Attribute-Based Access Control (ABAC)

Nicola Zannone
Role-Based Access Control

The RBAC Shift

From **DAC**
- User-permission assignment

To **RBAC**
- User-role assignment
- Permission-role assignment
Role-Based Access Control (RBAC)

- **RBAC<sub>0</sub>:** core model
- **RBAC<sub>1</sub>:** RBAC<sub>0</sub> + role hierarchies
- **RBAC<sub>2</sub>:** RBAC<sub>0</sub> + constraints
- **RBAC<sub>3</sub>:** RBAC<sub>1</sub> + RBAC<sub>2</sub>
Role Engineering

1. Role identification
2. Permission identification
   - Least privilege
   - Separation of duties
Benefits of RBAC (1)

Easy administration of access rights

40 assignments

25 assignments
Benefits of RBAC (1)

case#1: 5 users + 5 resources = 25 assignments

case#2: 100 users + 100 resources = 10000 assignments

---

case#1: 5 users + 5 resources = 10 assignments

case#2: 100 users + 100 resources = 200 assignments
Benefits of RBAC (2)

Hierarchy of roles

- Employee
  - Manager
    - Store Manager
    - Office Manager
    - Sales Manager
Benefits of RBAC (3)

Separation of duty
The following access matrix has been generated from an RBAC policy with the given hierarchy and where C has role $r_4$. Give the minimal User-Assignment and Permission-Assignment corresponding to the given Access Matrix such that you do not assign anything that can be already derived otherwise.

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
<th>$p_5$</th>
<th>$p_6$</th>
<th>$p_7$</th>
<th>$p_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>B</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>E</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>F</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Hint:** Users might have more than one role.
## Homework – Solution

### User Assignment

<table>
<thead>
<tr>
<th>User</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$r_2, r_6$</td>
</tr>
<tr>
<td>B</td>
<td>$r_5$</td>
</tr>
<tr>
<td>C</td>
<td>$r_4$</td>
</tr>
<tr>
<td>D</td>
<td>$r_6$</td>
</tr>
<tr>
<td>E</td>
<td>$r_3, r_4$</td>
</tr>
<tr>
<td>F</td>
<td>$r_3$</td>
</tr>
<tr>
<td>G</td>
<td>$r_2$</td>
</tr>
</tbody>
</table>

### Permission Assignment

<table>
<thead>
<tr>
<th>Role</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td></td>
</tr>
<tr>
<td>$r_2$</td>
<td>$p_1, p_3$</td>
</tr>
<tr>
<td>$r_3$</td>
<td>$p_2$</td>
</tr>
<tr>
<td>$r_4$</td>
<td>$p_5, p_7$</td>
</tr>
<tr>
<td>$r_5$</td>
<td>$p_6$</td>
</tr>
<tr>
<td>$r_6$</td>
<td>$p_4, p_8$</td>
</tr>
</tbody>
</table>
Homework – Solution

User Assignment

<table>
<thead>
<tr>
<th>User</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$r_2, r_6$</td>
</tr>
<tr>
<td>B</td>
<td>$r_5$</td>
</tr>
<tr>
<td>C</td>
<td>$r_4$</td>
</tr>
<tr>
<td>D</td>
<td>$r_6$</td>
</tr>
<tr>
<td>E</td>
<td>$r_3, r_4$</td>
</tr>
<tr>
<td>F</td>
<td>$r_3$</td>
</tr>
<tr>
<td>G</td>
<td>$r_2$</td>
</tr>
</tbody>
</table>

Permission Assignment

<table>
<thead>
<tr>
<th>Role</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td></td>
</tr>
<tr>
<td>$r_2$</td>
<td>$p_1, p_3$</td>
</tr>
<tr>
<td>$r_3$</td>
<td>$p_2$</td>
</tr>
<tr>
<td>$r_4$</td>
<td>$p_5, p_7$</td>
</tr>
<tr>
<td>$r_5$</td>
<td>$p_6$</td>
</tr>
<tr>
<td>$r_6$</td>
<td>$p_4, p_8$</td>
</tr>
</tbody>
</table>

Access Matrix

\[
\begin{array}{ccccccccc}
 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 \\
A  & \times & \times & \times & \times & \times & \times & \times & \times \\
B  & \times & \times & \times & \times & \times & \times & \times & \times \\
C  & \times & \times & \times & \times & \times & \times & \times & \times \\
D  & \times & \times & \times & \times & \times & \times & \times & \times \\
E  & \times & \times & \times & \times & \times & \times & \times & \times \\
F  & \times & \times & \times & \times & \times & \times & \times & \times \\
G  & \times & \times & \times & \times & \times & \times & \times & \times \\
\end{array}
\]
Outline

Limitations of RBAC

Attribute-Based Access Control
Limitations of RBAC

Attribute-Based Access Control
RBAC was designed for simpler systems
Limitations of RBAC

- Role design and engineering is difficult and expensive
  - Substantial research on role engineering top down and bottom up (1996-) and on role mining (2003-)
- Changes in business can lead to new role hierarchies
- RBAC requires attention all the time
  - Keeping user-role assignment updated is critical
  - Users can easily accumulate roles, leading to excess permissions
- RBAC does not take into account other parameters
  - Object metadata
  - Environmental context
- RBAC is not dynamic nor context-aware
  - RBAC cannot restrict access w.r.t. time, location, etc.
More importantly...

RBAC does not *really* scale!!
Role explosion

- Often large number of roles is required (role explosion)
  - Role granularity is not adequate
  - RBAC is unable to deal with relations

- Examples
  - Nurses can view medical records in their unit
  - Doctors can edit medical records of patients they are assigned to
  - Surgeries should be approved by two doctors

- Which roles should be defined?
  - Define a role *nurse* per each *unit* (e.g., nurse_unit)?
  - Define a role *primary doctor* per each *patient* (e.g., primary_doctor_of_patient_X)?
  - Does this scale?

- Many organizations claim a 10-to-1 role-to-employee ratio
Limitations of RBAC
Limitations of RBAC
Limitations of RBAC
Limitations of RBAC
Limitations of RBAC

Exercise

Define a RBAC$_3$ system to regulate permissions within a bank branch. The system should implement the following requirements:

1. A bank employee can be a clerk, a manager or the head of the bank branch.
2. A bank branch can have only one head.
3. Clerks can make loan offers to customers.
4. Loan offers should be reviewed by a different clerk or a manager before they can be approved.
5. If the amount of the loan offers is lower than $15K, the offer should be approved by a manager.
6. If the amount of the loan offers is equal or greater than $15K, the offer must be approved by two managers.
7. A bank employee cannot approve loan offers he made or reviewed.
Solution

- Roles, Objects, Operations
- Permission Assignment
- Role Hierarchy
- Constraints
  - Separation of Duty (Mutually Exclusive Roles)
  - Cardinality
  - Prerequisite roles
Solution: Roles, Objects, Operations

Roles = {employee, clerk, clerk-make, clerk-review, manager, manager-review, manager-approve1, manager-approve2, head}

Objects = {loan offer_{<15K}, loan offer_{≥15K}}

Operations = {make, review, approve, first_approve, second_approve}

Session: execution of an instance of the loan process.
Solution: Permission assignment & Role Hierarchy

Permission Assignment

<table>
<thead>
<tr>
<th>Role</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>clerk-make</td>
<td>(make, loan offer $&lt; 15K$) (make, loan offer $\geq 15K$)</td>
</tr>
<tr>
<td>clerk-review</td>
<td>(review, loan offer $&lt; 15K$) (review, loan offer $\geq 15K$)</td>
</tr>
<tr>
<td>manager-review</td>
<td>(review, loan offer $&lt; 15K$) (review, loan offer $\geq 15K$)</td>
</tr>
<tr>
<td>manager-approve</td>
<td>(approve, loan offer $&lt; 15K$)</td>
</tr>
<tr>
<td>manager-approve1</td>
<td>(first_approve, loan offer $\geq 15K$)</td>
</tr>
<tr>
<td>manager-approve2</td>
<td>(second_approve, loan offer $\geq 15K$)</td>
</tr>
</tbody>
</table>

Role Hierarchy

```
employee
  └── clerk
    └── clerk-make
    └── clerk-review
  └── manager
    └── manager-approve
      └── manager-approve1
      └── manager-approve2
  └── head
```
Solution: Constraints

- Mutually Exclusive Roles
  - Enforce separation of duty
  - $smer(rs, n)$: any user cannot be assigned to $n$ or more roles in role set $rs$
  - $dmer(rs, n)$: any user cannot simultaneously activate $n$ or more roles from role set $rs$ in one session

- Cardinality constraints
Solution: Constraints

Mutually Exclusive Roles
\[
dmer(\{\text{clerk-make, clerk-review}\}, 2)
\]
\[
dmer(\{\text{manager-approve1, manager-approve2}\}, 2)
\]
\[
dmer(\{\text{manager-review, manager-approve}\}, 2)
\]
\[
dmer(\{\text{clerk-review, manager-approve}\}, 2)
\]
\[
dmer(\{\text{clerk-make, manager-approve}\}, 2)
\]
\[
dmer(\{\text{manager-review, manager-approve1}\}, 2)
\]
\[
dmer(\{\text{clerk-review, manager-approve1}\}, 2)
\]
\[
dmer(\{\text{clerk-make, manager-approve1}\}, 2)
\]
\[
dmer(\{\text{manager-review, manager-approve2}\}, 2)
\]
\[
dmer(\{\text{clerk-review, manager-approve2}\}, 2)
\]
\[
dmer(\{\text{clerk-make, manager-approve2}\}, 2)
\]

Cardinality constraints
\[
\{u|(u, head) \in UA\} = 1
\]
Outline

Limitations of RBAC

Attribute-Based Access Control
Attribute-Based Access Control

The ABAC Shift

From **RBAC**
- Many users in one role

To **ABAC**
- Many attributes for users/resources/...
A logical access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes.

NIST Special Publication 800-162
By 2020, 70 percent of enterprises will use ABAC as the dominant mechanism to protect critical assets, up from less than 5 percent today.

Gartner, 2013
Attribute-Based Access Control

**Basic definitions**

- **Attributes** are characteristics that define specific aspects of the subject, object, environment conditions, and/or requested actions that are predefined and pre-assigned by an authority.

- A **subject** is an active entity (generally an individual, process, or device) that causes information to flow among objects or changes the system state.

- An **object** is a passive information system-related entity containing or receiving information. It can be a resource upon which an operation can be performed by a subject including data, applications, services, devices, and networks.

- **Environmental conditions** are dynamic factors, independent of subject and object, that may be used as attributes at decision time to influence an access decision. Examples of environment attributes include time, location, temperature, etc.

- An **operation** is the execution of a function at the request of a subject upon an object. Operations include read, write, delete, copy, execute, and modify.

- **Policy** is the formal representation of rules that define the set of operations a subject can perform on an object in permitted environment conditions.
In concrete...

- Attributes are name-value pairs
  - possibly chained
  - values can be complex data structures
- Associated with
  - subjects
  - objects
  - actions
  - environment
- Converted by policies into access rights at evaluation time
  - policies specified by security officer
  - attributes maintained by system administrator
  - but also possibly provided by external sources
Policies

- Policies directly reflect business requirements
- Both positive and negative authorization can be specified
- Policies comprises rules of the form “IF ... THEN ...”
Example Policies

1. Doctors can view medical records of patients assigned to them

   IF  subject_role = doctor AND object_type = medical_record AND
       action = view AND medical_record/patient_primary_doctor = userID
   THEN  Permit

2. Employees of the business unit cannot enter areas of the research unit outside office hours.

   IF  subject_role = employee AND subject_unit = business AND
       area_unit = research AND action = enter AND
       (time < 8:00 OR time > 17:00)
   THEN  Deny
System model

Access Control Policy

Environment Conditions

Subject Attributes

Object Attributes

Subject

Object

ABAC Access Control Mechanism

Decision

Enforce

1

2a

2b

2c

2d

3
## ABAC vs. RBAC

<table>
<thead>
<tr>
<th></th>
<th>RBAC</th>
<th>ABAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granularity</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Context-awareness</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Set up</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Administration</td>
<td>Easy</td>
<td>Hard</td>
</tr>
</tbody>
</table>
Policy and Attribute Administration

1. Who defines the policies?
2. Do policies reflect the given access requirements?
3. From where attributes are retrieved?
4. ...
Delegation

- Control is often decentralized
- Data owner may delegate the authority of policy definition to other entities

A model for delegation will be presented in the lectures on RT (next two lectures).
Policy Composition

- Each user can define more than one access rule
- Multiple users can define access rules
Policy Analysis

- Does a policy meet its requirements?
Attribute Retrieval

- Attributes are often retrieved from external sources
- What if those sources are not available?
- ...and if they provide incorrect attribute values?
Conclusion

- RBAC suffers several limitations
  - Expressiveness is quite limited
  - Context-awareness is not supported
  - Role engineering is difficult and time-consuming
- Also, RBAC is not really scalable
  - Role explosion
- ABAC is becoming the most predominant access control paradigm
- Still several open challenges!!
- In later lectures... eXtensible Access Control Markup Language
References