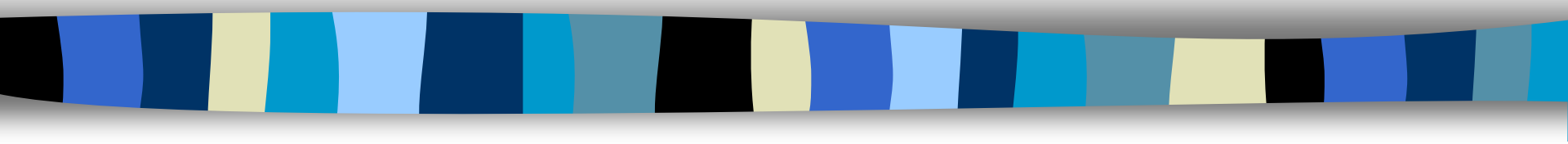
Data Security and Privacy



Topic 8: Role Based Access Control

Plan for this lecture

- CodeShield: towards personalized application whitelisting**.** Christopher S. Gates, Ninghui Li, Jing Chen, Robert W. Proctor: ACSAC 2012: 279-288
- RBAC96 Family
  - R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. "Role-Based Access Control Models". *IEEE Computer*, 29(2):38--47, February 1996.
- ANSI RBAC standard and its critique
  - N. Li, J.-W. Byun, and E. Bertino. "A Critique of the ANSI Standard on Role Based Access Control". *IEEE Security & Privacy, 5(6):41--49, November 2007.*

Application Whitelisting

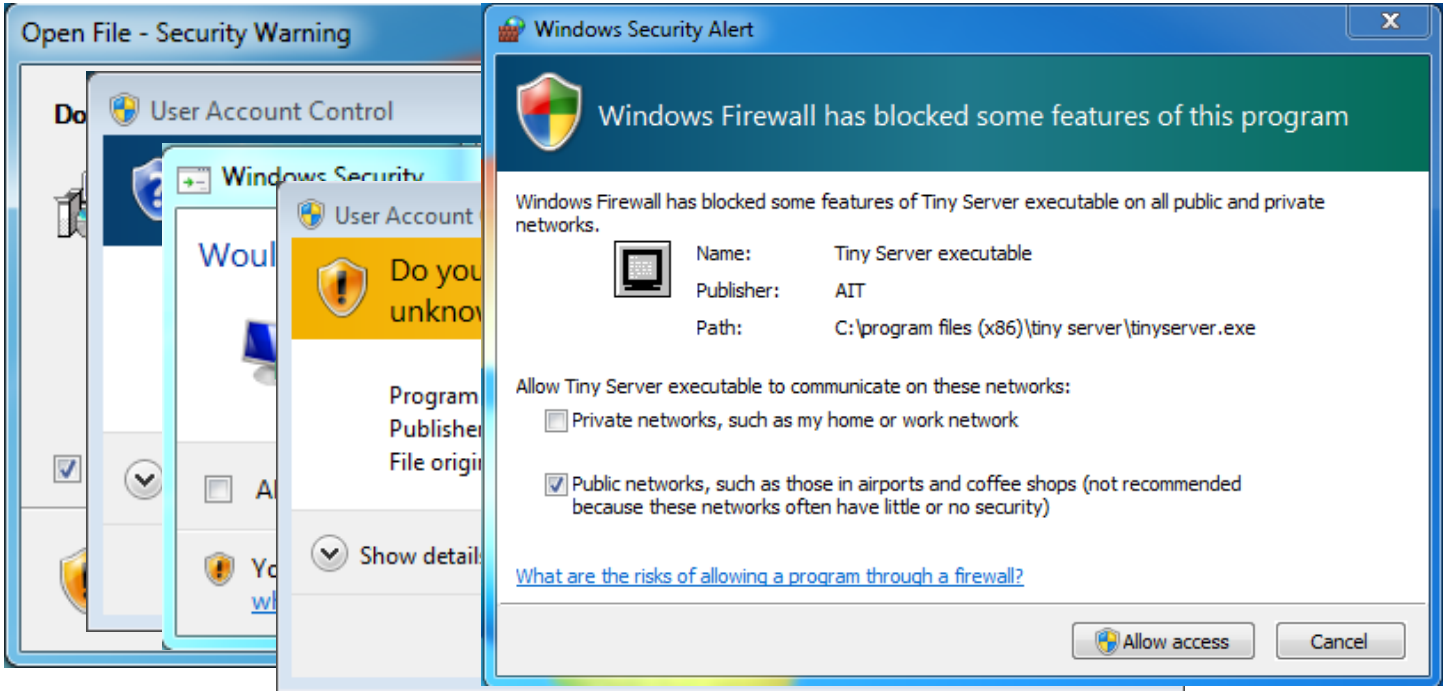
- Instead of finding malwares and stop then, list all known good/allowed programs and only run them.
- Typically deployed by enterprise, who can afford to maintain a list of allowed programs

CodeShield: Personalized Application Whitelisting

- Goal: Practical Application Whitelisting on Windows desktops
  - Give the user flexibility
    - Allow the user to add software to the whitelist
  - Maintain the security advantage of whitelisting
    - New software isn't automatically allowed onto whitelist
    - Protect against certain types of Social Engineering attacks
- Not designed to stop all infection
  - Make persistence harder
  - Prevent most current attacks
- Focus on usability
  - A key challenge of many security mechanisms is the ability for a typical user to understand and use it

Analysis of Existing Security Interface

- Users are asked questions they do not know how to answer and presented with info that is difficult to understand
- Users are asked to make a decision too often
- Users are made to passively respond and provided an easy and insecure way out



Design Principles

- Reduce – decrease the number of times users are asked to make a decisions
- Simplify – ask questions that a user can understand
- Safe – do not provide an easy and insecure way out.
- Active – avoid passively respond to security prompts

Design of Personalized Whitelisting

Normal Mode

- Only execute known software
- Trusted Signatures = add to whitelist
- Trusted Installers = add to whitelist
- All else blocked

Installation Mode

- Execute all software
- Executed = added to whitelist
- Written = added to whitelist
- Try to exit installation mode quickly

- "Stopping" vs "Warning" approach
- The decision a user needs to make
  - "Do I want to install new software now"

Design Principles in Practice

- Reduce – there is a single security decision to make for installing any application
- Simplify – this paradigm more closely matches how typical users understand their actions. “I’m adding something new”
- Safe – Not allowing new code is the easiest action
- Active – In order to add new software, the user needs to actively participate and initiate the action.

Installation Mode vs Normal Mode

- This dual mode can more closely match the mental model of a typical user.
  - Users may not understand “Do you want to allow this program to make changes”
  - But most can be educated about “Do you want to add something new to your computer right now”
- Furthermore, users can be educated about when not to enter installation mode.

The Burden Benefit of Installation Mode

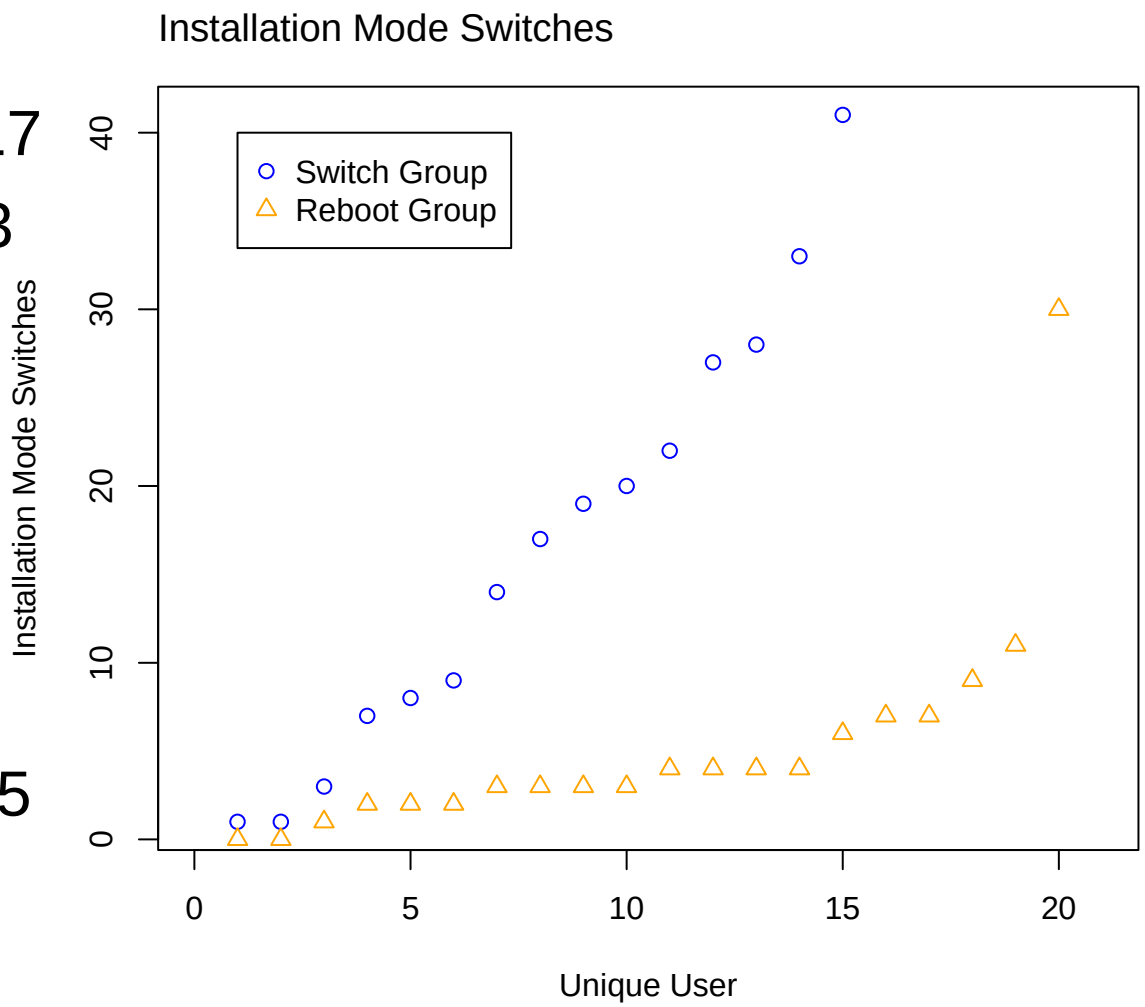
- Simple switch to installation mode
  - Advantage – it's easy
  - Disadvantage – user may enter installation mode often
- High overhead switch to installation mode (ex. reboot)
  - Advantage – it makes a user less likely to switch unless needed
  - Disadvantage – high overhead may lead to annoyance
- Advantage of reboot
  - Clear out memory, malware in memory can't take advantage of installation mode
  - Minimal number of applications active just after reboot

User Study

- 35 person user study running CodeShield for 6 weeks
- Longest use of CodeShield is 203 days (8 switches, 25 days/switch), next is 168 days (13 switches, 13 days/switch).
- Participants sat through a 30 minute training session
- Then installed CodeShield (standalone installer)
- Take a survey, Run for 6 weeks, Take a survey
- Uninstall if they want to
- 7 of 38 participants continued to use CodeShield at least 3 months after study ended.
  - 5 were using reboot only client
  - 2 using switch or reboot

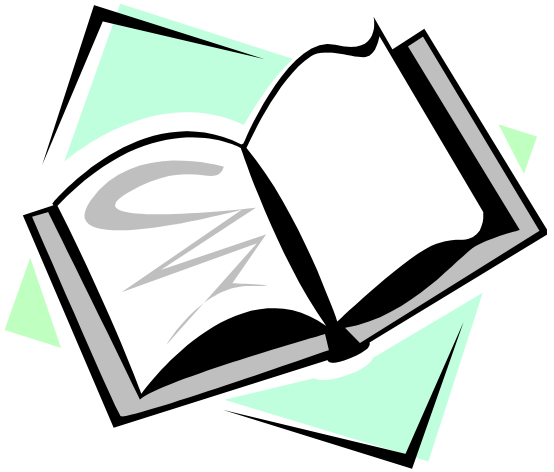
Switches to Installation Mode

- Switch
  - Median - 17
  - Useful - 13
- Reboot
  - Median - 3.5
  - Useful - 3.5



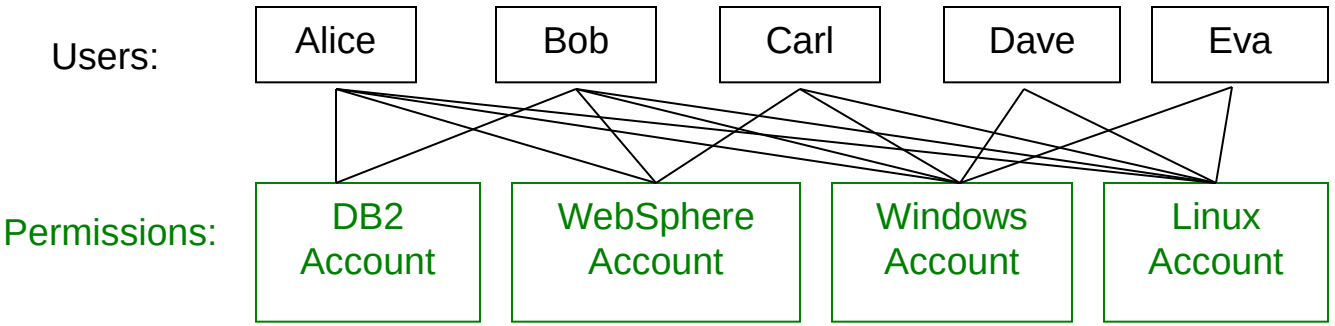
Readings for This Lecture

- RBAC96 Family
  - R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. “Role-Based Access Control Models”. *IEEE Computer*, 29(2):38--47, February 1996.

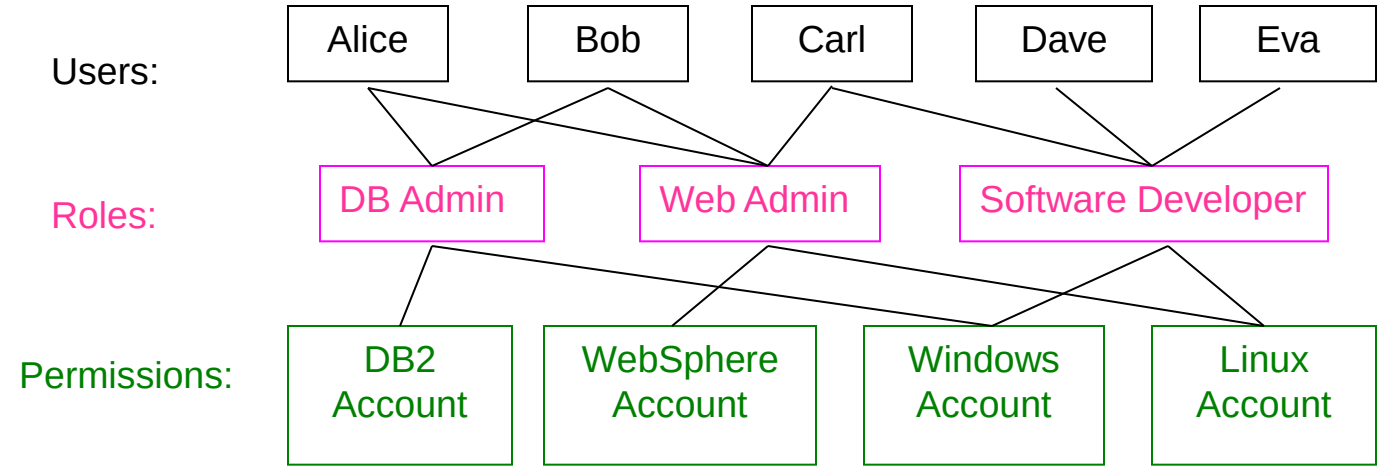


Background: Role Based Access Control

- Non-role-based systems



- Role-Based Access Control Systems (RBAC)



ROLE-BASED ACCESS CONTROL (RBAC)

- Motivating Problem: how to administer user-permission relation
  - Different from DAC and MAC, which deal with processes in operating systems
- Roles as a level of indirection
  - Butler Lampson or David Wheeler: "all problems in Computer Science can be solved by another level of indirection"
- RBAC is multi-faceted and open ended
  - Extensions: ARBAC (administrative), CBRAC (constraint), dRBAC (dynamic), ERBAC (enterprise), fRBAC (flexible), GRBAC (generalized), HRBAC (hierarchical), IRBAC (interoperability), JRBAC (Java), LRBAC (Location), MRBAC (Management), PRBAC (privacy), QRBAC (QoS), RRBAC(Rule), SRBAC(Spatial), TRBAC (temporal), V, W, x.
  - Non extension: OrBAC

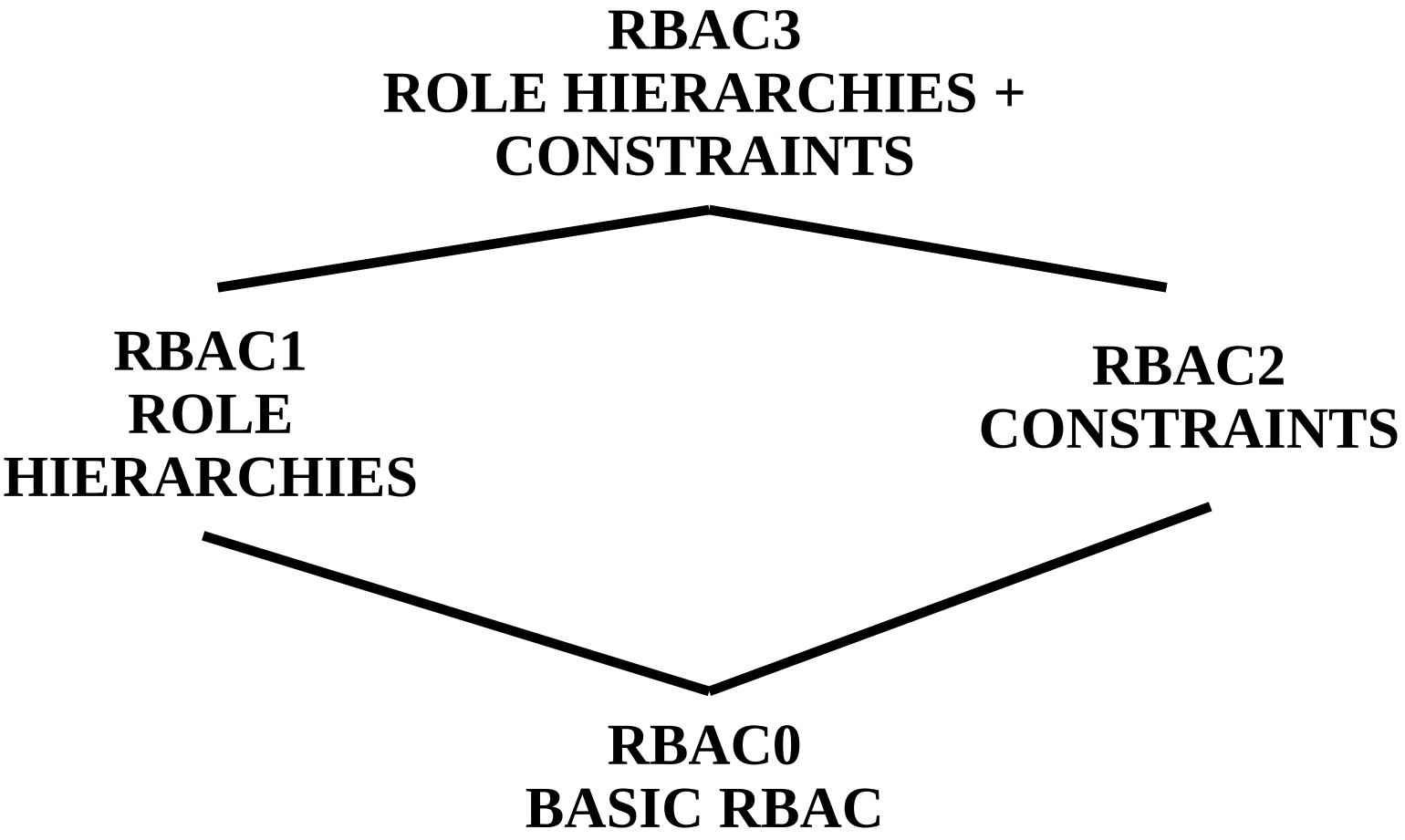
Why Roles?

- Fewer relationships to manage
  - possibly from O(mn) to O(m+n), where m is the number of users and n is the number of permissions
- Roles add a useful level of abstraction
- Organizations operate based on roles
- A role may be more stable than
  - the collection of users and the collection of permissions that are associated with it

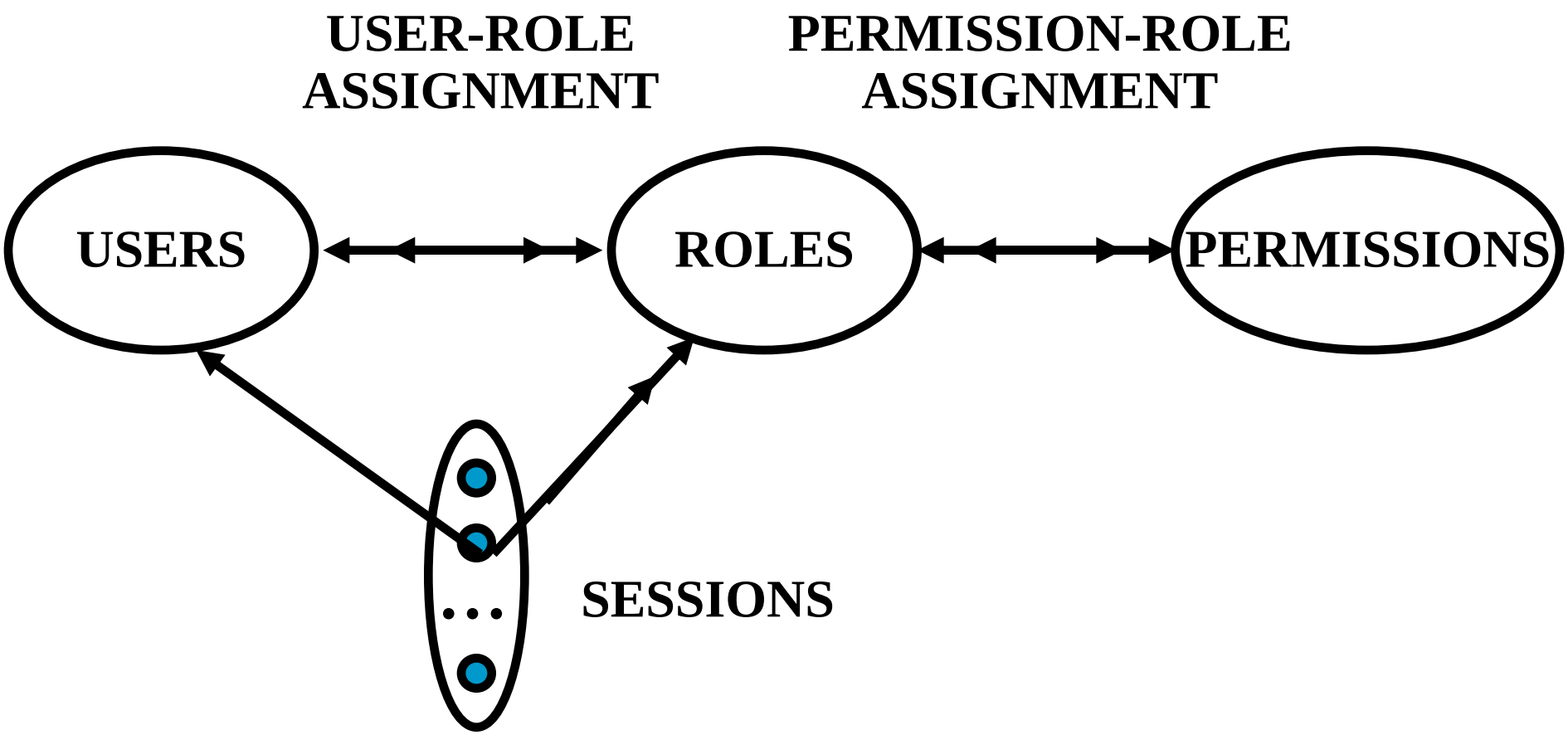
Groups vs. Roles

- Depending on the precise definition, can be the same or different.
- Some differences that may or may not be important, depending on the situation
  - Answer 1: sets of users vs. sets of users as well as permissions
  - Answer 2: roles can be activated and deactivated, groups cannot
    - Groups can be used to prevent access with negative authorization.
    - Roles can be deactivated for least privilege
  - Answer 3: can easily enumerate permissions that a role has, but not for groups

RBAC96 FAMILY OF MODELS (Sandhu et al.)



RBAC0


USER-ROLE ASSIGNMENT
PERMISSION-ROLE ASSIGNMENT
USERS
ROLES
PERMISSIONS
...
SESSIONS

PERMISSIONS

- Left abstract in the RBAC96 model

- Permissions are positive
- No negative permissions or denials
  - RBAC defines a closed policy, i.e., all accesses are denied unless they are explicitly authorized

- No duties or obligations
  - Example obligation: can access patient document, but must notify patient, or must delete after 30 days

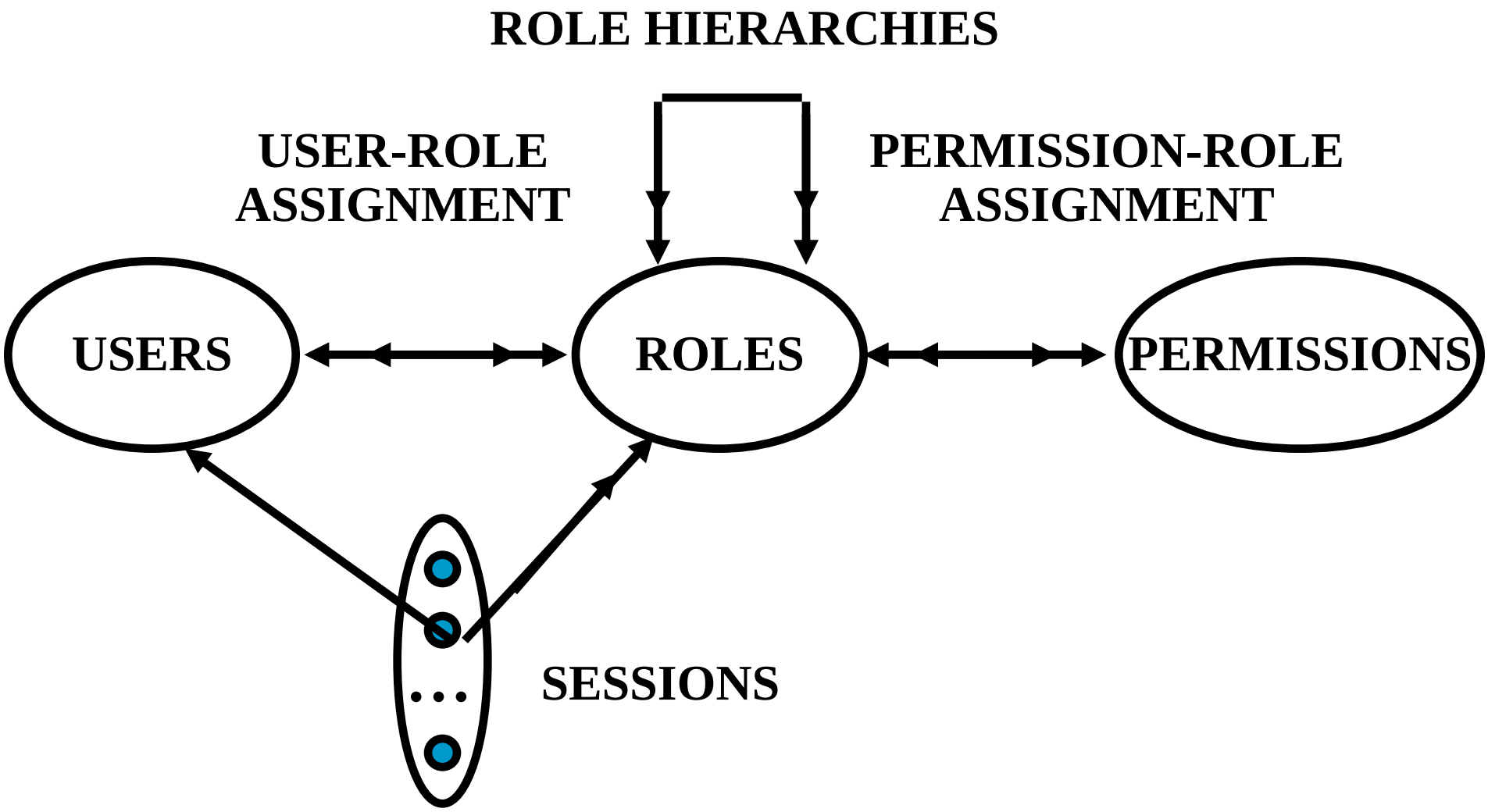
RBAC0: Formal Model

- Vocabulary: U, R, P, S (users, roles, permissions, and sessions)
- Static relations:
  - $PA \subseteq P \times R$ (permission assignment)
  - $UA \subseteq U \times R$ (user assignment)
- Dynamic relations:
  - user: $S \rightarrow U$ each session has one user
  - roles: $S \rightarrow 2^R$ and some activated roles
    - requires $roles(s) \subseteq \{ r \mid (user(s), r) \in UA \}$

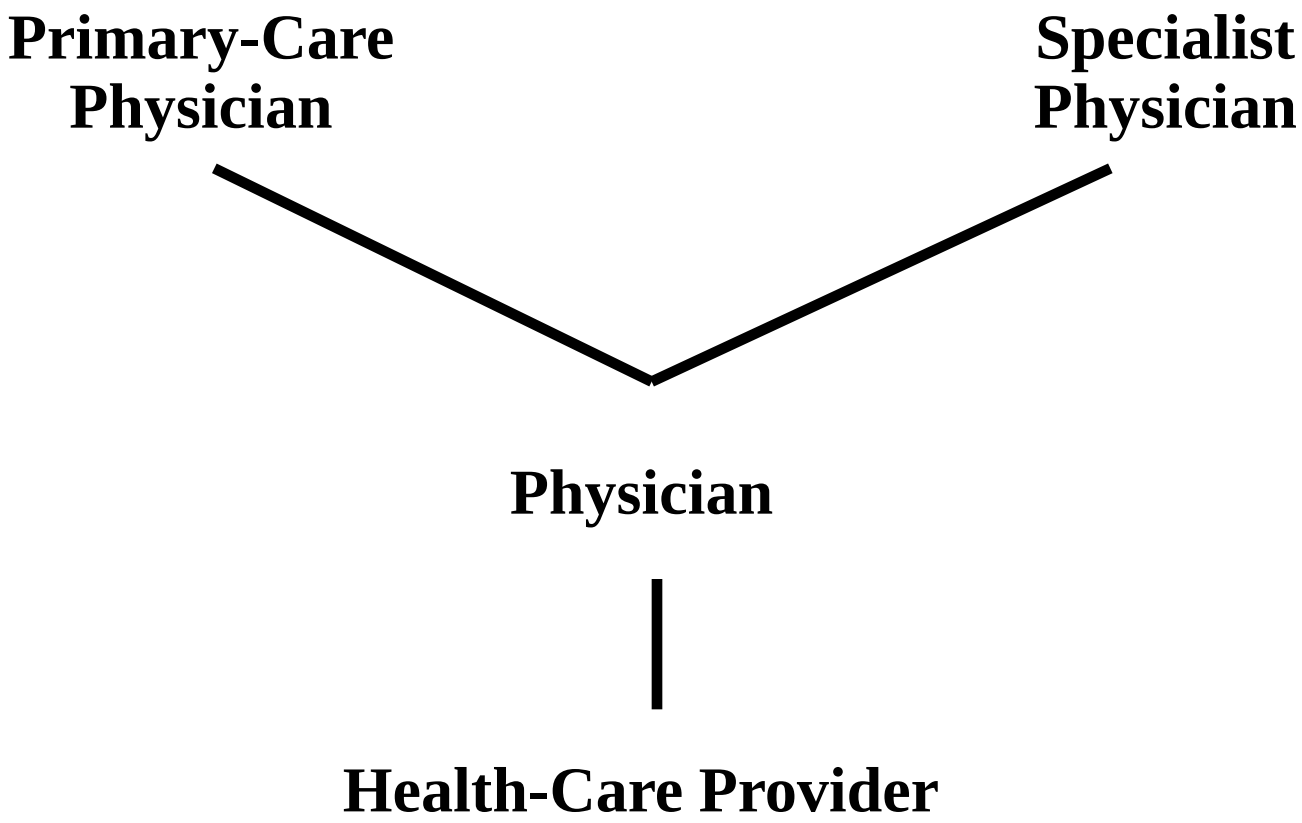
Session s has permissions

$$\bigcup_{r \in roles(s)} \{ p \mid (p, r) \in PA \}$$

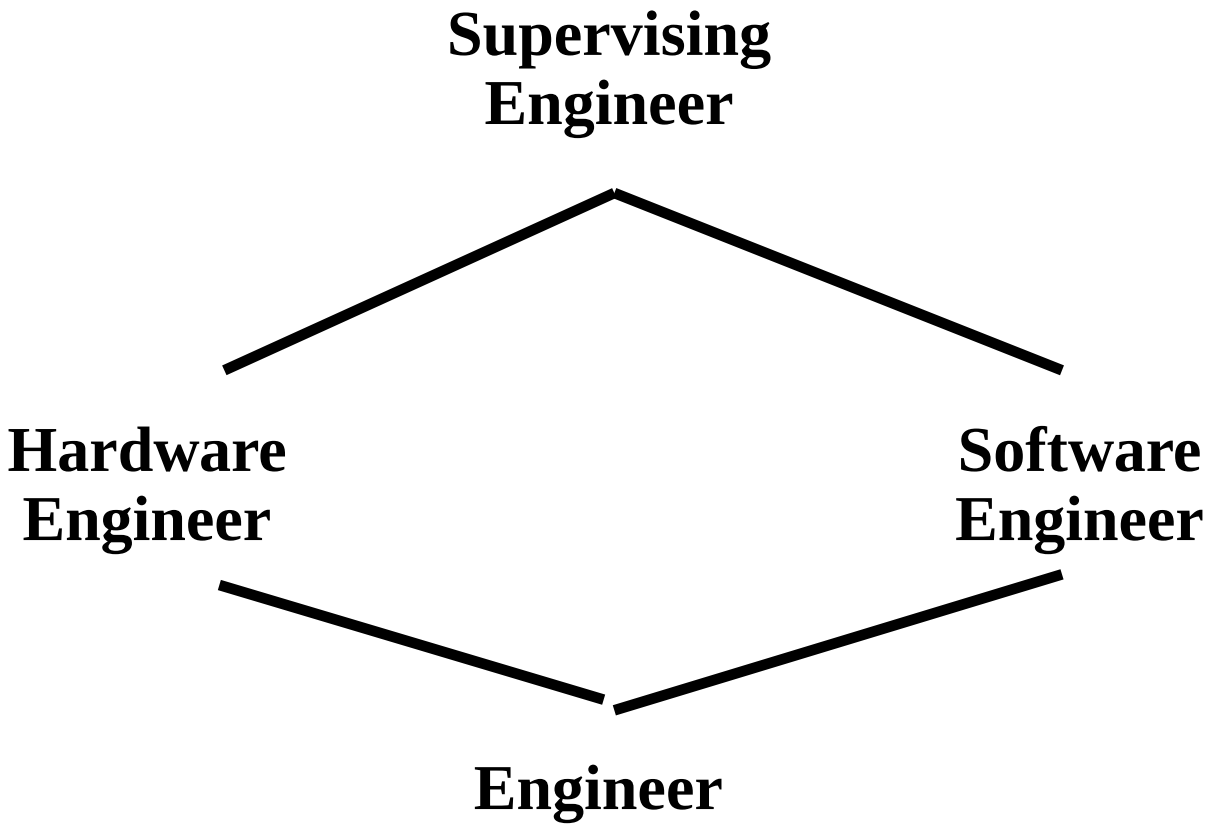
RBAC1


ROLE HIERARCHIES
USER-ROLE ASSIGNMENT
PERMISSION-ROLE ASSIGNMENT
USERS
ROLES
PERMISSIONS
SESSIONS

HIERARCHICAL ROLES (ex 1)

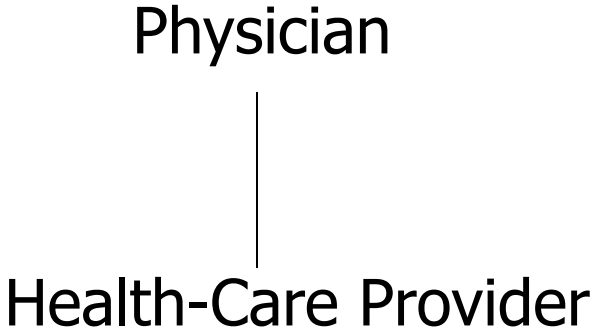


HIERARCHICAL ROLES (ex 2)



Semantics of Role Hierarchies

- User inheritance
  - r1≥r2 means every user that is a member of r1 is also a member of r2
- Permission inheritance
  - r1≥r2 means every permission that is authorized for r2 is also authorized r1
- Activation inheritance
  - r1≥r2 means that activating r1 will also activate r2



Permission and Activation inheritance have different effect when there are constraints about activation.

RBAC1: Formal Model

- U, R, P, S, PA, UA, and user unchanged from RBAC0
- RH $\subseteq$ R × R : a partial order on R, written as $\geq$
  - When r1 $\geq$ r2, we say r1 is a senior than r1, and r2 is a junior than r1
- roles: S $\rightarrow 2^R$
  - requires roles(s) $\subseteq$
    { r | $\exists$ r' [(r' $\geq$ r) & (user(s), r') $\in$ UA] }

Session s includes permissions

$$\bigcup_{r \in roles(s)} \{ p \mid \exists r'' [(r \geq r'') \& (p, r'') \in PA] \}$$

RBAC2: RBAC0 + Constraints

- No formal model specified
- Example constraints
  - Mutual exclusion
  - Pre-condition: Must satisfy some condition to be member of some role
    - E.g., a user must be an undergrad student before being assigned the UTA role
  - Cardinality

Mutual Exclusion Constraints

- Mutually Exclusive Roles
  - Static Exclusion: No user can hold both roles
    - often referred to as Static Separation of Duty constraints
    - Preventing a single user from having too much permissions
  - Dynamic Exclusion: No user can activate both roles in one session
    - Often referred to as Dynamic Separation of Duty constraints
    - Interact with role hierarchy interpretation

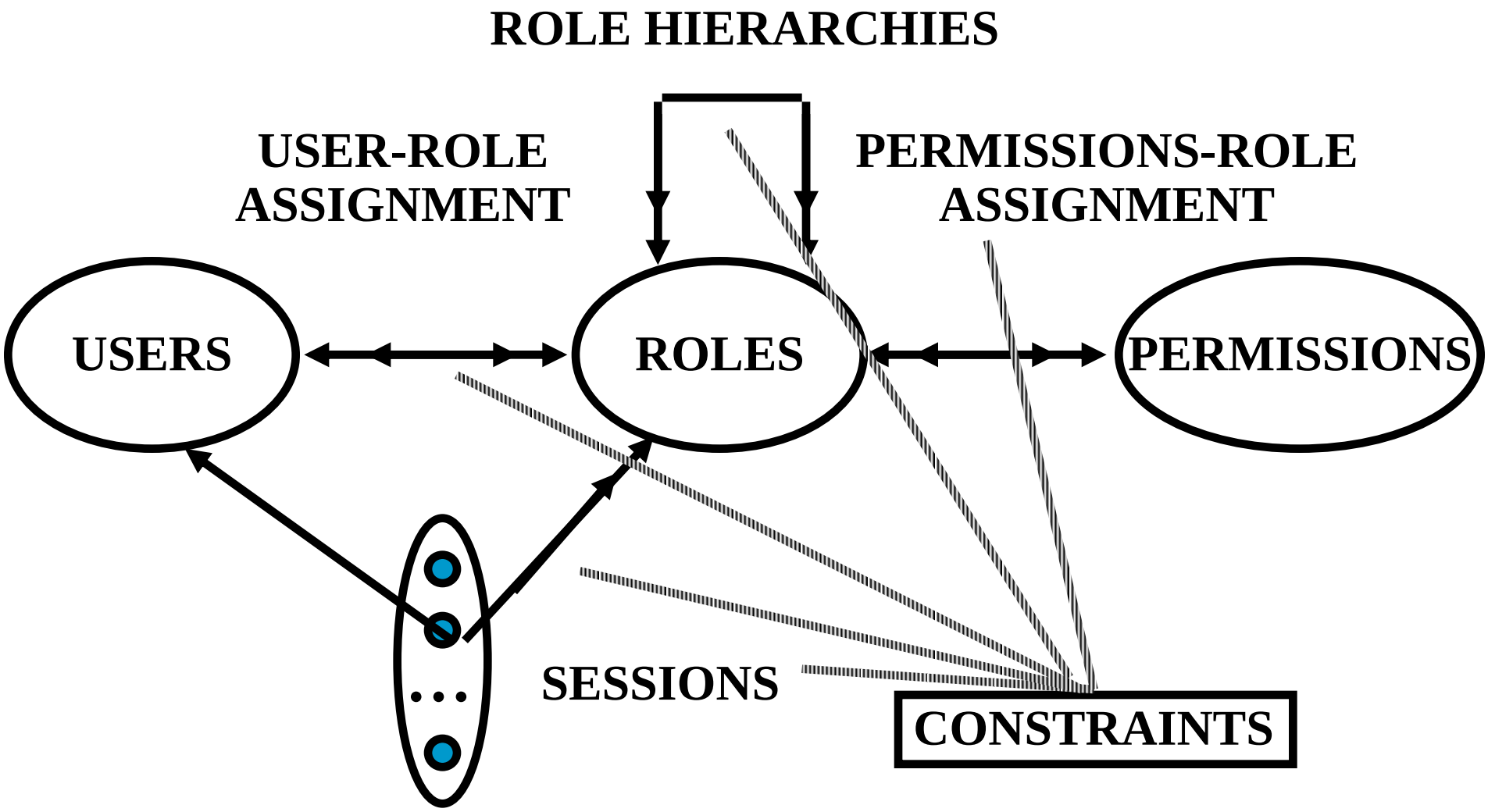
Cardinality Constraints

- On User-Role Assignment
  - at most k users can belong to the role
  - at least k users must belong to the role
  - exactly k users must belong to the role
- On activation
  - at most k users can activate a role
  - …

Why Using Constraints?

- For laying out higher level organization policy
  - Only a tool for convenience and error checking when admin is centralized
    - Not absolutely necessary if admin is always vigilant, as admin can check all organization policies are met when making any changes to RBAC policies
  - A tool to enforce high-level policies when admin is decentralized

RBAC3


ROLE HIERARCHIES
USER-ROLE ASSIGNMENT
PERMISSIONS-ROLE ASSIGNMENT
USERS
ROLES
PERMISSIONS
SESSIONS
CONSTRAINTS

Products Using RBAC

- Data Base Management Systems (DBMS)
- Enterprise Security Management
  - IBM Tivoli Identity Manager (central administration and provisioning of accounts, resources, etc)
- Many operating systems claim to use roles

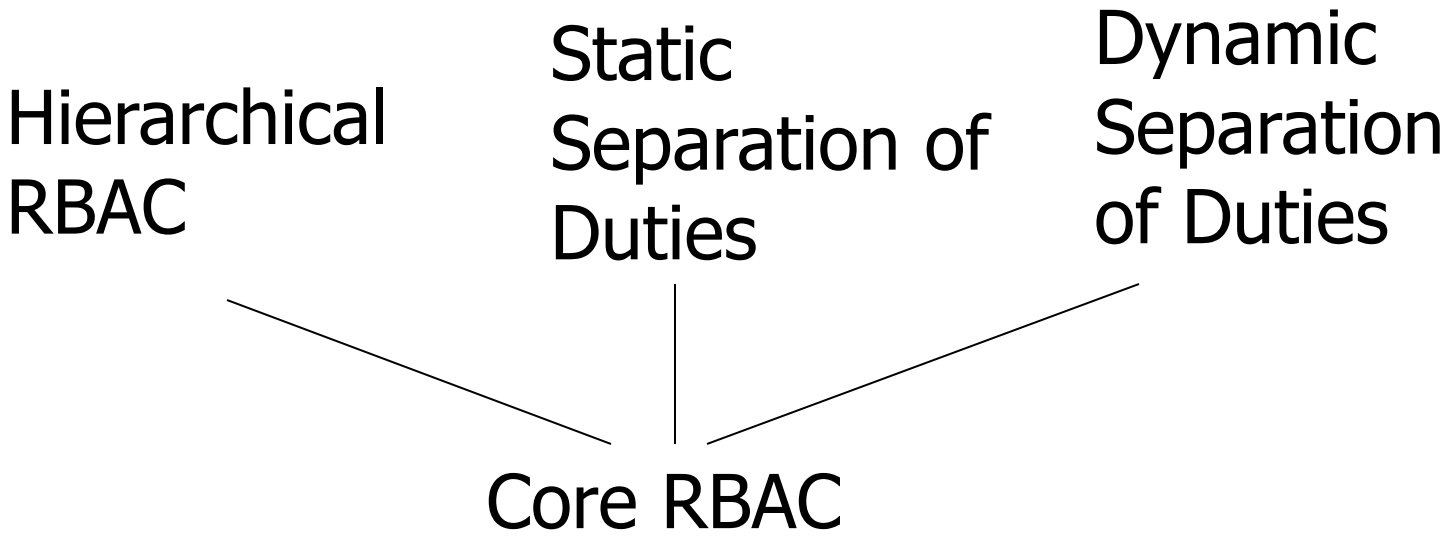
RBAC Economic Impact Study in 2002

- Based on interviews with software developers and companies that integrate RBAC products into their business operations (end users), the Research Triangle Institute (RTI) estimates that by 2006 between 30 and 50 percent of employees in the service sector and between 10 and 25 percent of employees in the non-service sectors will be managed by RBAC systems. RTI also estimates that this degree of market penetration will result in economic benefits to the U.S. economy through 2006 of approximately $671 million in net present value terms. This estimate is conservative because it reflects only the administrative and productivity benefits from RBAC.

The NIST Standard

- Proposed NIST Standard for Role-Based Access Control. David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. TISSEC, August 2001.
- American National Standards Institute Standard, 2004

Overview of the NIST Standard for RBAC



Our Critique of the ANSI RBAC Standard

- Many errors
  - Inheritance has been described in terms of permissions; i.e., r1 inherits r2 if all privileges of r2 are also privileges of r1. . . .
    - mistake in cause-effect relationship
  - define permission inheritance as “formally, authorized_permissions(r) = {p $\in$ PRMS | r′ $\geq$ r, (p, r′) $\in$ PA}.”
    - should be r $\geq$ r’
  - The standard defines *r*1 >> *r*2 (r1 is immediate parent role of r2) when “there’s no role *r*3 in the role hierarchy such that *r*1 $\geq$ *r*3 $\geq$ *r*2, where *r*1 $\neq$ *r*2 and *r*2 $\neq$ *r*3”
    - should be r1 $\neq$ *r*3
- A number of other limitations and design flaws

Our Suggestions for Improving ANSI RBAC Standard

- Remove sessions from core RBAC
- Accommodate single-role sessions
- Clearly distinguish based and derived relations
- Maintain role-domination relationships explicitly
- Clearly specify role-inheritance semantics

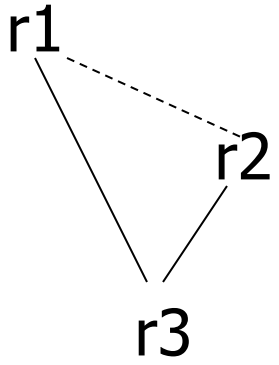
Whether to Allow Multiple Roles to be Activated?

- RBAC96 allows this Multi Role Activation
- [Baldwin'90] does not
- Observations:
  - one can define new role to achieve the effect of activating multiple roles
  - dynamic constraints are implicit when only one role can be activated in a session
  - Single-Role Activation is better
    - easier to enforce least privilege
    - better satisfies the fail-safe defaults principle

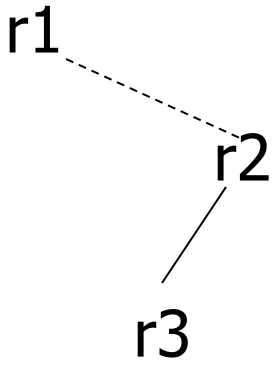
On Modeling Role Hierarchy As A Partial Order

- Modeling RH as a partial order may miss some important information
- Consider the two examples to the right
  - where the dashed edge is added and removed
- Better approach seems to remember the base edges and then compute their transitive and reflexive closure

EX1:

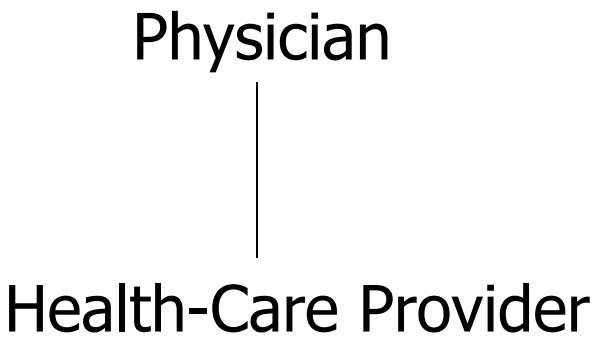


EX2:



Semantics of Role Hierarchies

- User inheritance
  - r1≥r2 means every user that is a member of r1 is also a member of r2
- Permission inheritance
  - r1≥r2 means every permission that is authorized for r2 is also authorized r1
- Activation inheritance
  - r1≥r2 means that activating r1 will also activate r2



They interact with static and dynamic role mutual exclusion constraints.

Coming Attractions …

- Database access control