

# Access Control

## Role-based Access Control

---

Prof.dr. Ferucio Laurențiu Tiplea

Fall 2022

Department of Computer Science  
"Alexandru Ioan Cuza" University of Iași  
Iași 700506, Romania

e-mail: [ferucio.tiplea@uaic.ro](mailto:ferucio.tiplea@uaic.ro)

# Outline

Introduction to RBAC

Base RBAC model

Hierarchical RBAC model

Constrained RBAC

Consolidated RBAC

RBAC, DAC, and MAC

RBAC Implementations

Concluding remarks

# Introduction to RBAC

---

# Role-based Access Control

**RBAC:** Access to objects is regulated by the role user has in an organization.

A bit of history:

- Formal approaches to access control were initiated in the early 1970s through DAC and MAC:
  - MAC was introduced based on military security needs, which sometimes makes it unsuitable for civilian applications. For example, MAC cannot properly enforce access control based on competencies, conflict-of-interest rules, or a strict concept of least privilege;
  - DAC has a focus on civilian applications, but it assumes that subjects own the resources. However, the resources are not owned by subjects in companies (organizations, corporations, agencies, enterprises);

# Role-based Access Control

A bit of history (cont.):

- A solution to meet these needs was proposed by Ferraiolo and Kuhn (1992) in the form of a non-discretionary access control model called [role-based access control](#) (RBAC);
- Later, Sandhu et al. (1996) introduced a framework for RBAC models, commonly known as [RBAC96](#), also establishing a modular structure;
- Following a NIST initiative to reach an international consensus for the use of RBAC, Sandhu et al. (2000) proposed an RBAC standard. The standard was approved in 2004 and adopted as INCITS 359-2004 ([NIST model for RBAC](#));
- The revision initiated in 2010 by NIST adds attributes to RBAC (Kuhn et al. (2010)). The latest version of the standard is INCITS 359-2012.

## Base RBAC model

---

# RBAC components

An RBAC model can be viewed as consisting of two separate but dependent classes of components:

- **Static components:** users, roles, permissions, and the relations between them;
- **Dynamic components:** subjects, role authorization, and object access authorization.

# Users, roles, permissions

**User** – any person who interacts directly with a computer system;

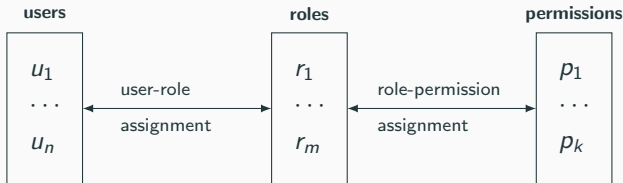
**Role** – a set of actions and responsibilities associated with a particular working activity;

**Permission** – a description of (a type of) authorized access to resources or to do something:

- The RBAC model accommodates many interpretations for permissions. The nature of a permission depends on system and implementation;
- **Permissions are positive**: they confer the ability to perform actions in system;
- Constraints will model the prohibition to execute an action (some authors name them **negative permissions**).



# The static components of an RBAC model



- $U$ : set of users;
- $R$ : set of roles;
- $P$ : set of permissions;
- $UR \subseteq U \times R$ : **user-role assignment**;
- $RP \subseteq R \times P$ : **role-permission assignment**.

We may view a permission as an abstract concept binding operations and objects

$$p \subseteq Op \times O,$$

where  $Op$  is a set of operations and  $O$  is a set of objects.

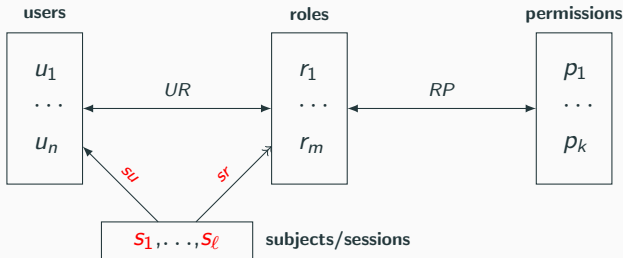
# Roles vs. groups and compartments

- **Organizations operate based on roles:** identifying roles (as sets of permissions) in a system is more priority than identifying groups (as sets of users). This leads to the advantage of simplifying the understanding and management of permissions in the system. For instance, enumerating all permissions for roles is easy;
- **Roles add a useful level of abstraction** and may be more stable than groups;
- **Groups are implementation-specific.** For example, a file can be associated with only one group in some operating systems, while in others, a file can be associated with several groups. But the role is defined by permissions offered to users who receive it, which makes the role independent of implementation;
- The compartments in a lattice-based model allow one-way information flow based on a specific policy. Roles are not associated with any policy.

# Subjects and sessions

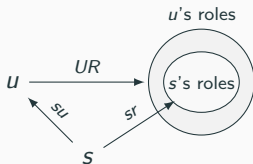
- A **session** is an instance of a user's dialog with a system;
- A subject is a running computer program (process) acting on behalf of a user;
- As user's actions on a computer system are performed through some programs running on the computer, we **often identify subjects and sessions**.

# The dynamic components of an RBAC model



subject-to-user mapping  $su : S \rightarrow U$

subject-to-role mapping  $sr : S \rightarrow \mathcal{P}(R)$



$$sr(s) \subseteq UR(su(s))$$

# Role authorization and object access authorization

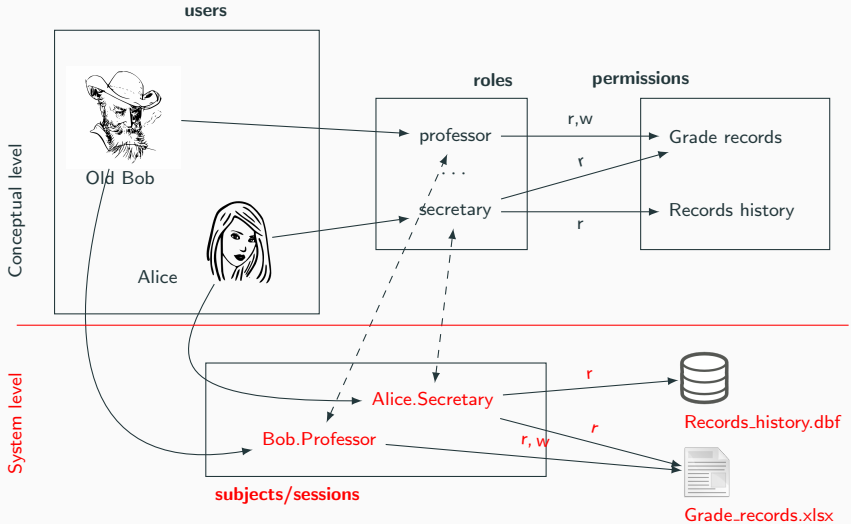
**Role authorization:** a subject can never have an active role that is not authorized for its user

$$(\forall s \in S)(\forall r \in R)(r \in sr(s) \Rightarrow (su(s), r) \in UR)$$

**Object access authorization:** A subject  $s$  can perform an operation  $op$  on object  $o$  only if there exists a role  $r$  that is included in the subject's active role set and there exists a permission that is assigned to  $r$  such that the permission authorizes the performance of  $op$  on  $o$

$$access(s, op, o) \Rightarrow (\exists r \in R)(\exists p \in P)(r \in sr(s) \wedge (r, p) \in PR \wedge (op, o) \in p)$$

# Conceptual level vs. system level

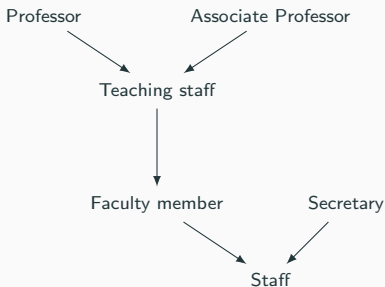


## **Hierarchical RBAC model**

---

# Role hierarchies

Individual roles within an organization often have overlapping functions!



- A **role inheritance relation/scheme** is a partial order relation  $\geq$  on the set  $R$  of roles;
- If  $r_1 \geq r_2$  we say that  $r_1$  **inherits**  $r_2$ .



# Inheritance schemes

Role inheritance schemes may be defined in various ways, such as:

- **Permission-based inheritance** – here, the role is viewed just as a set of permissions. Then,

$$r_1 \geq r_2 \Leftrightarrow RP(r_2) \subseteq RP(r_1)$$

- **User- and permission-based inheritance** – here, the role is viewed both as a set of permissions and a set of users. Then,

$$r_1 \geq r_2 \Leftrightarrow RP(r_2) \subseteq RP(r_1) \wedge UR^{-1}(r_1) \subseteq UR^{-1}(r_2)$$

- **User-based inheritance** – here, permissions are assigned to groups and groups are mapped to roles. Then,

$$r_1 \geq r_2 \Leftrightarrow UR^{-1}(r_1) \subseteq UR^{-1}(r_2)$$

# Role authorized users and permissions

An **user**  $u$  is authorized for a role  $r$  if  $u$  has assigned a role  $r'$  that inherits  $r$ . The set of **users authorized for the role**  $r$  is

$$\{u \in U \mid \exists r' : (u, r') \in UR \wedge r' \geq r\}.$$

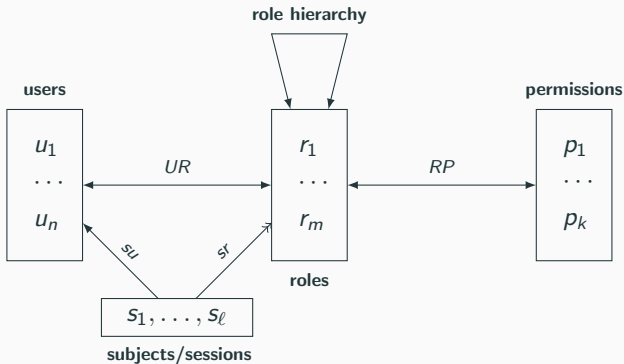
Users with more powerful roles may play less powerful roles!

A **permission**  $p$  is authorized for a role  $r$  if  $p$  is assigned to a role  $r'$  inherited by  $r$ . The set of **permissions authorized for the role**  $r$  is

$$\{p \in P \mid \exists r' : (r', p) \in RP \wedge r' \leq r\}.$$

Permissions of less powerful roles are available to more powerful roles!

# The hierarchical RBAC model



# Constrained RBAC

---

# Constraints

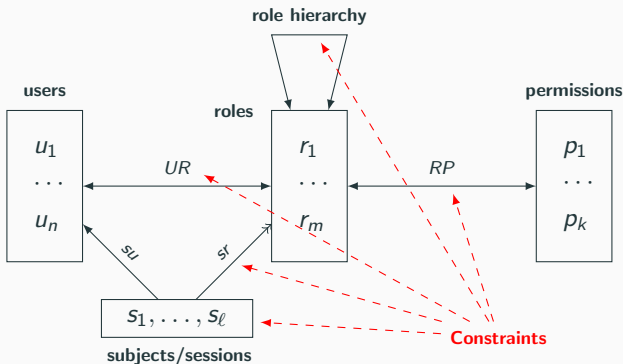
Constraints play an essential role in shaping a system. They can

- simplify the design;
- create a better view on the system;
- bring more power.

In RBAC, constraints play a significant role. They can help

- create **roles** or **permissions with mutual exclusion**;
- control the **assignment of roles per user** or **permissions per role**:
  - number of roles/permissions;
  - prerequisite roles/permissions;
  - etc.

# Enforcing constraints



**Static constraints** – applied to the static components ( $UR$ ,  $RP$ , role hierarchy)

**Dynamic constraints** – applied to the dynamic components (sessions,  $sr$ )

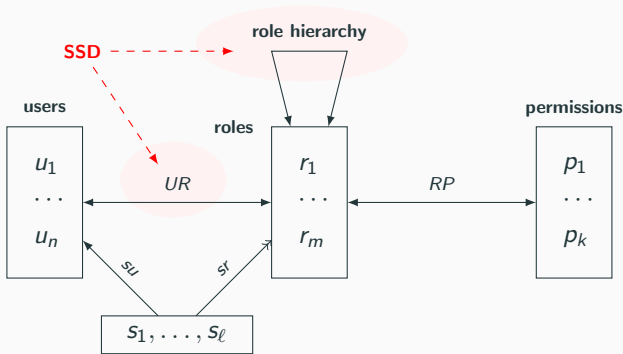
# Separation of duty

American National Standards Institute: [Separation of Duty](#) (SoD) means “Dividing responsibility for sensitive information so that no individual acting alone can compromise the security of the data processing system.”

There exists a great variety of SoD models (see Simon and Zurko (1997)). Two broad categories of SoD methods are:

- [static](#) (SSD) – place constraints on roles at the time users are authorized for roles, and
- [dynamic](#) (DSD) – are invoked when the users are actively using the system.

# Static SoD

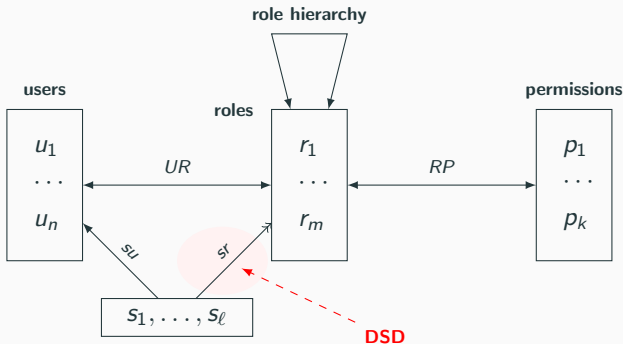


## Example 1

Professor Bob cannot be a member of both the **Examination Board** and **Appeal Examination Board**.



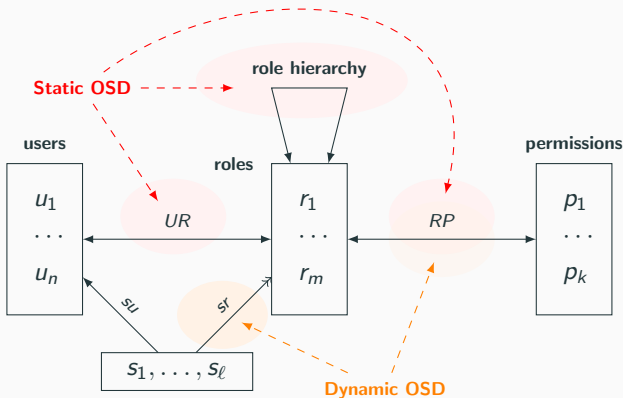
# Dynamic SoD



## Example 2

Bob can be a member both of the Examination Board and Appeal Examination Board, but **not both roles can be active in the same session**.

# Operational SoD



**Operational SoD (OSD)** – no single user is allowed to perform all operations required to a critical function.

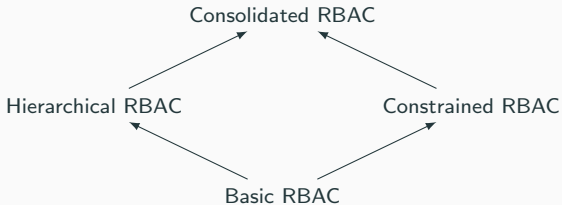
OSD can be enforced as an SSD/DSD with additional constraints on  $RP$ .

# Consolidated RBAC

---

# Consolidated RBAC

Consolidated RBAC combines hierarchical and constrained RBAC

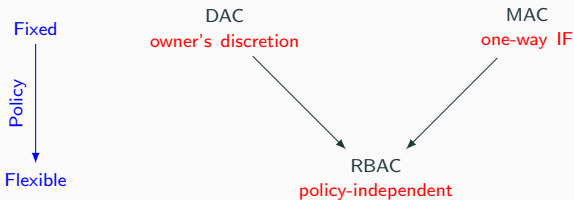


# **RBAC, DAC, and MAC**

---

# RBAC, DAC, and MAC

RBAC is neither DAC nor MAC! However, RBAC can be configured to do both DAC and MAC (details, which are more of a theoretical interest, can be found in Osborn et al. (2000)).



In practice, we never simulate DAC or MAC through RBAC (it is too costly). However, we often combine DAC, MAC, and RBAC.

# RBAC Implementations

---

# Integrating RBAC with enterprise IT infrastructures

RBAC can be integrated with technologies such as:

- Workflow management systems (Kandala and Sandhu (2001));
- Web applications (Park et al. (2001));
- Windows and UNIX OS;
- Distributed or network file systems (Dekker et al. (2008));
- Java;
- Oracle API Gateway, which is a comprehensive platform for managing, delivering, and securing Web APIs, uses the RBAC model. It is available on Windows, Linux, and Solaris;
- Microsoft Azure RBAC, which provides fine-grained access to Azure resources (see Microsoft Azure RBAC doc);
- Windows 365 (see Microsoft Windows 365 Enterprise doc);
- etc.



# RBAC in Windows 365

Microsoft | Docs Documentation Learn Q&A Code Sample

Docs / Windows 365 / Windows 365 Enterprise

Filter by title

Windows 365 Enterprise docs

- > Overview
- > Concepts
- ▼ How-to guides
  - ▼ Plan
    - Planning guide
    - > Requirements
    - Cloud PC role-based access control**
    - Intune role-based access control
    - Cloud PC sizes
  - > Deploy
  - > Provisioning
  - > Profiles and policies
  - > User settings
  - > Security
  - > Apps

## Role-based access control

11/04/2021 • 2

Role-based access control allows you to manage permissions for users and groups. For more info, see [Role-based access control in Windows 365 Enterprise](#).

## Windows 365

Windows 365 is a cloud-based operating system that allows you to run Windows applications and desktops in the cloud. It is designed to be secure, scalable, and easy to manage. For more info, see [Windows 365 overview](#).

## Cloud PC

Cloud PC is a virtual machine that runs in the cloud and is accessible from any device. It is designed to be secure, scalable, and easy to manage. For more info, see [Cloud PC overview](#).

Microsoft | Docs Documentation Learn Q&A

Azure Product documentation ▼ Architecture ▼ Learn Azure


Azure / Role-based access control

Filter by title

## Azure RBAC documentation

- > Overview
- > Quickstarts
- > Tutorials
- > Concepts
- > How-to guides
- > Reference
- > Resources

# RBAC UNIX-like OSs

 Documentation Search in AIX 7.1

## AIX

Change version

7.1

☐ Show full table of contents

Filter on titles

### Security

- What's new
- Securing the base operating system
  - Secure system installation and configuration
  - Users, groups, and passwords
  - Role-based access control**
  - Traditional UNIX administration limitations
  - Elements of RBAC
  - AIX RBAC
  - Using Enhanced RBAC


## AIX / 7.1 /

# Role-based access control

System administrator addition to security

Most environments that no single user can achieve these goals:

- Traditional UNIX administration
- Elements of RBAC
- AIX RBAC
- Using Enhanced RBAC
- RBAC-related
- RBAC-related
- Using enhanced RBAC

 gentoo linux® Wiki

Main page Recent changes Help Gentoo Documentation

Page Discussion

## SELinux/Role-based access control

< SELinux

To provide segregation of duties and privileges based on the *least privilege* model, are not allowed to perform tasks of another role (segregation of duties), and additionally not get too many privileges.

### Contents [hide]

- 1 Introduction
- 2 RBAC in SELinux
  - 2.1 Assigning roles to users
  - 2.2 Assigning permissions to roles and domains
  - 2.3 Permissions or domain transitions
- 3 Default roles

## Introduction

In a true RBAC situation, people are assigned one or more roles which grant or deny permissions. In an RBAC model, there are a couple of important aspects to have in an implementation:

- Permissions are always granted through roles - no direct assignment to users
- Users must be explicitly granted roles - no role, no rights

RBAC by itself is not all that hard to implement. On a Linux system, one can make and group privileges.

## RBAC in SELinux

The implementation of *Role Based Access Control (RBAC)* in SELinux is as follows.

## **Concluding remarks**

---

## Concluding remarks

- RBAC simplifies security administration by using roles, hierarchies, and constraints;
- RBAC reduces costs within an organization because it takes into account that employees change much more frequently than the duties within positions;
- RBAC can be configured to support a large variety of access control policies, including DAC and MAC policies;
- RBAC is suited to a large variety of applications and software system environments.

# Readings

In addition to the materials indicated so far, I recommend:

- Ferraiolo et al. (2007);
- Chapters 6 and 7 of Conrad et al. (2016);
- Chapters 3 and 4 of Andress (2014);
- Chapter 11 of Collins (2014);
- Chapter 23 of Bertino (2012);
- Samarati and de Capitani di Vimercati (2001).

## References

---

- Andress, J. (2014). *The Basics of Information Security. Understanding the Fundamentals of Infosec in Theory and Practice*. Syngress, Elsevier, Boston, 2nd edition.
- Bertino, E. (2012). Chapter 23 - Policies, access control, and formal methods. In Das, S. K., Kant, K., and Zhang, N., editors, *Handbook on Securing Cyber-Physical Critical Infrastructure*, pages 573–594. Morgan Kaufmann, Boston.
- Collins, L. (2014). Chapter 11 - Access controls. In Vacca, J. R., editor, *Cyber Security and IT Infrastructure Protection*, pages 269–280. Syngress, Boston.
- Conrad, E., Misenar, S., and Feldman, J. (2016). *CISSP Study Guide*. Singress, Elsevier, 3rd edition.
- Dekker, M., Crampton, J., and Etalle, S. (2008). Rbac administration in distributed systems. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT '08*, page 93–102, New York, NY, USA. Association for Computing Machinery.
- Ferraiolo, D. and Kuhn, R. (1992). Role-based access controls. In *15th National Computer Security Conference Proceedings "Information Systems Security: Building Blocks to the Future"*, pages 554–563, Baltimore, Maryland, US. National Institute of Standards and Technology, National Computer Security Center.

## References (cont.)

- Ferraiolo, D. F., Kuhn, D. R., and Chandramouli, R. (2007). *Role-Based Access Control*. Artech House, Inc., USA, 2nd edition.
- Kandala, S. and Sandhu, R. S. (2001). Secure role-based workflow models. In *DBSec*.
- Kuhn, D. R., Coyne, E. J., and Weil, T. R. (2010). Adding attributes to role-based access control. *Computer*, 43(6):79–81.
- Osborn, S., Sandhu, R., and Munawar, Q. (2000). Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security*, 3(2):85–106.
- Park, J. S., Sandhu, R., and Ahn, G.-J. (2001). Role-based access control on the web. *ACM Trans. Inf. Syst. Secur.*, 4(1):37–71.
- Samarati, P. and de Capitani di Vimercati, S. (2001). Access control: Policies, models, and mechanisms. In Focardi, R. and Gorrieri, R., editors, *Foundations of Security Analysis and Design*, pages 137–196, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Sandhu, R., Ferraiolo, D., and Kuhn, R. (2000). The NIST model for role-based access control: Towards a unified standard. In *Proceedings of the Fifth ACM Workshop on Role-Based Access Control*, RBAC '00, page 47–63, New York, NY, USA. Association for Computing Machinery.
- Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.
- Simon, R. and Zurko, M. (1997). Separation of duty in role-based environments. In *Proceedings 10th Computer Security Foundations Workshop*, pages 183–194.