

Access Control and Trust Negotiation

CS 6463 Lecture 6

Prof. William Winsborough
February 7, 2007

Business

- I will soon post a collection of papers that you can consider presenting in class
 - When you find a paper you would be interested in giving a try, send me a note
 - Each paper will be allocated to the person that first requests it
 - Some papers no one selects I will present

February 7, 2007

Winsborough CS 6463 Lecture 6

2

Harrison, Ruzzo and Ullman, 1976

- We continue studying this classical paper in access control

February 7, 2007

Winsborough CS 6463 Lecture 6

3

Decision Procedure for all Mono-operational Systems

- When the commands are parameters to the decision procedure, the problem of deciding whether a protection system is safe is co-NP-complete
- Reduce k-clique to safety
- Question: why doesn't this result contradict the result that for a given protection system, a polynomial algorithm can be found?

February 7, 2007

Winsborough CS 6463 Lecture 6

4

Undecidability in the General Case

- Reduction of any Turing machine to a protection system that enters r into a cell if and only if the Turing machine halts

February 7, 2007

Winsborough CS 6463 Lecture 6

5

Recall What a Turing Machine Is

- Each Turing machine T consists of
 - A finite set of *states* K and
 - A distinct finite set of *tape symbols* P
 - P includes the *blank* symbol, B
- B initially appears on each cell of an infinite tape
 - Tape cells are numbered $1, 2, \dots, i, \dots$
- There is a *tape head* which is always *scanning* (located at) some cell of the tape
- Moves are specified by

February 7, 2007

Winsborough CS 6463 Lecture 6

6

Turing Machine Computation

- Moves of T are specified by a function $\delta: K \times F \rightarrow K \times F \times \{L, R\}$
- If $\delta(q, X) = (p, Y, R)$ for states p and q and tape symbols X and Y, then should the Turing machine T find itself in state q, with its tape head scanning a cell holding symbol X, then T enters state p, erases X and prints Y on the tape cell scanned and moves its tape head one cell to the right
- If $\delta(q, X) = (p, Y, L)$, the same thing happens, but the tape head moves one cell left (but never off the left end of the tape at cell 1)

February 7, 2007

Winsborough CS 6463 Lecture 6

7

Termination is Undecidable

- Initially, T is in state q_0 , the *initial state*, with its head at cell 1
- Each tape cell holds the blank.
- There is a particular state q_f , known as the *final state*
- It is known to be undecidable whether started as above, an arbitrary Turing machine T will eventually enter state q_f

February 7, 2007

Winsborough CS 6463 Lecture 6

8

Theorem 2

- It is undecidable whether a given configuration of a given protection system is safe for a given generic right

February 7, 2007

Winsborough CS 6463 Lecture 6

9

Reducing Turing Machine Termination to the Safety Problem

Fig. 3. Representing a tape.

	s_1	s_2	s_3	s_4
s_1	{W}	{own}		
s_2		{X,q}	{own}	
s_3			{Y}	{own}
s_4				{Z,end}

February 7, 2007

Winsborough CS 6463 Lecture 6

10

$\delta(q, X) = (p, Y, L)$,

then there is

command $C_{q,x}(s, s')$

```

if
  own in (s, s') and
  q in (s', s') and
  X in (s', s')
then
  delete q from (s', s')
  delete X from (s', s')
  enter p into (s, s)
  enter Y into (s', s')
end
    
```

February 7, 2007

Winsborough CS 6463 Lecture 6

11

$\delta(q, X) = (p, Y, R)$,

that is, the tape head moves right, then we have two commands, depending whether or not the head passes the current end of the tape, that is, the *end* right. There is

command $C_{q,x}(s, s')$

```

if
  own in (s, s') and
  q in (s, s) and
  X in (s, s)
then
  delete q from (s, s)
  delete X from (s, s)
  enter p into (s', s')
  enter Y into (s, s)
end
    
```

February 7, 2007

Winsborough CS 6463 Lecture 6

12

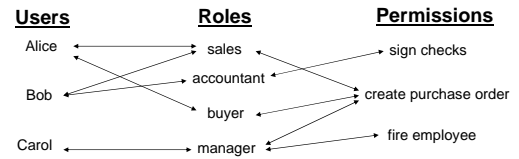
```

command  $D_{ex}(s, s')$ 
if
  end in  $(s, s)$  and
   $q$  in  $(s, s)$  and
   $X$  in  $(s, s)$ 
then
  delete  $q$  from  $(s, s)$ 
  delete  $X$  from  $(s, s)$ 
  create subject  $s'$ 
  enter  $B$  into  $(s', s')$ 
  enter  $p$  into  $(s', s')$ 
  enter  $Y$  into  $(s, s)$ 
  delete end from  $(s, s)$ 
  enter end into  $(s', s')$ 
  enter own into  $(s, s')$ 
end

```

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



Goals and Features

- Facilitate security administration and review
 - Centralized administration
 - Define security policy in terms of organizational roles
- Relationships can exist between roles
 - Mutual exclusion
 - Role inheritance

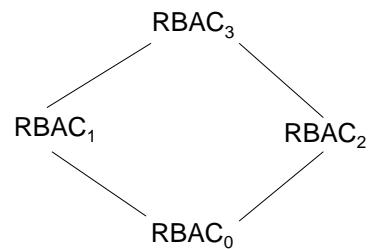
Issues

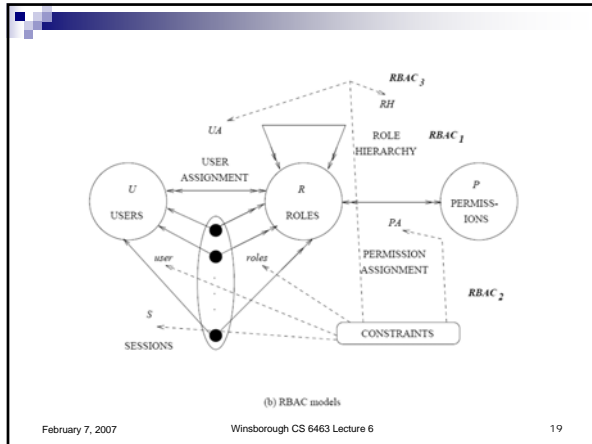
- Level at which implemented:
 - Typically done at the application or middleware level
 - Little support exists at OS level
- “Policy neutral”
 - Consistent with both
 - Mandatory Access Control (MAC)
 - Discretionary Access Control (DAC)

Supported Security Principles

- Least privilege
 - Can configure system giving each role only permissions required to perform associated tasks
- Separation of duties
 - Mutual exclusion of roles
- Data abstraction
 - Depends on implementation details: application or middleware

Four “Reference” Models





Base Model: RBAC₀

- 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 7, 2007 Winsborough CS 6463 Lecture 6 20

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

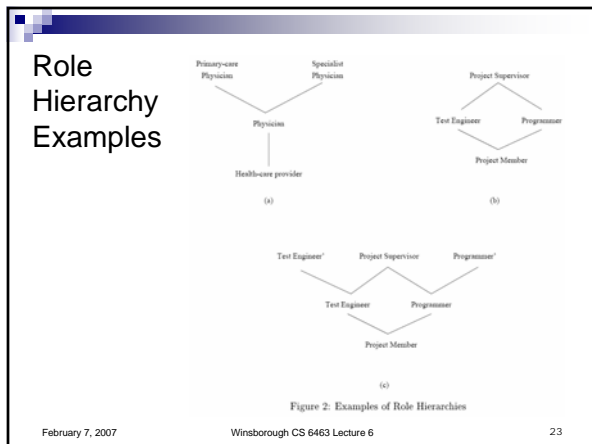
- $U, R, P,$ and S (users, roles, permissions and sessions respectively),
- $PA \subseteq P \times R$, a many-to-many permission to role assignment relation,
- $UA \subseteq U \times R$, a many-to-many user to role assignment relation,
- $user : S \rightarrow U$, a function mapping each session s_i to 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 $\cup_{r \in roles(s_i)} \{p \mid (p, r) \in PA\}$. □

February 7, 2007 Winsborough CS 6463 Lecture 6 21

Role Hierarchies: RBAC₁

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

February 7, 2007 Winsborough CS 6463 Lecture 6 22



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 $\cup_{r \in roles(s_i)} \{p \mid (\exists r'' \leq r)[(p, r'') \in PA]\}$. □

February 7, 2007 Winsborough CS 6463 Lecture 6 24