Attribute-Based Access Control

Shamik Sural
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
shamik@cse.iitkgp.ac.in

Agenda

- Access Control Preliminaries
- Traditional Access Control Models
- Attribute Based Access Control (ABAC)
- Enforcing ABAC Policies
- Baseline Approach

Access Control Preliminaries

What is Access Control?

- Let’s look at some facts in our daily life
  - *During my school exams, my parents did not allow me to watch TV serials*
  - *At IIT Kharagpur, during end-semester examinations, Central Library is kept open at night for students*
- Idea of controlling access to resources is realistic and natural
- Fundamental questions are
  - How to represent required access control
  - How to enforce access control

Access Control – Needs Vary

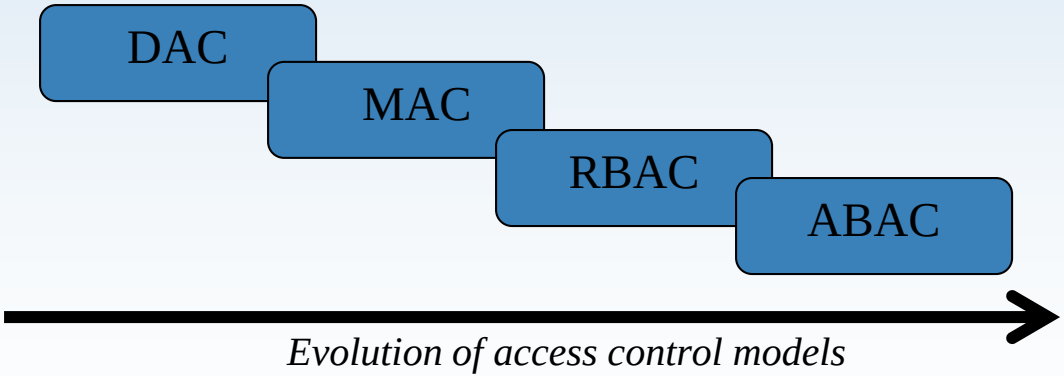
- Access control needs to be enforced at various levels in a computer system
  - An OS has to manage resources, e.g., read, write, execute access to files and directories
  - A DBMS can do the same for database objects, e.g., the users and privileges granted in ORACLE
  - Application level security manager does the same for an application
    - Internet banking by bank customers
    - Getting patient's medical record in healthcare applications
    - A remote method being called by a JAVA object

Access Control Policies

- There ought to be some uniformity among them. Isn’t it?? Is it there??
- Different kinds of access control requirement in different situations
- Access control in a system dictated by the system ***Security Policy***

Access Control Models

- Abstract representation of security policies



Discretionary Access Control (DAC) Model

- Decentralized
- At owner's discretion
  - *Prof. X can choose to allow a certain set of people to access her lab facility*
- Typically enforced through access control lists
- Based on the identity of individuals

Discretionary Access Control

Objects Subjects	O1	O2	S1	S2
S1	Read	Read, Write, Own	NULL	NULL
S2	Read, Own	NULL	NULL	NULL

Characteristics	PROS	CONS
➢ Decentralized ➢ Owner discretion ➢ Enforcement through ACL ➢ Identity based ➢ Permission attached to objects	➢ Easy to implement ➢ Flexibility ➢ Built-in in most OS, DBMS	➢ Possibility of ACL explosion ➢ Prone to mistakes

Mandatory Access Control (MAC) Model

- Centralized
- Access distribution is enforced by the system
- Subjects are assigned clearance levels such as *top secret*, *secret*, *confidential*, *etc.*
- Objects are assigned similar classification levels

Mandatory Access Control

CLEARANCE LEVEL	CLASSIFICATION
Level 5	Top Secret, Secret, Classified, Unclassified
Level 4	Secret, Classified, Unclassified
Level 3	Classified, Unclassified
Level 2	Unclassified

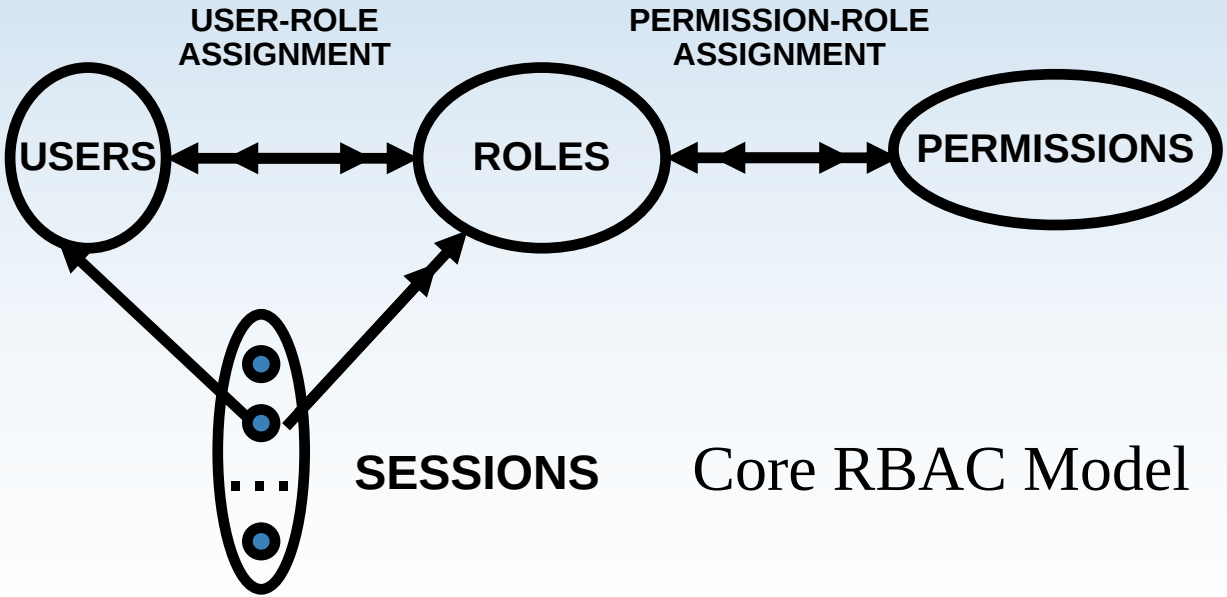
Characteristics
➢ Centralized ➢ Enforced through clearance and classification ➢ Read allowed for subjects with same or higher clearance than the classification level of object

PROS	CONS
➢ Easy to Scale ➢Secure	➢ Not Flexible ➢ Limited user functionality ➢ High admin overhead

Role-Based Access Control (RBAC) Model

- The concept of *Role* is very natural in any organization
  - Professor, Chairman, Dean, President, etc.
- Popularity of RBAC mainly due to its success in commercial applications
- Most of the databases support RBAC through SQL
- RBAC has less administrative overhead

Role-Based Access Control



Core RBAC Model

Why Yet Another Access Control Model

- Inability to handle context like date and time of access, server load, etc.
- Fine grained access control
- Ad hoc access to new users
- Bringing most of the existing models to a common model

Attribute-Based Access Control (ABAC)

- Based on the notion of attributes
- Attributes are characteristics of user, object and environment
- Each entity is associated with a set of well-defined attributes
- Each attribute can assume one or more possible values
    - For example, a user $\boldsymbol{u_1}$ can have the value ***professor*** for the user attribute ***designation*** and the value ***CS*** for the user attribute ***department***

Components of ABAC

- **User (U)**
  - Entities that request for access to resource
  - u_1, u_2, u_3, u_4
- **Object (O)**
  - Resources to be protected from unauthorized access
  - o_1, o_2, o_3, o_4
- **Environmental Condition (E)**
  - Context in which access requests are made
  - e_1, e_2
- **Operation (OP)**
  - Activities performed by users on objects
  - *read*, *write*, *execute*, *print*

Components of ABAC

- **User Attribute (UA)**
  - Set of possible attributes associated with a user like *designation*, *department*, etc.
  - a ∈ UA is associated with a set V_a^u of possible values
  - $V_{designation}$: {*Professor*, *Student*}, $V_{department}$: {*CS*, *ECE*}
- **Object Attribute (OA)**
  - Set of possible attributes such as *type*, *confidentiality*, etc., associated with an object that can affect access decisions
  - Each a ∈ OA is associated with a set V_a^o of possible values
  - V_{type} : {*Assignment*, *Question paper*}, $V_{confidentiality}$: {*Low*, *High*}

Components of ABAC

- **Environmental Attribute (EA)**
  - Set of possible attributes such as *day of request*, *source subnet*, etc., associated with an environmental condition that can affect access decisions
  - Each member is associated with a range set V_a^e of possible values it can acquire
  - V_{day} : {*Weekday*, *Weekend*}

Components of ABAC

- **Policy (P)**
  - Consists of a set of authorization rules
  - Each rule $r_i \in P$ is of the form $c_i^u \wedge c_i^o \wedge c_i^e \wedge op_i$
  - c_i^u , c_i^o and c_i^e represent user condition, object condition and environmental condition of the form

    $(a_1^u = v_1 \wedge a_2^u = v_2, \ldots, a_m^u = v_m)$

    $(a_1^o = v_1 \wedge a_2^o = v_2, \ldots, a_n^u = v_n)$

    $(a_1^e = v_1 \wedge a_2^e = v_2, \ldots, a_k^e = v_k)$

Components of ABAC

- **Example ABAC policy**
  - r_1 : {Designation = Professor} ^ {Type = Assignment} ^ {Day = Weekday} ^ {op = Modify}
  - r_2 : {Designation = Student} ^ {Type = Assignment} ^ {Day = Weekend} ^ {op = Read}
  - r_3 : {Designation = Professor} ^ {Type = Question paper} ^ {Day = Weekday} ^ {op = Modify}
  - r_4 : {Designation = Student} ^ {Type = Assignment} ^ {Day = Weekday} ^ {op = Submit}

Components of ABAC

- **Rules with “*”**
  - Scenarios where an attribute in a rule can assume all possible values
    - For example, A professor can modify assignments on **any** day
    - **any** is represented as **“*”** in ABAC

- **Representation of rule**
  - {Designation = **Professor**} ^ {Type = **Assignment**} ^ {Day = *} ^ {op = **Modify**}

Components of ABAC

User	Designation
u_1	Student
u_2	Professor
u_3	Student
u_4	Professor

UV = User attribute-value pair assignment

Object	Type
o_1	Assignment
o_2	Question paper
o_3	Question paper
o_4	Assignment

OV = Object attribute-value pair assignment

Environmental Condition	Day
e_1	Weekday
e_2	Weekend

EV = Environmental attribute-value pair assignment

Components of ABAC

- **Example ABAC policy**

Rule	Designation	Type	Day	Operation
r_1	Professor	Assignment	Weekday	Modify
r_2	Student	Assignment	Weekend	Read
r_3	Professor	Question paper	Weekday	Modify
r_4	Student	Assignment	Weekend	Submit

Enforcing ABAC Policy

- An access request is a request made by a user to access an object at a certain environmental condition
- Requesting user has certain values for various user attributes
- Requested object has certain values for various object attributes
- Policy enforcement process consults the ABAC policy to determine whether the access should be granted or denied
- Decision depends on the user and object attribute values as well as the rules in the current policy
- Access decision not fixed due to environmental attributes

Motivation

- Time required to resolve an access request depends on
  - Number of rules in the policy
  - Number of attribute-value pairs in the rules
- Way to reduce the number of comparisons
  - Get a rule early that provides the access
  - Evaluate a rule until an attribute-value pair mismatches with the access request
  - For example, a user who is a ***professor*** cannot use a rule where ***designation*** has the value ***student***.

Baseline Approaches

- An access request is represented as
  - $\langle u, o, e, op \rangle$ where $u \in U, o \in O, e \in E$ *and* $op \in OP$
- Two baseline approaches
  - **Sequential Searching of the Rules in a Policy**
    - Sequentially traverse rules to **search for a rule that permits *u* to perform *op* on *o***.
  - **Rule Re-ordering for Improved Sequential Search**
    - Improve sequential search by **rearranging the rules** followed by **re-shuffling of attribute-value pairs within each rule**

References

- V. C. Hu, D. Ferraiolo, D. R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. Guide to Attribute-Based Access Control (ABAC) definition and considerations. Technical report, NIST Special Publication 800-162, 2014.
- A. X. Liu. Firewall policy verification and troubleshooting. Computer Networks: The International Journal of Computer and Telecommunications Networking, pages 2800-2809, 2009.
- S. Khummanee and K. Tientanopajai. The policy mapping algorithm for high-speed firewall policy verifying. International Journal of Network Security, pages 433-444, 2016.
- M. Narouei, H. Khanpour, H. Takabi, N. Parde, and R. Nielsen. Towards a top-down policy engineering framework for attribute-based access control. In ACM Symposium on Access Control Models and Technologies, pages 103-114, 2017.
- E. S. Al-Shaer and H. H. Hamed. Firewall policy advisor for anomaly discovery and rule editing. In International Symposium on Integrated Network Management, pages 17-30, 2003.