Towards an ABAC Family of Models

David Ferraiolo
National Institute of Standards and Technology
dferraiolo@nist.gov

Why a Family of Models

- ABAC Approaches share common features, but differ in other legitimate aspects.
  - Need a common framework and terminology
- Due to a lack of consensus on ABAC features, users cannot accurately assess the benefits and challenges associated with different ABAC features.
- ABAC is also a rich and open-ended technology
  - Treating ABAC as a single model would be a mistake.

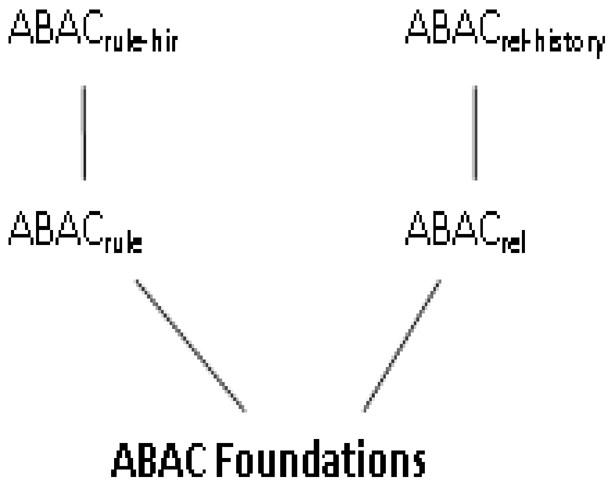
Scope

- Do not want to stifle innovation, but need to limit scope
- Canonical features of an ABAC model:
  - Should be well understood and well represented within ABAC literature and/or standards base **AND** Should be known to be viable in that there exists at least one example commercial or reference implementation
  - Also included are features that can be expressed as a natural extension to these existing features.

Guidance

- Two Standards that apply ABAC
  - eXtensible Access Control Markup Language (XACML)
  - ANSI/INCITS Next Generation Access Control (NGAC) based on the Policy Machine (PM) reference implementation
- XACML is widely accepted and implemented
- NGAC three sub-projects: (1) Implementation Requirements, Protocols and API Definitions; (2) Functional Architecture; (3) Generic Operations & Abstract Data Structures.
- (2) is now available as ANSI INCITS 499-2013
- The Policy Machine evolved from a concept, to a reference implementations, and will soon be promoted as an open source project.

A Family of ABAC Models



ABAC Foundation

- Elements
  - authorized users (U)
  - user attributes (UA),
  - operations (Op),
  - objects (O),
  - object attributes (OA), and
  - environmental attributes (EA).
- Relations
  - user to user attribute assignments (UUA), and
  - object to object attribute assignments (OOA).
- Reference Mediation: Determines which users can perform which operations on which objects based on attributes assigned to the user, attributes assigned to the object, environmental conditions and either a set of rules and/or a set of relations

Generic ABAC Administrative Advantages

- User provisioning is achieved by simply assigning the user to appropriate user attributes.
- User re-provisioning is achieved by deleting old attribute assignments and creating new assignments.
- User de-provisioning is achieved by deleting attribute assignments.
- Object provisioning is automated by simply assigning the object to appropriate object attributes.
- Object re-provisioning is achieved deleting old attribute assignments and creating new assignments.
- Object de-provisioning is achieved by deleting attribute assignments.

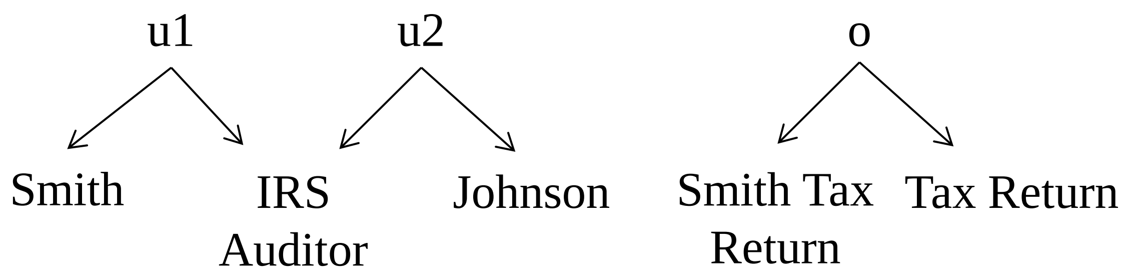
ABAC$_{\text{rule}}$

- Includes basic elements and relations of ABAC foundations, plus:
- A() – A function returning the set of attributes of a user, object, or the environment
- A set of policy rules, and
- Reference mediation: a Boolean function that determines whether a user *u* can perform an operation *op* on an object *o* in a particular environment *e*, under the set of rules:

$$\{\text{grant}, \text{deny}\} \leftarrow \text{decision}(A(u) \times A(o) \times A(e) \times \text{Rules} \times \text{op})$$

- If access can be granted, (u, op, o) is a privilege, and (op, o) is a capability for u, and (u, op) is an access control entry for o.

Example



Current_Time(e) = 09:30

Policy1
Rules:
IRS Auditors can read and write Tax Returns
Tax Returns can be accessed between 08:00 and 18:00
Deny (u1, write, Smith Tax Return)

Under policy1 user u1 (Smith) can read object o (Smith Tax Return), and user u2 can read and write object o.

Resulting privileges:
(u1, r, o), (u2, r, o), (u2, w, o)

$\text{ABAC}_{\text{rule}}$ Advantages

In addition to the generic administrative advantages, $\text{ABAC}_{\text{rule}}$ adds the following:

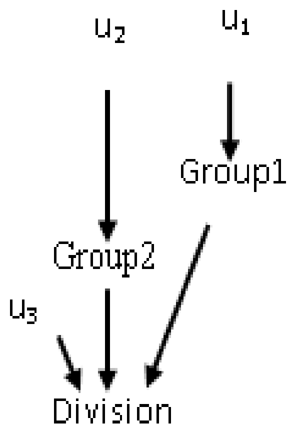
•Policy expression is flexible. A wide variety of mandatory access control policies can be expressed.

•Rules can be set up quickly.

$\text{ABAC}_{\text{rule-hier}}$

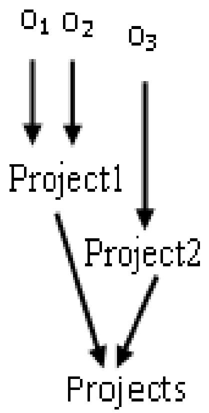
- In addition to the relations that are specified in $\text{ABAC}_{\text{rule}}$, the $\text{ABAC}_{\text{rule-hier}}$ model adds the notion of user and object attribute hierarchies.
- Includes two additional types of assignment relations; user attribute to user attribute assignments (UAUA) and object attribute to object attribute assignments (OAOA).
- The additional assignments relations results in inherited attributes for users and objects.

Example



Rules:

(1) users in Division can read objects in Projects
(2) users in Group1 can write to objects in Project1
(3) users in Group2 can write to objects in Project2



Resulting Privileges:

(u1, r, o1), (u1, w, o1),
(u1, r, o2), (u1, w, o2),
(u1, r, o3), (u2, r, o1),
(u2, r, o2), (u2, r, o3),
(u2, w, o3), (u3, r, o1),
(u3, r, o2), (u3, r, o3)

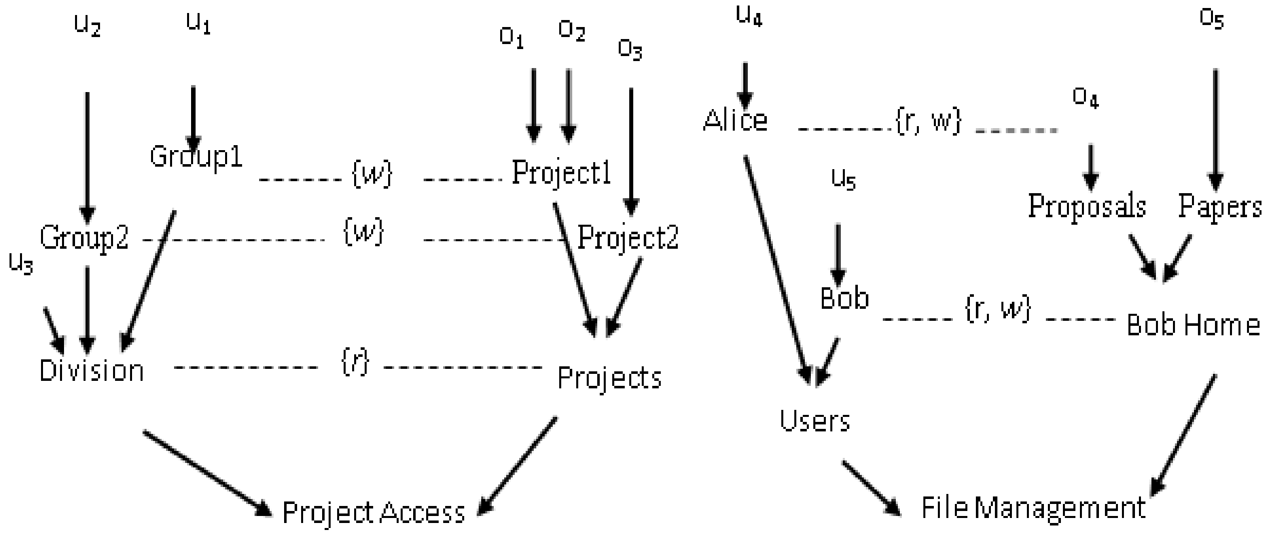
$\text{ABAC}_{\text{rule-hier}}$ Advantages

- In addition to the administrative advantages that belong to the $\text{ABAC}_{\text{rule}}$ model, $\text{ABAC}_{\text{rule-hier}}$ adds the following:
- Fewer rules. Rules and associated capabilities (op, o) and access control entries (u, op) are respectively inherited up the user attribute and object attribute hierarchies.
- Fewer UUA and OOA relations.
- Added visibility. It is generally more intuitive to create an assignment than to specify potentially multiple additional rules.

ABAC_{rel}

- In addition to relations defined by ABAC Foundation, ABAC_{rel} includes UAUA, OAOA, and attribute association relations (AA), Policy Classes (PC), object attribute to policy class assignments (OAPC), and user deny relations (U-DENY).
- A standard set of administrative operations
- Reference Mediation: A user access request <op, o> from *u* is granted iff there exists a privilege (*u*, *op*, *o*) and capability (*op*, *o*) has not been denied for *u*.
- Note: U-DENY are exceptions to privileges

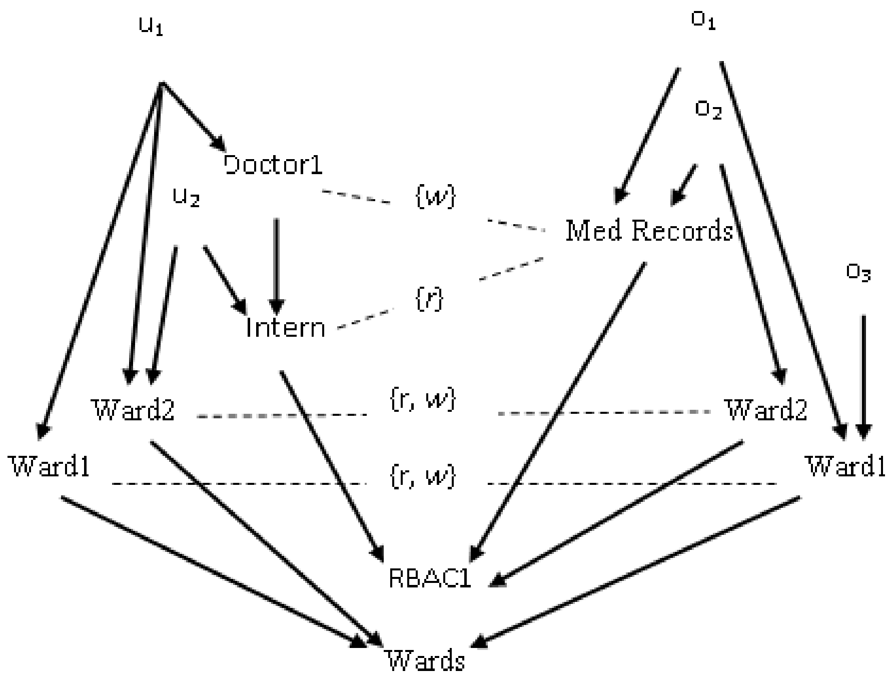
Example ABAC$_{rel}$ relations shown as graphs



(*u*1, *r*, *o*1), (*u*1, *w*, *o*1), (*u*1, *r*, *o*2), (u1, *w*, *o*2), (*u*1, *r*, *o*3), (*u*2, *r*, *o*1), (*u*2, *r*, *o*2), (*u*2, *r*, *o*3), (*u*2, *w*, *o*3), (*u*3, *r*, *o*1), (*u*3, *r*, *o*2), (*u*3, *r*, *o*3)	**(*u*4, *r*, *o*4), (*u*4, *w*, *o*4), (*u*5, *r*, *o*4), (*u*5, *w*, *o*4), (*u*5, *r*, *o*5), (*u*5, *w*, *o*5)**

Derived Privileges

Policy Combinations



Implied Rules:
Doctors in Ward1 can read and write Medical Records in Ward1.
Doctors in Ward2 can read and write Medical Records in Ward2.
Doctors or Interns in Ward1 can read Medical Records in Ward1.
Doctors or Interns in Ward2 can read Medical Records in Ward2.

(u, op, o) is a privilege iff for each policy class, *pc* for which *o* is contained (through a chain of assignments) there exists an association (*ua*, *ops*, *oa*), such that *u* is in *ua*, *op* is in *ops*, and *o* is in *oa* and *oa* is in *pc*.

(*u*1, *r*, *o*1), (*u*1, *w*, *o*1), (*u*1, *r*, *o*2), (*u*1, *w*, *o*2), (*u*1, *r*, *o*3), (*u*1, *w*, *o*3), (*u*2, *r*, *o*2), (*u*2, *w*, *o*2)

Admin Privileges

- Administrative privileges are also specified through AA.
- The operation set is a set of parameterized administrative operations, objects are data elements and relations and attributes (containers) serve as the actual parameters of the operation.
- E.g., administration of user assignments in Group2 could be specified using the following association: Group2-Admin, {create uua(), delete uua()}, Group2
- For convenience parameterized admin commands enable the execution of a sequence of admin operations (E.g., Create DAC User (user, user name, user home))

ABAC_{rel} Advantages

- Privilege review. Can be used:
  - To validate or invalidate compliance to policy
  - By an ordinary users to discover his/her capabilities to access objects.
  - To determine who has access to a specific object
- Flexible mandatory access control policy specification. Accommodate same OR, AND and Deny operators of $\text{ABAC}_{\text{rule}}$.
- Decentralized administration.
- Discretionary Access Controls.
- Policy Combinations (e.g., DAC and RBAC)

ABAC$_{\text{rel-history}}$

- In addition to the elements and relations of ABAC$_{\text{rel}}$, ABAC$_{\text{rel-history}}$ adds a process as an element, and a process-based deny relation, and event pattern/response relation.
- We denote by *process user*(*p*) the user associated with process *p*.
- We denote by <*op*, *o*>p a process access request, where *op* is an operation and *o* is an object.
- Reference Mediation: A process access request <*op*, *o*>p is granted iff there exists a privilege (*u*, *op*, *o*), where *u* = *process user*(*p*), capability (*op*, *o*) has not been denied for either *u* or *p*

event pattern/response relations

- An event pattern/response relation is a pair (*ep*, *r*) (usually denoted **when** *ep*, **do** *r*), where *ep* is an *event pattern* and *r* is a sequence of administrative operations, called a *response*.
- The event pattern specifies conditions that if matched by the context surrounding a “process’ successful execution of an operation on an object” (an event), the response is immediately executed, thereby changing the state of the policy.

process deny relations

- Similar to a user deny relation, a process deny relation is a triple of the form *p_deny*(*p*, *ops*, *os*), where *p* is a process, *ops* is an operation set, and *os* is an object set.
- Its meaning is that the process *p* cannot perform operations in *ops* on the objects in *os*.
- E.g., **when:** (p, r, o in Top_Secret), **do:** create p_deny(p, w, ¬Top_Secret)

Other History-based policies

Achieved through formulation of event pattern/response relations, e.g.,:

•Forms of History based Separation of Duty

•Conflict of Interest

•Data Tracking

•Facilitate workflow

For more details see: D. Ferraiolo, V. Atluri, and S. Gavrila, “The Policy Machine: “A Novel Architecture and Framework for Access Control Policy Specification and Enforcement,” J. Systems Architecture, vol. 57, no. 4, 2011, pp. 412–424.

ABAC$_{\text{rel-history}}$ Advantages

- Expression and enforcement of history-based access control policies.
- Dynamically alter access state in response to ____.

Questions?