

# Access Control and Trust Negotiation

## CS 6463 Lecture 7

Prof. William Winsborough  
February 12, 2007

## Business

- I will post a few papers tonight after class

February 12, 2007 Winsborough CS 6463 Lecture 7 2

## Role-Based Access Control (RBAC)

- Roles introduce a level of indirection between users and permissions
  - Permissions subsume objects and rights on those objects
  - While groups focus on aggregating users, roles are also meant to aggregate permissions
    - Significance of this distinction is debatable

Users	Roles	Permissions
Alice	sales	sign checks
Bob	accountant	create purchase order
Carol	buyer	fire employee
	manager	

February 12, 2007 Winsborough CS 6463 Lecture 7 3

## Four "Reference" Models

February 12, 2007 Winsborough CS 6463 Lecture 7 4

(b) RBAC models

February 12, 2007 Winsborough CS 6463 Lecture 7 5

## Base Model: RBAC<sub>0</sub>

- Three sets of entities:
  - U – users
  - R – roles
  - P – permissions
    - Subsumes object and mode of access
    - Always positive
  - S – sessions
    - Associated with a single user
    - User can dynamically activate and deactivate roles within each of his sessions

February 12, 2007 Winsborough CS 6463 Lecture 7 6

**Definition 1** The  $RBAC_0$  model has the following components:

- $U, R, P,$  and  $S$  (users, roles, permissions and sessions respectively),
- $PA \subseteq P \times R$ , a many-to-many permission to role assignment relation,
- $UA \subseteq U \times R$ , a many-to-many user to role assignment relation,
- $user : S \rightarrow U$ , a function mapping each session  $s_i$  to the single user  $user(s_i)$  (constant for the session's lifetime), and
- $roles : S \rightarrow 2^R$ , a function mapping each session  $s_i$  to a set of roles  $roles(s_i) \subseteq \{r \mid (user(s_i), r) \in UA\}$  (which can change with time) and session  $s_i$  has the permissions  $\cup_{r \in roles(s_i)} \{p \mid (p, r) \in PA\}$ .  $\square$

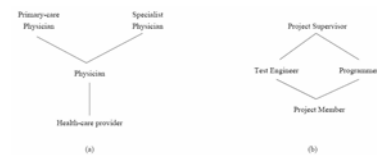
## Role Hierarchies: $RBAC_1$

- Structure roles to reflect an organization's lines of authority and responsibility
  - Senior roles toward the top
  - Junior roles toward the bottom
- Mathematical partial orders
  - Reflexive, transitive, anti-symmetric

## Inheritance

- Members of senior roles inherit permissions of more junior roles
  - Adding a user to a role effectively adds that user to all junior roles, too
  - Adding a permission to a role effectively adds the permission to all senior roles, too

## Role Hierarchy Examples



- Test Engineer' is a *private role*
  - Has permissions not inherited by Project Supervisor

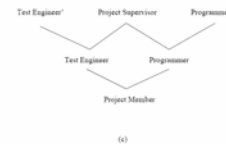


Figure 2: Examples of Role Hierarchies

**Definition 2** The  $RBAC_1$  model has the following components:

- $U, R, P, S, PA, UA,$  and  $user$  are unchanged from  $RBAC_0$ ,
- $RH \subseteq R \times R$  is a partial order on  $R$  called the role hierarchy or role dominance relation, also written as  $\geq$ , and
- $roles : S \rightarrow 2^R$  is modified from  $RBAC_0$  to require  $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA]\}$  (which can change with time) and session  $s_i$  has the permissions  $\cup_{r \in roles(s_i)} \{p \mid (\exists r'' \leq r)[(p, r'') \in PA]\}$ .  $\square$

## $RBAC_2$ : The Addition of Constraints

- Constraints are used to prevent unsafe configuration of RBAC components
  - Classic example: mutually exclusive roles
  - The constraint that states  $r_1$  and  $r_2$  are mutually exclusive prevents any user belonging to both roles
    - Intended to provide separation of duty
    - E.g., purchasing manager and accounts payable manager
- By defining constraints, the security officer can delegate more responsibility for updating the system when personnel changes

## Other Constraints

- Mutual exclusion in terms of PA
  - Only one role should write checks
- Cardinality constraints
- Prerequisite roles
  - For users and for permissions

February 12, 2007

Winsborough CS 6463 Lecture 7

13

## Requirements for Effective Use of Constraints

- Each person should have only one user id
- Each operation should be permitted by only one permission

February 12, 2007

Winsborough CS 6463 Lecture 7

14

## Constraints on Sessions

- Dynamic mutual exclusion constraints
  - Prevents activating two roles in the same session
- Can limit the number of sessions a user can have active at one time

February 12, 2007

Winsborough CS 6463 Lecture 7

15

## RBAC<sub>3</sub>: Consolidated Model

- Combines both hierarchies and constraints
- Allows constraints on the hierarchy
  - E.g., two roles can be required to have no common senior (or junior) roles
- Meaning of constraints is more subtle in the presence of hierarchy
  - E.g., if Test Engineer and Programmer are mutually exclusive, does that mean Project Supervisor must be uninhabited?



February 12, 2007

Winsborough CS 6463 Lecture 7

16

## Management Model: Administration

- In many environments, the chief security officer needs assistance
- Administrative roles (AR) and administrative permissions (AP)
  - Only regular permissions can be assigned to regular roles
  - Only administrative permissions can be assigned to administrative roles

February 12, 2007

Winsborough CS 6463 Lecture 7

17

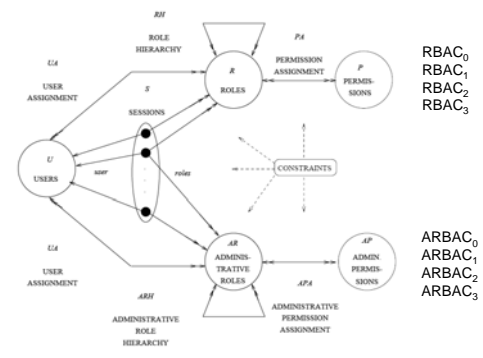


Figure 4: RBAC Administrative Model

February 12, 2007

Winsborough CS 6463 Lecture 7

18

## Administrative Permissions

- Permission to modify
  - User assignment
  - Permission assignment
  - Role hierarchy
- Exact means by which these are defined is not stated in this paper

February 12, 2007

Winsborough CS 6463 Lecture 7

19

## Critique of ANSI Standard

- A Critique of the ANSI Standard on Role Based Access Control
  - Ninghui Li, Ji-Won Byun, Elisa Bertino
- Identifies and suggests corrections to “limitations, design flaws, and technical errors”

February 12, 2007

Winsborough CS 6463 Lecture 7

20

## Overview of ANSI Standard

- Four components to standard
  - Required:
    - Core RBAC
  - Optional:
    - Role hierarchy
    - Static Separation of Duty Relations (SSD)
    - Dynamic Separation of Duty Relations (DSD)
- Model is slightly different from Sandhu'96 and much more precise in several respects

February 12, 2007

Winsborough CS 6463 Lecture 7

21

## Core RBAC (From the Standard)

- $USERS$ ,  $ROLES$ ,  $OPS$ , and  $OBS$  (users, roles, operations and objects respectively).
- $UA \subseteq USERS \times ROLES$ , a many-to-many mapping user-to-role assignment relation.
- $assigned\_users : (r : ROLES) \rightarrow 2^{USERS}$ , the mapping of role  $r$  onto a set of users. Formally:  $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$
- $PRMS = 2^{(OPS \times OBS)}$ , the set of permissions.
- $PA \subseteq PRMS \times ROLES$ , a many-to-many mapping permission-to-role assignment relation.
- $assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}$ , the mapping of role  $r$  onto<sup>1</sup> a set of permissions. Formally:  $assigned\_permissions(r) = \{p \in PRMS \mid (p, r) \in PA\}$
- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$ , the permission to operation mapping, which gives the set of operations associated with permission  $p$ .
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$ , the permission to object mapping, which gives the set of objects associated with permission  $p$ .

February 12, 2007

Winsborough CS 6463 Lecture 7

22

## Core RBAC, Continued

- $SESSIONS$  = the set of sessions
- $session\_users(s : SESSIONS) \rightarrow USERS$ , the mapping of session  $s$  onto the corresponding user.
- $session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$ , the mapping of session  $s$  onto a set of roles. Formally:  $session\_roles(s) \subseteq \{r \in ROLES \mid (session\_users(s), r) \in UA\}$
- $avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}$ , the permissions available to a user in a session =  $\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$

February 12, 2007

Winsborough CS 6463 Lecture 7

23

## General Role Hierarchies

### General Role Hierarchies

- $BH \subseteq ROLES \times ROLES$  is a partial order on  $ROLES$  called the inheritance relation, written as  $\succeq$ , where  $r_1 \succeq r_2$  only if all permissions of  $r_2$  are also permissions of  $r_1$ , and all users of  $r_1$  are also users of  $r_2$ , i.e.,  $r_1 \succeq r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$ .
- $authorized\_users(r : ROLES) \rightarrow 2^{USERS}$ , the mapping of role  $r$  onto a set of users in the presence of a role hierarchy. Formally:  $authorized\_users(r) = \{u \in USERS \mid r' \succeq r, (u, r') \in UA\}$
- $authorized\_permissions(r : ROLES) \rightarrow 2^{PRMS}$ , the mapping of role  $r$  onto a set of permissions in the presence of a role hierarchy. Formally:  $authorized\_permissions(r) = \{p \in PRMS \mid r' \succeq r, (p, r') \in PA\}$ <sup>3</sup>

Node  $r_1$  is represented as an immediate descendant of  $r_2$  by  $r_1 \succ r_2$ , if  $r_1 \succeq r_2$ , but no role in the role hierarchy lies between  $r_1$  and  $r_2$ . That is, there exists no role  $r_3$  in the role hierarchy such that  $r_1 \succeq r_3 \succeq r_2$ , where  $r_1 \neq r_3$  and  $r_2 \neq r_3$ .<sup>4</sup>

February 12, 2007

Winsborough CS 6463 Lecture 7

24