd
\end{aligned}
$$

Core RBAC
administrative command - AddRole

$$AddRole(role : NAME) \lhd$$
$$role \notin ROLES$$
$$ROLES' = ROLES \cup \{role\}$$
$$assigned\_users' = assigned\_users \cup \{role \mapsto \emptyset\}$$
$$assigned\_permissions' = assigned\_permissions \cup \{role \mapsto \emptyset\} \rhd$$

Core RBAC
administrative command - DeleteRole

$$
\begin{aligned}
&DeleteRole(role : NAME) \lhd \\
&\quad role \in ROLES \\
&\quad [\forall s \in SESSIONS \bullet role \in sessions\_roles(s) \Rightarrow DeleteSession(s)] \\
&\quad UA' = UA \backslash \{u : USERS \bullet u \mapsto role\} \\
&\quad assigned\_users' = assigned\_users \backslash \{role \mapsto assigned\_users(role)\} \\
&\quad PA' = PA \backslash \{op : OPS, obj : OBJS \bullet (op, obj) \mapsto role\} \\
&\quad assigned\_permissions' = assigned\_permissions \backslash \\
&\qquad \{role \mapsto assigned\_permissions(role)\} \\
&\quad ROLES' = ROLES \backslash \{role\} \rhd
\end{aligned}
$$

Core RBAC
administrative command - AssignUser

$$AssignUser(user, role : NAME) \lhd$$
$$user \in USERS; role \in ROLES; (user \mapsto role) \notin UA$$
$$UA' = UA \cup \{user \mapsto role\}$$
$$assigned\_users' = assigned\_users \backslash \{role \mapsto assigned\_users(role)\}$$
$$\cup \{role \mapsto (assigned\_users(role) \cup \{user\})\} \rhd$$

Core RBAC
administrative command - DeassignUser

$$
\begin{aligned}
&DeassignUser(user, role : NAME) \lhd \\
&\quad user \in USERS; role \in ROLES; (user \mapsto role) \in UA \\
&\quad [\forall s : SESSIONS \bullet s \in user\_sessions(user) \land role \in \\
&\qquad session\_roles(s) \Rightarrow DeleteSession(s)] \\
&\quad UA' = UA \backslash \{user \mapsto role\} \\
&\quad assigned\_users' = assigned\_users \backslash \{role \mapsto assigned\_users(role)\} \\
&\qquad \cup \{role \mapsto (assigned\_users(role) \backslash \{user\})\} \rhd
\end{aligned}
$$

Core RBAC
administrative command - GrantPermission

$$
\begin{aligned}
&GrantPermission(object, operation, role : NAME) \lhd \\
&\quad (operation, object) \in PERMS; role \in ROLES \\
&\quad PA' = PA \cup \{(operation, object) \mapsto role\} \\
&\quad assigned\_permissions' = assigned\_permissions \setminus \\
&\qquad \{role \mapsto assigned\_permissions(role)\} \cup \\
&\qquad \{role \mapsto (assigned\_permissions(role) \cup \\
&\qquad \{(operation, object)\})\} \rhd
\end{aligned}
$$

Core RBAC
administrative command - RevokePermission

$$
\begin{aligned}
&RevokePermission(operation, object, role : NAME) \lhd \\
&\quad (operation, object) \in PERMS; role \in ROLES; \\
&\quad\quad ((operation, object) \mapsto role) \in PA \\
&\quad PA' = PA \backslash \{(operation, object) \mapsto role\} \\
&\quad assigned\_permissions' = assigned\_permissions \backslash \\
&\quad\quad \{role \mapsto assigned\_permissions(role)\} \cup \\
&\quad\quad \{role \mapsto (assigned\_permissions(role) \backslash \{(operation, object)\})\} \rhd
\end{aligned}
$$

Supporting System Functions

- CreateSession
- DeleteSession
- AddActiveRole
- DropActiveRole
- CheckAccess

Core RBAC

supporting system function - CreateSession

$$
\begin{aligned}
&CreateSession(user : NAME; ars : 2^{NAME}; session : NAME) \lhd \\
&\quad user \in USERS; ars \subseteq \{r : ROLE | (user \mapsto r) \in UA\}; \\
&\quad\quad session \notin SESSIONS \\
&\quad SESSIONS' = SESSIONS \cup \{session\} \\
&\quad user\_sessions' = user\_sessions \backslash \{user \mapsto user\_sessions(user)\} \cup \\
&\quad\quad \{user \mapsto (user\_sessions(user) \cup \{sessions\})\} \\
&\quad session\_roles' = session\_roles \cup \{session \mapsto ars\} \rhd
\end{aligned}
$$

Core RBAC

supporting system function - DeleteSession

$$
\begin{aligned}
&DeleteSession(user, session : NAME) \lhd \\
&\quad user \in USERS; session \in SESSIONS; \\
&\quad\quad session \in user\_sessions(user) \\
&\quad user\_sessions' = user\_sessions \backslash \{user \mapsto user\_sessions(user)\} \cup \\
&\quad\quad \{user \mapsto (user\_sessions(user) \backslash \{sessions\})\} \\
&\quad session\_roles' = session\_roles \backslash \{session \mapsto session\_roles(session)\} \\
&\quad SESSIONS' = SESSIONS \backslash \{session\} \rhd
\end{aligned}
$$

Core RBAC

supporting system function - AddActiveRole

$$
\begin{aligned}
&AddActiveRole(user, session, role : NAME) \lhd \\
&\quad user \in USERS; session \in SESSIONS; role \in ROLES; \\
&\qquad session \in user\_sessions(user) \\
&\quad (user \mapsto role) \in UA; role \notin session\_roles(session) \\
&\quad session\_roles' = session\_roles \setminus \\
&\qquad \{session \mapsto session\_roles(session)\} \cup \\
&\qquad \{session \mapsto (session\_roles(session) \cup \{role\})\} \rhd
\end{aligned}
$$

Core RBAC

supporting system function - DropActiveRole

$$DropActiveRole(user, session, role : NAME) \lhd$$
$$user \in USERS; role \in ROLES; session \in SESSIONS$$
$$session \in user\_sessions(user); role \in session\_roles(session)$$
$$session\_roles' = session\_roles \backslash \{session \mapsto session\_roles(session)\}$$
$$\cup \{session \mapsto (session\_roles(session) \backslash \{role\})\} \rhd$$

Core RBAC

supporting system function - CheckAccess

$$
\begin{aligned}
&\mathit{CheckAccess}(\mathit{session}, \mathit{operation}, \mathit{object} : \mathit{NAME};\\
&\quad \mathit{out}\ \mathit{result} : \mathit{BOOLEAN}) \lhd\\
&\quad \mathit{session} \in \mathit{SESSIONS}; \mathit{operation} \in \mathit{OPS}; \mathit{object} \in \mathit{OBJS}\\
&\quad \mathit{result} = (\exists r : \mathit{ROLES} \bullet r \in \mathit{session\_roles}(\mathit{session}) \land\\
&\qquad ((\mathit{operation}, \mathit{object}) \mapsto r) \in \mathit{PA}) \rhd
\end{aligned}
$$

Review Functions

- AssignedUsers
- AssignedRoles

Core RBAC

review function - AssignedUsers

$$AssignedUsers(role : NAME; out\ result : 2^{USERS}) \lhd$$
$$role \in ROLES$$
$$result = \{u : USERS | (u \mapsto role) \in UA\} \rhd$$

Core RBAC
review function - AssignedRoles

$$AssignedRoles(user : NAME; result : 2^{ROLES}) \lhd$$
$$user \in USERS$$
$$result = \{r : ROLES | (user \mapsto r) \in UA\} \rhd$$

Advanced Review Functions

- RolePermissions
- UserPermissions
- SessionRoles
- SessionPermissions
- RoleOperationsOnObject
- UserOperationsOnObject

Core RBAC

advanced review function - RolePermissions

$$RolePermissions(role : NAME; result : 2^{PERMS}) \lhd$$
$$role \in ROLES$$
$$result = \{op : OPS; obj : OBJS | ((op, obj) \mapsto role) \in PA\} \rhd$$

Core RBAC

advanced review function - UserPermissions

$$UserPermissions(user : NAME; result : 2^{PERMS}) \lhd$$
$$user \in USERS$$
$$result = \{r : ROLES; op : OPS; obj : OBJS | (user \mapsto r) \in UA \wedge ((op, obj) \mapsto r) \in PA \bullet (op, obj)\} \rhd$$

Core RBAC

advanced review function - SessionRoles

$$SessionRoles(session : NAME; out\ result : 2^{ROLES}) \lhd$$
$$session \in SESSIONS$$
$$result = session\_roles(session) \rhd$$

Core RBAC

advanced review function - SessionPermissions

$$SessionPermissions(session : NAME; out\ result : 2^{PERMS}) \lhd$$
$$session \in SESSIONS$$
$$result = \{r : ROLES; op \in OPS; obj \in OBJS |$$
$$r \in session\_roles(session) \wedge ((op, obj) \mapsto r) \in PA \bullet (op, obj)\} \rhd$$

Core RBAC

advanced review function - RoleOperationsOnObject

$$RoleOperationsOnObject(role : NAME; obj : NAME; result : 2^{OPS}) \lhd$$
$$role \in ROLES$$
$$obj \in OBJS$$
$$result = \{op : OPS | ((op, obj) \mapsto role \in PA\} \rhd$$

Core RBAC

advanced review function - UserOperationsOnObject

$$UserOperationsOnObject(user : NAME; obj : NAME; result : 2^{OPS}) \lhd$$
$$user \in USERS$$
$$obj \in OBJS$$
$$result = \{r : ROLES; op : OPS | (user \mapsto r) \in UA \wedge ((op, obj) \mapsto r) \in PA \bullet op\} \rhd$$

General Role Hierarchies
administrative commands

- AddInheritance
- DeleteInheritance
- AddAscendant
- AddDescendant

General Role Hierarchies

administrative command - AddInheritance

$$AddInheritance(r\_asc, r\_desc : NAME) \lhd$$
$$r\_asc \in ROLES; r\_desc \in ROLES; \neg(r\_asc \gg r\_desc);$$
$$\neg(r\_desc \geq r\_asc)$$
$$\geq' = \geq \cup \{r, q : ROLES | r \geq r\_asc \wedge r\_desc \geq q\bullet \mapsto q\} \rhd$$

General Role Hierarchies

administrative command - DeleteInheritance

$$DeleteInheritance(r\_asc, r\_desc : NAME) \lhd$$
$$r\_asc \in ROLES; r\_desc \in ROLES; r\_asc \gg r\_desc$$
$$\geq' = (\gg \setminus \{r\_asc \mapsto r\_desc\})^* \rhd$$

General Role Hierarchies
administrative command - AddAscendant

$AddAscendant(r\_asc, r\_desc : NAME) \lhd$
$\quad AddRole(r\_asc)$
$\quad AddInheritance(r\_asc, r\_desc) \rhd$

General Role Hierarchies
administrative command - AddDescendant

$AddDescendant(r\_asc, r\_desc : NAME) \lhd$
$\quad AddRole(r\_desc)$
$\quad AddInheritance(r\_asc, r\_desc) \rhd$

General Role Hierarchies

supporting system function

- CreateSessions
- AddActiveRole

General Role Hierarchies

supporting system function - CreateSessions

$$
\begin{aligned}
&CreateSession(user : NAME; ars : 2^{NAME}; session : NAME) \lhd \\
&\quad user \in USERS; ars \subseteq \{r, q : ROLES | (user \mapsto q) \in \\
&\qquad UA \wedge q \geq r \bullet r\}; session \notin SESSIONS \\
&\quad SESSIONS' = SESSIONS \cup \{session\} \\
&\quad user\_sessions' = user\_sessions \backslash \{user \mapsto user\_sessions(user)\} \cup \\
&\qquad \{user \mapsto (user\_sessions(user) \cup \{session\})\} \\
&\quad session\_roles' = session\_roles \cup \{session \mapsto ars\} \rhd
\end{aligned}
$$

General Role Hierarchies

supporting system function - AddActiveRole

$$
\begin{aligned}
&AddActiveRole(user, session, role : NAME) \lhd \\
&\quad user \in USERS; session \in SESSIONS; role \in ROLES; \\
&\quad\quad session \in user\_sessions(user) \\
&\quad user \in authorized\_users(role); role \notin session\_roles(session) \\
&\quad session\_roles' = session\_roles \setminus \\
&\quad\quad \{session \mapsto session\_roles(session)\} \cup \\
&\quad\quad \{session \mapsto (session\_roles(session) \cup \{role\}))\} \rhd
\end{aligned}
$$

General Role Hierarchies
review functions

- AuthorizedUsers
- AuthorizedRoles

General Role Hierarchies

review function - AuthorizedUsers

$$AuthorizedUsers(role : NAME; outresult : 2^{USERS}) \lhd$$
$$role \in ROLES$$
$$result = authorized\_users(role) \rhd$$

General Role Hierarchies

review function - AuthorizedRoles

$$AuthorizedUsers(user : NAME; result : 2^{ROLES}) \lhd$$
$$user \in USERS$$
$$result = \{r, q : ROLES | (user \mapsto q) \in UA \wedge q \geq r\} \rhd$$

General Role Hierarchies

advanced review functions

- RolePermissions
- UserPermissions
- RoleOperationsOnObject
- UserOperationsOnObject

General Role Hierarchies

advanced review function - RolePermissions

$$RolePermissions(role : NAME; result : 2^{PERMS}) \lhd$$
$$role \in ROLES$$
$$result = \{q : ROLES; op : OPS; obj : OBJS |$$
$$(role \geq q) \wedge ((op, obj) \mapsto role) \in PA \bullet (op, obj)\} \rhd$$

General Role Hierarchies

advanced review function - UserPermissions

$$UserPermissions(user : NAME; result : 2^{PERMS}) \lhd$$
$$user \in USERS$$
$$result = \{r, q : ROLES; op : OPS; obj : OBJS | (user \mapsto q) \in$$
$$UA \wedge (q \geq r) \wedge ((op, obj) \mapsto r) \in PA \bullet (op, obj)\} \rhd$$

General Role Hierarchies

advanced review function - RoleOperationsOnObject

$$RoleOperationsOnObject(role : NAME; obj : NAME; result : 2^{OPS})$$
$$role \in ROLES$$
$$obj \in OBJS$$
$$result = \{q : ROLES; op : OPS | (role \geq q) \wedge ((op, obj) \mapsto role) \in PA \bullet op\} \rhd$$

General Role Hierarchies

advanced review function - UserOperationsOnObject

$$
\begin{aligned}
&UserOperationsOnObject(user : NAME; obj : NAME;\\
&\quad result : 2^{OPS}) \lhd\\
&\quad user \in USERS\\
&\quad obj \in OBJS\\
&\quad result = \{r, q : ROLES; op : OPS \mid (user \mapsto q) \in\\
&\qquad UA \wedge (q \geq r) \wedge ((op, obj) \mapsto r) \in PA \bullet op\} \rhd
\end{aligned}
$$

Limited Role Hierarchies
administrative commands

- AddInheritance

supporting system functions
review functions
advanced review functions, remain valid.

Limited Role Hierarchies

administrative command - AddInheritance

$$AddInheritance(r\_asc, r\_desc : NAME) \lhd$$
$$r\_asc \in ROLES; r\_desc \in ROLES;$$
$$\forall r \in ROLES \bullet \neg(r\_asc \gg r); \neg(r\_desc \geq r\_asc)$$
$$\geq' = \geq \cup \{r, q : ROLES | r \geq r\_asc \wedge r\_desc \geq q \bullet r \mapsto q\} \rhd$$

Core RBAC

- *

SSD: Core RBAC
administrative commands

- AssignUser
- CreateSsdSet
- AddSsdRoleMember
- DeleteSsdRoleMember
- DeleteSsdSet
- SetSsdSetCardinality

SSD: Core RBAC

administrative command - AssignUser

$$AssignUser(user, role : NAME) \lhd$$
$$user \in USERS; role \in ROLES; (user \mapsto role) \notin UA$$

$$\forall ssd \in SSD \bullet \bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(ssd) \\ |subset| = ssd\_card(ssd) \\ us = if\ \ r = role\ \ then\ \{user\}\ \ else\ \emptyset}} (assigned\_users(r) \cup us) = \emptyset$$

$$UA' = UA \cup \{user \mapsto role\}$$
$$assigned\_users' = assigned\_users \setminus \{role \mapsto assigned\_users(role)\}$$
$$\cup \{role \mapsto (assigned\_users(role) \cup \{user\})\} \rhd$$

SSD: Core RBAC

administrative command - CreateSsdSet

$$CreateSsdSet(set\_name : NAME; role\_set : 2^{NAME}; n : N) \lhd$$
$$set\_name \notin SSD; (n \geq 2) \wedge (n \leq |role\_set|); role\_set \subseteq ROLES$$

$$\bigcap_{\substack{r \in subset \\ subset \subseteq role\_set \\ |subset| = n}} assigned\_users(r) = \emptyset$$

$$SSD' = SSD \cup \{set\_name\}$$
$$ssd\_set' = ssd\_set \cup \{set\_name \mapsto role\_set\}$$
$$ssd\_card' = ssd\_card \cup \{set\_name \mapsto n\} \rhd$$

SSD: Core RBAC

administrative command - AddSsdRoleMember

$$AddSsdRoleMember(set\_name : NAME; role : NAME) \lhd$$
$$set\_name \in SSD; role \in ROLES; role \notin ssd\_set(set\_name)$$

$$\bigcup_{\substack{r \in subset \\ subset \subseteq ssd\_set(set\_name) \cup \{role\} \\ |subset| = n}} assigned\_users(r) = \emptyset$$

$$ssd\_set' = ssd\_set \setminus \{set\_name \mapsto ssd\_set(set\_name)\} \cup$$
$$\{set\_name \mapsto (ssd\_set(set\_name) \cup \{role\})\} \rhd$$

SSD: Core RBAC

administrative command - DeleteSsdRoleMember

$$DeleteSsdRoleMember(set\_name : NAME; role : NAME) \lhd$$
$$set\_name \in SSD; role \in ssd\_set(set\_name);$$
$$ssd\_card(set\_name) < |ssd\_set(set\_name)|$$
$$ssd\_set' = ssd\_set \backslash \{set\_name \mapsto ssd\_set(set\_name)\} \cup$$
$$\{set\_name \mapsto (ssd\_set(set\_name) \backslash \{role\})\} \rhd$$

SSD: Core RBAC

administrative command - DeleteSsdSet

$$
\begin{aligned}
&DeleteSsdSet(set\_name : NAME) \lhd \\
&\quad set\_name \in SSD; ssd\_card' = ssd\_card \backslash \\
&\qquad \{set\_name \mapsto ssd\_card(set\_name)\} \\
&\quad ssd\_set' = ssd\_set \backslash \{set\_name \mapsto ssd\_set(set\_name)\} \\
&\quad SSD' = SSD \backslash \{set\_name\} \rhd
\end{aligned}
$$

SSD: Core RBAC

administrative command - SetSsdSetCardinality

$$SetSsdSetCardinality(set\_name : NAME; n : N) \lhd$$
$$set\_name \in SSD; (n \geq 2) \wedge (n \leq |ssd\_set(set\_name)|)$$

$$\bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(set\_name) \\ |subset| = n}} assigned\_users(r) = \emptyset$$

$$ssd\_card' = ssd\_card \backslash \{set\_name \mapsto ssd\_card(set\_name)\} \cup$$
$$\{set\_name \mapsto n\} \rhd$$

SSD: Core RBAC

supporting system functions

same

SSD: Core RBAC
review functions

- SsdRoleSets
- SsdRoleSetRoles
- SsdRoleSetCardinality

SSD: Core RBAC

review function - SsdRoleSets

$$SsdRoleSets(outresult : 2^{NAME}) \lhd result = SSD \rhd$$

SSD: Core RBAC

review function - SsdRoleSetRoles

$$SsdRoleSetRoles(set\_name : NAME; outresult : 2^{ROLES}) \lhd$$
$$set\_name \in SSD$$
$$result = ssd\_set(set\_name) \rhd$$

SSD: Core RBAC

review function - SsdRoleSetCardinality

$$SsdRoleSetCardinality(set\_name : NAME; outresult : N) \lhd$$
$$set\_name \in SSD$$
$$result = ssd\_card(set\_name) \rhd$$

SSD: Core RBAC
advanced review functions

same

SSD: General Role Hierarchy

administrative commands

- AssignUser
- AddInheritance
- CreateSsdSet
- AddSsdRoleMember
- SetSsdSetCardinality

supporting system functions
review functions
advanced review functions, remain valid

SSD: General Role Hierarchy

administrative command - AssignUser

$$AssignUser(user, role : NAME) \lhd$$
$$user \in USERS; role \in ROLES; (user \mapsto role) \notin UA$$

$$\forall ssd \in SSD \bullet \bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(ssd) \\ |subset| = ssd\_card(ssd) \\ au = if\ \ r\ = role\ \ then\ \{user\}\ \ else\ \emptyset}} (authorized\_users(r) \cup au) = \emptyset$$

$$UA' = UA \cup \{user \mapsto role\}$$
$$assigned\_users' = assigned\_users \setminus \{role \mapsto assigned\_users(role)\}$$
$$\cup \{role \mapsto (assigned\_users(role) \cup \{user\})\} \rhd$$

SSD: General Role Hierarchy

administrative command - AddInheritance

$$AddInheritance(r\_asc, r\_desc : NAME) \lhd$$
$$r\_asc \in ROLES; r\_desc \in ROLES;$$
$$\neg(r\_asc \gg r\_desc); \neg(r\_desc \geq r\_asc)$$

$$\forall ssd \in SSD \bullet \bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(ssd) \\ |subset| = ssd\_card(ssd) \\ au = if\ r = r\_desc\ then\ authorized\_users(r\_asc)\ else\ \emptyset}}$$

$$(authorized\_users(r) \cup au) = \emptyset$$
$$\geq' = \geq \cup \{r, q : ROLES | r \geq r\_asc \wedge r\_desc \geq q \bullet r \mapsto q\} \rhd$$

SSD: General Role Hierarchy

administrative command - CreateSsdSet

$$CreateSsdSet(set\_name : NAME; role\_set : 2^{NAME}; n : N) \lhd$$
$$set\_name \notin SSD; (n \geq 2) \wedge (n \leq |role\_set|); role\_set \in ROLES$$

$$\bigcap_{\substack{r \in subset \\ subset \subseteq role\_set \\ |subset| = n}} authorized\_users(r) = \emptyset$$

$$SSD' = SSD \cup \{set\_name\}$$
$$ssd\_set' = ssd\_set \cup \{set\_name \mapsto role\_set\}$$
$$ssd\_card' = ssd\_card \cup \{set\_name \mapsto n\} \rhd$$

SSD: General Role Hierarchy

administrative command - AddSsdRoleMember

$$AddSsdRoleMember(set\_name : NAME; role : NAME) \lhd$$
$$set\_name \in SSD; role \in ROLES; role \notin ssd\_set(set\_name)$$

$$\bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(set\_name) \cup \{role\} \\ |subset| = n}} authorized\_users(r) = \emptyset$$

$$ssd\_set' = ssd\_set \setminus \{set\_name \mapsto ssd\_set(set\_name)\} \cup$$
$$\{set\_name \mapsto (ssd\_set(set\_name) \cup \{role\})\} \rhd$$

SSD: General Role Hierarchy

administrative command - SetSsdSetCardinality

$$SetSsdSetCardinality(set\_name : NAME; n : N) \lhd$$
$$set\_name \in SSD; (n \geq 2) \wedge (n \leq |ssd\_set(set\_name)|)$$

$$\bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(set\_name) \\ |subset| = n}} authorized\_users(r) = \emptyset$$

$$ssd\_card' = ssd\_card \backslash \{set\_name \mapsto ssd\_card(set\_name)\} \cup$$
$$\{set\_name \mapsto n\} \rhd$$

SSD: Limited Role Hierarchy

administrative commands

- AddInheritance

supporting system functions
review functions
advanced review functions, remain valid

SSD: Limited Role Hierarchy

administrative command - AddInheritance

$$AddInheritance(r\_asc, r\_desc : NAME) \lhd$$
$$r\_asc \in ROLES; r\_desc \in ROLES;$$
$$\forall r \in ROLES \bullet \neg(r\_asc \gg r); \neg(r\_desc \geq r\_asc)$$

$$\forall ssd \in SSD \bullet \bigcap_{\substack{r \in subset \\ subset \subseteq ssd\_set(ssd) \\ |subset| = ssd\_card(ssd) \\ au = \text{if } r = r\_desc \text{ then } authorized\_users(r\_asc) \text{ else } \emptyset}}$$

$$(authorized\_users(r) \cup au) = \emptyset$$
$$\geq' = \geq \cup \{r, q : ROLES \mid r \geq r\_asc \wedge r\_desc \geq q \bullet r \mapsto q\} \rhd$$

DSD: Core RBAC
administrative commands

- CreateDsdSet
- AddDsdRoleMember
- DeleteDsdRoleMember
- DeleteDsdSet
- SetDsdSetCardinality

DSD: Core RBAC

administrative command - CreateDsdSet

$$
\begin{aligned}
&CreateDsdSet(set\_name : NAME; role\_set : 2^{NAME}; n : N) \lhd \\
&\quad set\_name \notin DSD; (n \geq 2) \wedge (n \leq |role\_set|); role\_set \subseteq ROLES \\
&\quad \forall s : SESSIONS; role\_subset : 2^{role\_set} \bullet role\_subset \subseteq \\
&\qquad session\_roles(s) \Rightarrow |role\_subset| < n \\
&\quad DSD' = DSD \cup \{set\_name\} \\
&\quad dsd\_set' = dsd\_set \cup \{set\_name \mapsto role\_set\} \\
&\quad dsd\_card' = dsd\_card \cup \{set\_name \mapsto n\} \rhd
\end{aligned}
$$

DSD: Core RBAC

administrative command - AddDsdRoleMember

$$
\begin{aligned}
&AddDsdRoleMember(set\_name : NAME; role\_name : NAME) \lhd \\
&\quad set\_name \in DSD; role \in ROLES; role \notin dsd\_set(set\_name) \\
&\quad \forall s : SESSIONS; role\_subset : 2^{dsd\_set(set\_name) \cup \{role\}} \bullet role\_subset \\
&\qquad \subseteq session\_roles(s) \Rightarrow |role\_subset| < dsd\_card(set\_name) \\
&\quad dsd\_set' = dsd\_set \backslash \{set\_name \mapsto dsd\_set(set\_name)\} \cup \\
&\qquad \{set\_name \mapsto (dsd\_set(set\_name) \cup \{role\})\} \rhd
\end{aligned}
$$

DSD: Core RBAC

administrative command - DeleteDsdRoleMember

$$
\begin{aligned}
&DeleteDsdRoleMember(set\_name : NAME; role\_name : NAME) \lhd \\
&\quad set\_name \in DSD; role \in dsd\_set(set\_name); \\
&\quad\quad dsd\_card(set\_name) < |dsd\_set(set\_name)| \\
&\quad dsd\_set' = dsd\_set \backslash \{set\_name \mapsto dsd\_set(set\_name)\} \cup \\
&\quad\quad \{set\_name \mapsto (dsd\_set(set\_name) \backslash \{role\})\} \rhd
\end{aligned}
$$

DSD: Core RBAC

administrative command - DeleteDsdSet

$$
\begin{aligned}
&DeleteDsdSet(set\_name : NAME) \lhd \\
&\quad set\_name \in DSD \\
&\quad dsd\_card' = dsd\_card \backslash \{set\_name \mapsto dsd\_card(set\_name)\} \\
&\quad dsd\_set' = dsd\_set \backslash \{set\_name \mapsto dsd\_set(set\_name)\} \\
&\quad DSD' = DSD \backslash \{set\_name\} \rhd
\end{aligned}
$$

DSD: Core RBAC

administrative command - SetDsdSetCardinality

$$SetDsdSetCardinality(set\_name : NAME; n : N) \lhd$$
$$set\_name \in DSD; (n \geq 2) \wedge (n \leq |dsd\_set(set\_name)|)$$
$$\forall s : SESSIONS; role\_subset : 2^{dsd\_set(set\_name)} \bullet$$
$$role\_subset \subseteq session\_roles(s) \Rightarrow |role\_subset| < n$$
$$dsd\_card' = dsd\_card \backslash \{set\_name \mapsto dsd\_card(set\_name)\} \cup$$
$$\{set\_name \mapsto n\} \rhd$$

DSD: Core RBAC

supporting system functions

- CreateSession
- AddActiveRole

DSD: Core RBAC

supporting system function - CreateSession

$$
\begin{aligned}
&CreateSession(user : NAME; ars : 2^{NAME}; session : NAME) \lhd \\
&\quad user \in USERS; ars \subseteq \{r : ROLES | (user \mapsto r) \in UA\}; \\
&\quad\quad session \notin SESSIONS \\
&\quad \forall dset : DSD; rset : 2^{NAME} \bullet \\
&\quad\quad rset \subseteq dsd\_set(dset) \land rset \subseteq ars \Rightarrow |rset| < dsd\_card(dset) \\
&\quad SESSIONS' = SESSIONS \cup \{session\} \\
&\quad user\_sessions' = user\_sessions \backslash \{user \mapsto user\_sessions(user)\} \cup \\
&\quad\quad \{user \mapsto (user\_sessions(user) \cup \{session\})\} \\
&\quad session\_roles' = session\_roles \cup \{session \mapsto ars\} \rhd
\end{aligned}
$$

DSD: Core RBAC

supporting system function - AddActiveRole

$$
\begin{aligned}
&AddActiveRole(user, session, role : NAME) \lhd \\
&\quad user \in USERS; session \in SESSIONS; role \in ROLES; \\
&\qquad session \in user\_sessions(user) \\
&\quad user \in assigned\_users(role); role \notin session\_roles(session) \\
&\quad \forall dset : DSD; rset : 2^{NAME} \bullet \\
&\qquad rset \subseteq dsd\_set(dset) \land rset \subseteq session\_roles(session) \cup \\
&\qquad \{role\} \Rightarrow |rset| < dsd\_card(dset) \\
&\quad session\_roles' = session\_roles \setminus \{session \mapsto session\_roles(session)\} \\
&\qquad \cup \{session \mapsto (session\_roles(session) \cup \{role\})\} \rhd
\end{aligned}
$$

DSD: Core RBAC
review functions

- DsdRoleSets
- DsdRoleSetRoles
- DsdRoleSetCardinality

Advanced review functions, remain valid

DSD: Core RBAC

review function - DsdRoleSets

$$DsdRoleSets(outresult : 2^{NAME}) \lhd result = DSD \rhd$$

DSD: Core RBAC

review function - DsdRoleSetRoles

$$DsdRoleSetRoles(set\_name : NAME; outresult : 2^{ROLES}) \lhd$$
$$set\_name \in DSD$$
$$result = dsd\_set(set\_name) \rhd$$

DSD: Core RBAC

review function - DsdRoleSetCardinality

$$DsdRoleSetCardinality(set\_name : NAME; outresult : N) \lhd$$
$$set\_name \in DSD$$
$$result = dsd\_card(set\_name) \rhd$$

DSD: General Role Hierarchy

supporting system functions

Administrative commands, remain valid

- CreateSession
- AddActiveRole

Review functions

Advanced review functions, remain valid

DSD: General Role Hierarchy

supporting system function - CreateSession

$$CreateSession(user : NAME; ars : 2^{NAME}; session : NAME) \lhd$$

$$user \in USERS; ars \subseteq \{r, q : ROLES | (user \mapsto q) \in UA \land q \geq r \bullet r\}$$
$$session \notin SESSIONS$$
$$\forall dset : DSD; rset : 2^{NAME} \bullet$$
$$rset \subseteq dsd\_set(dset) \land rset \subseteq ars \Rightarrow |rset| < dsd\_card(dset)$$
$$SESSIONS' = SESSIONS \cup \{session\}$$
$$user\_sessions' = user\_sessions \backslash \{user \mapsto user\_sessions(user)\} \cup$$
$$\{user \mapsto (user\_sessions(user) \cup \{session\})\}$$
$$session\_roles' = session\_roles \cup \{session \mapsto ars\} \rhd$$

DSD: General Role Hierarchy

supporting system function - AddActiveRole

$$
\begin{aligned}
&AddActiveRole(user, session, role : NAME) \lhd \\
&\quad user \in USERS; session \in SESSIONS; role \in ROLES; \\
&\quad\quad session \in user\_session(user) \\
&\quad \forall dset : DSD; rset : 2^{NAME} \bullet \\
&\quad\quad rset \subseteq dsd\_set(dset) \land rset \subseteq session\_roles(session) \cup \\
&\quad\quad \{role\} \Rightarrow |rset| < dsd\_card(dset) \\
&\quad session\_roles' = session\_roles \setminus \\
&\quad\quad \{session \mapsto session\_roles(session)\} \cup \\
&\quad\quad \{session \mapsto (session\_roles(session) \cup \{role\})\} \rhd
\end{aligned}
$$

DSD: Limited Role Hierarchy

Administrative commands
Supporting system functions
Review functions
Advanced review functions, remain valid