Administrative Privilege Inheritance in RBAC

Open Session Talk, FOSAD 2006

Marnix Dekker, TNO ICT, Security group

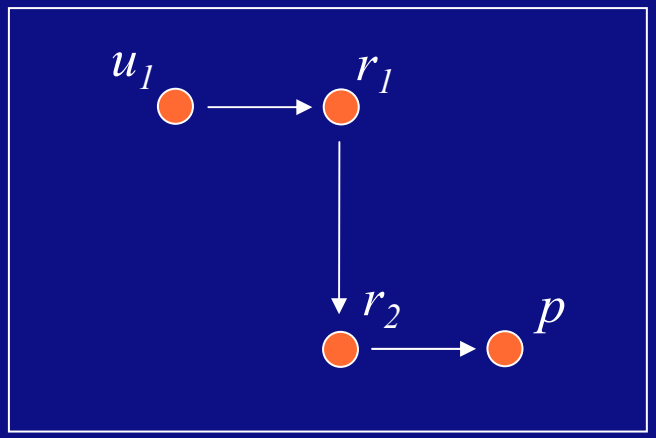
Administrative Privilege Inheritance in RBAC

- This talk is about ongoing work with:
  - with Jan Cederquist*, Jason Crampton** and Sandro Etalle*
    *) Distributed and Embedded Systems Group,
    University of Twente
    **) Information Security Group,
    Royal Holloway, University of London

- My research is funded by
  TNO (a large non-profit research institute in the Netherlands) and
  SenterNovem (an agency of the Dutch Ministry of Economics)

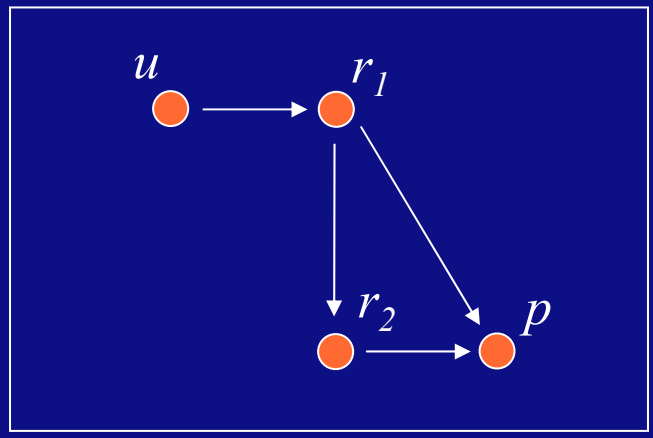
Preliminaries

- Standard RBAC (Sandhu, 1996):
  - Sets of users U, roles R and privileges P.
  - User-role *UA*, role hierarchy *RH*, role privilege *PA.*
- In the sequel:
  - I will not use/describe *constraints (RBAC$_2$).*
  - A graph with *unique* nodes.
  - the RBAC *state* is formed by the *directed* edges between the nodes.



Preliminaries

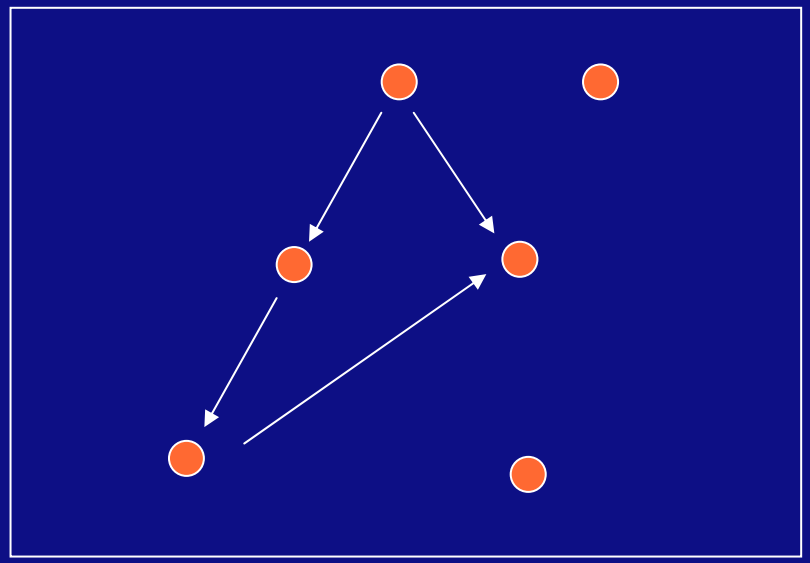
- I write r » p for: A user in role *r* has a privilege *p.*
- RBAC's *Privilege Inheritance*:
  - r » p if (r, p) ϵ PA
  - r » p if (r, r') ϵ RH and r' » p



- PI spares the user from going through the passes of actually activating a lower role, without repetitve definitions.

Administrative Privileges

- *Administrative* privileges allow *administrative operations*, (while actions on resources, e.g. files, are allowed by so-called *user privileges)*.



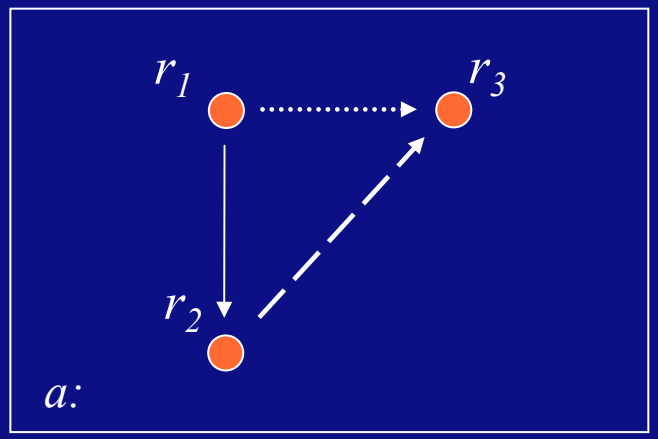
Administrative Privileges

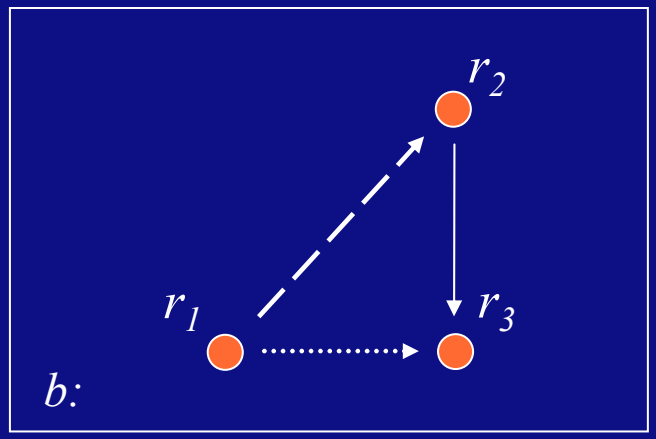
- In RBAC the set of privileges P is finite (mayprint, mayquery, etc.).
- Administrative Privileges P* are not finite, why?
  r may give r' the privilege to give r'' the privilege mayprint
- So we use a (straightforward) grammar for the privileges:

  p := q | root | addUser(u,r) | addEdge(r_i,r_j) | addPrivilege(r,p)

- The set of "well-formed" privileges *P** is inifinite,
  although U,R,P are finite.

Administrative Privileges





"The *right to add* a dashed edge, is *stronger* than the right to add the dotted edge. "

Administrative Privileges

- Adding the dashed edge is *more safe* then adding the dotted edge, because your only excluding some users from using the edge to get privileges.
- From a security point of view, you want the weaker administrative privilege to be available, because you would prefer it to be used instead. (kind of *Principle of least privilege)*

Ordering Administrative Privileges

- Using the role-hierarchy, we “order” the administrative privileges:

- When is an admin priv stronger then another:
  The $\rightarrow$ on P* is defined as:
  - $addEdge(r_2, r) \rightarrow addEdge(r_1, r)$ if $(r_1, r_2) \in RH$
  - ...

  (just like in the pictures. And additionally...)

  - $root \rightarrow p$
  - $addPrivilege(r, p_1) \rightarrow addPrivilege(r, p_2)$ *if* $p_1 \rightarrow p_2$

Extended privilege inheritance

- The extended privilege inheritance relation is defined as:
  - r » p if (r, p) ϵ PA
  - r » p if (r, r') ϵ RH and r' » p

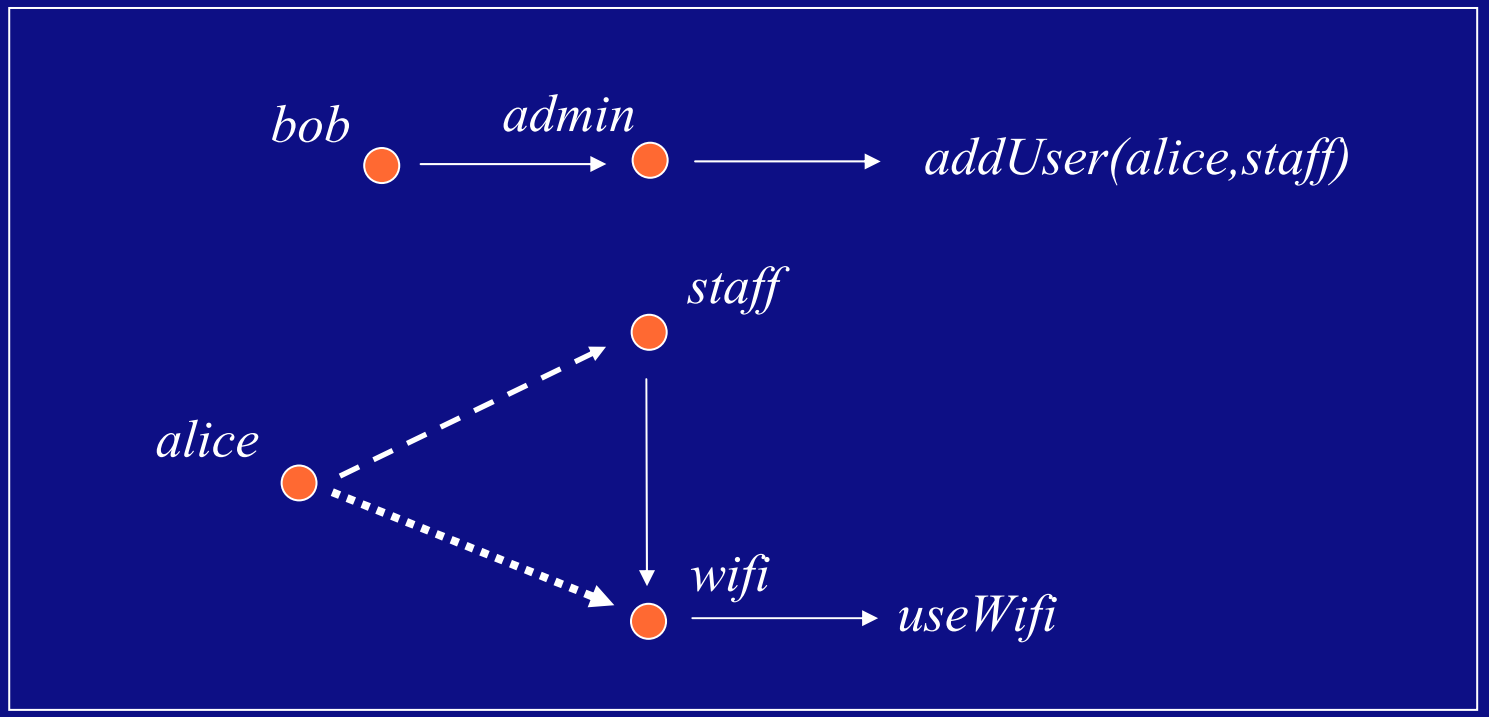
  (just as in standard RBAC. And additionally...)

  - r » p if r » p' and p'→ p

- In words, a role inherits a privilege, either directly, or through a role below it, or because it inherits a *stronger* administrative privilege.

Practical example

- standard RBAC: Bob can *only* assign Alice to the staff role.



- extended RBAC: Bob can *also* assign Alice to the wifi role.

Decidability

1. r » p if (r, p) ϵ PA
2. r » p if (r, r') ϵ RH and r' » p
3. r » p if r » p' and p'→ p

- The standard privilege inheritance is decidable:
  Given a role r we can find the set of all privileges p such that r » p. We call it Z.
- Since the 2nd rule and the 3rd commute, we can use the 3rd rule last. So take Z, and repeatedly apply the 3rd rule.
- Given a p' can we determine all the privileges p, such that p'→*p?*
  Not in general, but... Given p', p, it *is* decidable whether p'→ p.

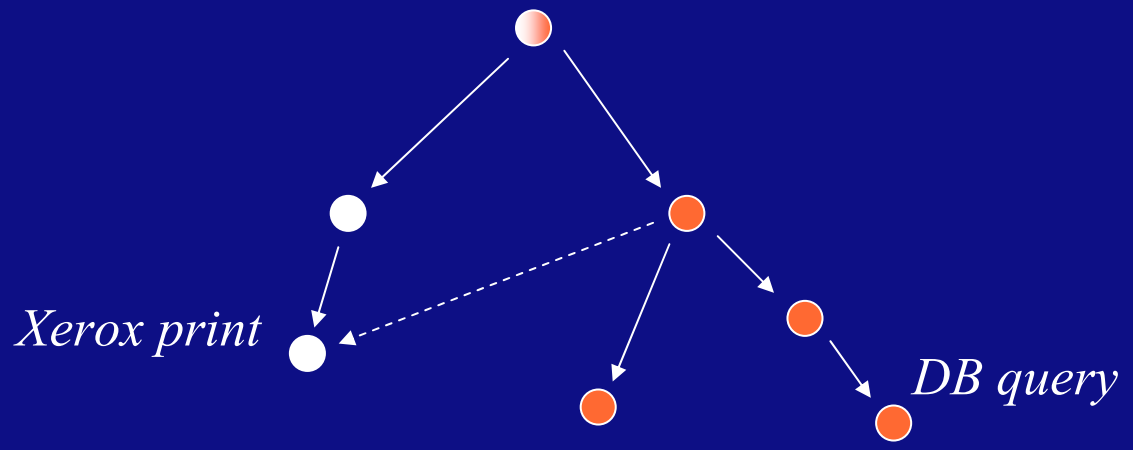
Conclusions

- There is a more “adequate” way of inheriting administrative privileges
  - which is not *less* safe
  - which is more flexible
  - which can be implemented (decidable)
- Recall: PI spares the user from going through the passes of actually activating a lower role, without repetitve definitions.
- Extended PI spares the user from going through the passes of actually requesting a weaker administrative privilege, without repetitive definitions.

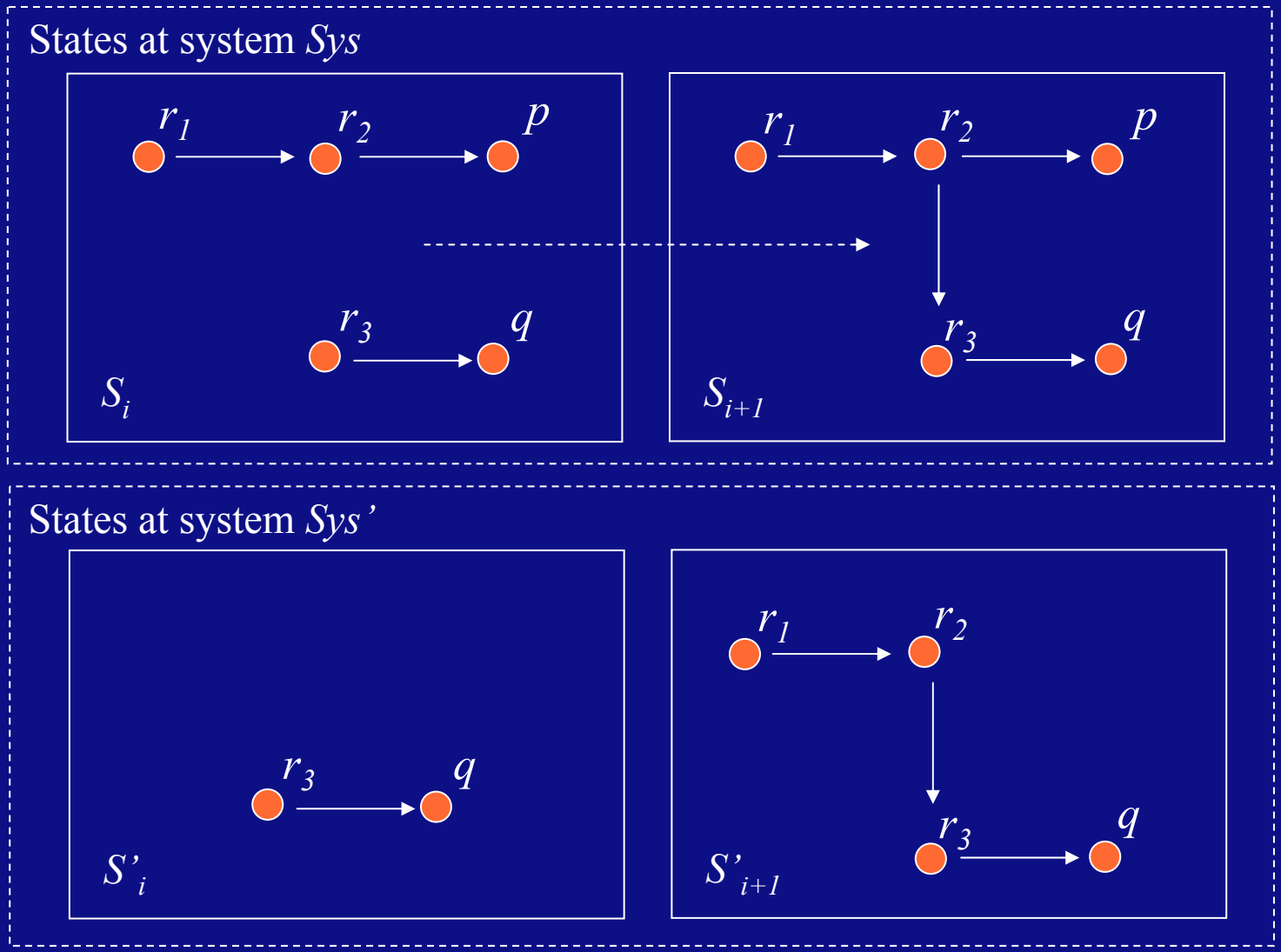
Practical Example

- For example, an enterprise, with, among others a printer system. The enterprise uses an RBAC hierarchy to protect resources. Only the printing privileges, and roles and users assigned to it, are relevant for the printer (similar distinction may hold for administrative privileges).

- In the picture: only the white nodes are relevant for the printer. But if edges are added, this may change (see the dashed arrow). This means that, in addition to an administrative operation, *extra* information may need to be sent.



Example


States at system Sys
r1
r2
p
r3
q
Si
r1
r2
p
r3
q
Si+1
States at system Sys'
r3
q
S'i
r1
r2
r3
q
S'i+1

Completeness and Up-to-date-ness

- **Completeness**:
  An RBAC state S' *is complete for privilege p in* S, if every role r that inherits p in S, also inherits p in S'.

- **Up to date-ness**:
  A system A, with privileges P, in state S is up to date, with respect to another system B in state S' , if the state S is *complete* for all the privileges P in S'.

- Basically, edges outside the upward closure of the relevant privileges are irrelevant. The upward closure may change due to administrative operations...

Completing information σ for Administrative Operations

- σ([ADD r_1, r_2]) is S(↑ r_1).
- σ([ADD r, p]) is S(↑ r) if p is a user privilege.
  - if p is an administrative privilege, then
    - if p = *root*, then σ is S
    - if p = addEdge(r_1, r_2), then σ is S(↑ r) and S(↑ r_1) and S(↑ r_2).
    - if p = addPriv(r_1, p), then σ is S(↑ r) and S(↑ r_1).

Monotonicity of Completing information

- Given two administrative operations a_1 and a_2, guarded by the administrative privileges p_1 and p_2 then

  if p1 $\rightarrow$ p2, then $\sigma(a_2)$ is a subset of $\sigma(a_1)$

- No need for discovery protocols like in distributed credential systems!

marnix.dekker@tno.nl

Security Group, TNO ICT,Delft

Distributed and Embedded Systems group, University of Twente