Access Control

Notes-Set5 Attribute Based Access Control(ABAC)

Traditional Access Control Models- Limitations

- In DAC, information may be accessed by unauthorized users because **there is no control on copies of objects.**
- MAC deals with information flow and solves this problem by attaching security levels on both users and objects. **All users are required to obtain certain clearance to access objects**. **Security labels propagate to derivative objects, including copies**. However, the policies in DAC and MAC are fixed and there is no room for flexible access control.
- RBAC has become the dominant form of access control in practice. Recently there has been growing practitioner concern with the limitations of RBAC.
  - For example, **role explosion** is caused by the situation where **each role requires different sets of permissions and large number of roles have to be defined**.
  - **Role engineering also delays the deployment of RBAC** as it is the most costly process before deployment.
  - Due to the **static nature**, RBAC is **unable** to model **policies** that depend on **contextual** details.
  - Various roles assigned to a given user could contain conflicting data
  - Those limitations have been met by researchers in two different way by extending RBAC in numerous directions.
    - Role activation process has been extended to be constrained by contextual information such as time and location, prerequisite roles and so on.
    - Users are associated with additional information besides role, such as organization, group and team.
    - Permission is also extended to include purpose and conditions to support privacy aware RBAC.

Traditional A C Models – Pros & Cons

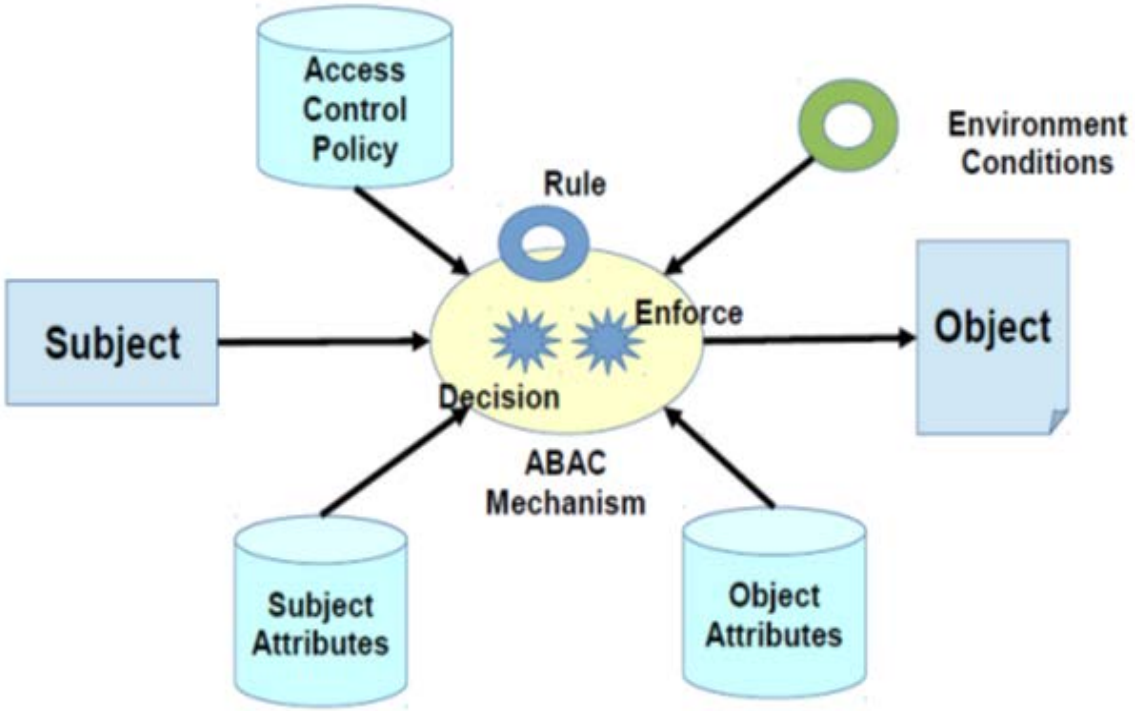
	Pros	Cons
DAC	• Easy to implement • Highly flexible	• Doesn't scale well • ACL explosion possibility • Prone to mistakes
MAC	• Most secure • Easy to scale	• Not flexible • Limited user Functionality • High admin overhead
RBAC	• Scalable • Flexible – user & permission are loosely coupled • Less administration required	• Roles needs provisioning and maintenance • Possibility of role explosion • Unable to accommodate real-time context

Today's Need

- Today, defining authorization policies based solely on a user's role is not good enough.
- The context surrounding that user, their data, and the interaction between the two are also important to provide access to the right user, at the right time, in the right location, and by meeting regulatory compliance.
- Knowing a user's role is no longer enough to ensure safe and secure access control.
- Authorization must answers:
  - **What is a user's role(s)?**
  - **Who or what is that user related to?**
  - **What resources should he/she have access to?**
  - **What can they do with this data?**
  - **Where is data being accessed from?**
  - **At what time?**
- These relationships are key to defining safe and secure access control in an ever-changing environment.
- The context and the relationship information between various entities involved in the system are required.
- Attribute-based systems provide fine-granularity, high flexibility, rich semantics and other beneficial features like partial authentication and natural support for role-based access control. These features are more advantageous as organizations move to greater collaboration and data sharing across and outside the enterprise

Attribute-Based Access Control (ABAC)

NIST defines ABAC as "*An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions*"

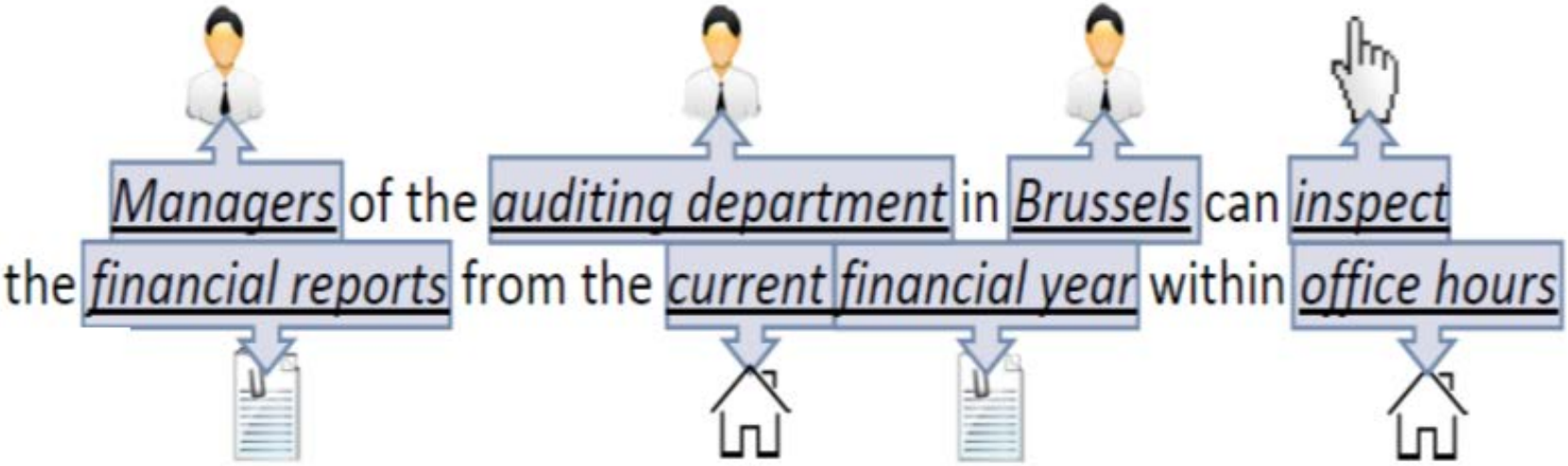


Attribute-Based Access Control (ABAC)

- Attribute-based access control takes a **very different approach to authorization**.
- Also known as **Policy-based access control or or claims based access control**, where access rights are granted to users through the use of policies which combine attributes together.
- The policies can use any type of attributes (user attributes, resource attributes, object, environment attributes etc.).
- Attributes are Used to Express Rich Policies
- This model supports Boolean logic, in which rules contain "IF, THEN" statements about who is making the request, the resource, and the action. It allows unprecedented control of **access to restricted resources based** on **fine-grained attributes evaluated at run-time.**
  - For example: IF the requestor is a manager, THEN allow read/write access to sensitive data.
- **ABAC can combine multiple attributes to make a context-aware decision regarding individual requests for access.**
- The **rules based on these attributes can also incorporate contextual privacy requirements** such as:
  - Restrictions on outsourcing personal data.
  - Data minimization where only personal data that is required for the provision of the requested product or service should be collected by an organization. It is not permissible to collect data that “might be” useful at some point in the future.
  - Purpose specification that only allows an organization to use personal data for the express purpose for which it was collected, and where any secondary use requires the consent of the individual.

ABAC

1. fine-grained access control
2. context-aware access control
3. dynamic access control



ABAC

- Attribute-based access control (ABAC) could encompass the demonstrated benefits of DAC, MAC and RBAC while transcending their limitations.
- An attribute is a property expressed as a **name: value pair** associated with any entity in the system, including users, subjects and objects and even attributes themselves.
- Appropriate attributes can capture identities and access control lists (DAC), security labels, clearances and classifications (MAC) and roles (RBAC). In fact, technically **ABAC is capable of enforcing DAC, MAC, and RBAC**.
- Languages for specifying permitted accesses based on the values and relationships among these attributes provide policy flexibility and customization.
- ABAC supplements and subsumes rather than supplants these currently dominant models.
- Moreover any number of additional attributes such as location, time of day, strength of authentication, departmental affiliation, qualification, and frequent flyer status, can be brought into consideration within the same extensible framework of attributes.
- ABAC is considered a **"next generation" authorization model** because it provides **dynamic, context-aware and risk-intelligent access control** to resources allowing access control policies that include specific attributes from many different information systems to be defined to resolve an authorization and achieve an efficient regulatory compliance, allowing enterprises flexibility in their implementations based on their existing infrastructures.
- **Gartner predicts that ABAC will be used by 70% of enterprises by 2020, replacing RBAC as the dominant mechanism to protect critical assets**

ABAC

- Attributes are name:value pairs
  - possibly chained
  - values can be complex data structures
- Associated with
  - actions
  - users
  - subjects
  - objects
  - contexts
  - policies
- Converted by policies into rights just in time
  - policies specified by security architects
  - attributes maintained by security administrators
  - but also possibly by users OR reputation and trust mechanisms
- Inherently extensible

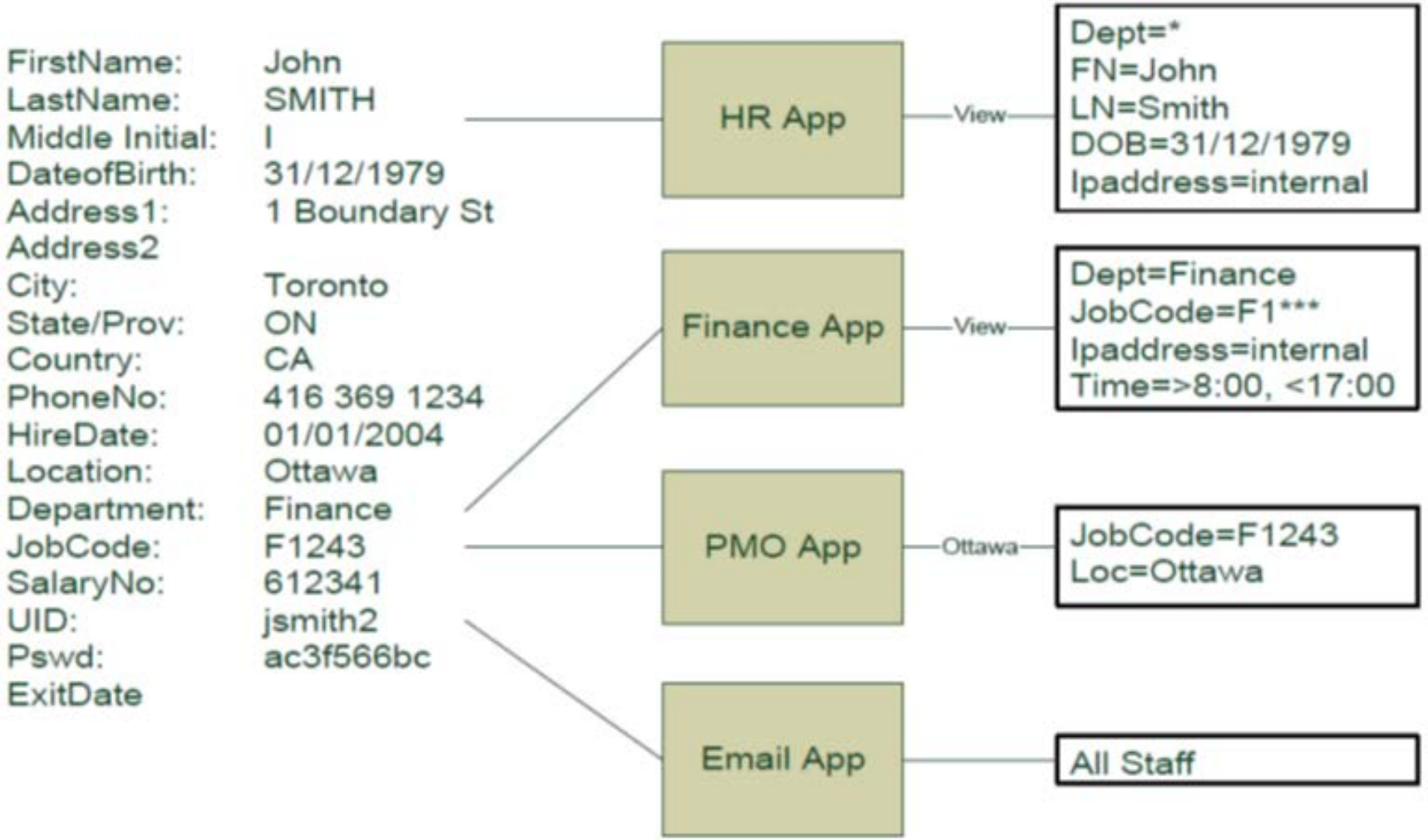
ABAC- Example

- Attribute evaluation then enables effective **policy-based authorization**.
- Let's consider this example:
  - A **policy** states that "all users belonging to the sales department should have read access to proposals sent to customers within the sales region to which the user belongs".
  - An **access request evaluation** based on the following attributes and attribute values should therefore return PERMIT:
    Subject's "department"="sales"
    Subject's "sales region"="NW"
    Action="read"
    Resource "type"="proposal document"
    Resource "region"="NW"

ABAC - Example

- In attribute-based access control (ABAC), one or more profile attributes can be used to make a context-aware access decision.
  - For example, for a patient whose age is below 18, his parent may access certain patient medical files without prior permission. When the patient passes his 18th birthday, neither his role nor his parent's role changes, but his age attribute does change, resulting in reduced access for his parents.
  - **Use case examples:**
    - Show customer account information only to supervisors who have completed 15 hours of training and who make the request during business hours.
    - Permit system access only to users who are logging in from specific geographic locations.
    - If a supervisor tries to access a customer's account information five times during a single day, deny access and email an alert to the security manager.
    - An engineer that is reassigned to a different project can automatically access information related to the new project but not their previous one
    - An account executive that is reassigned to a new territory is automatically able to see accounts and products in their new territory but can no longer access anything from their old territory
    - A finance manager can only download docs when he or she is physically in the US.

ABAC Example



ABAC - Example

Online Movie Viewing Application

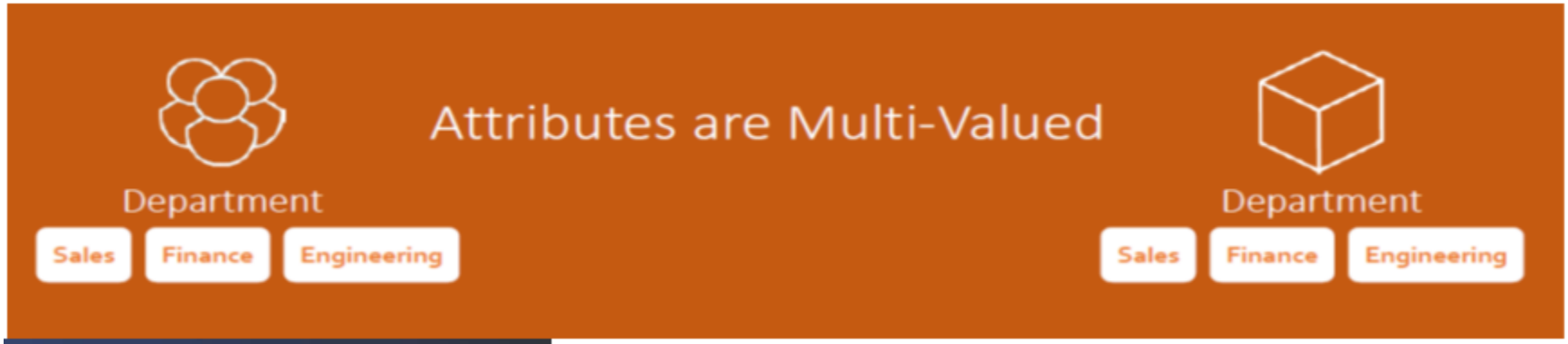
- **Basic Policy: Access to a movie will be granted based on age of user & rating of movie.**
- **For Ex. i.e. childern will be allowed to watch movie with G rating**

$$R1: can_{access}(u, m, e) \leftarrow$$
$$(Age(u) \geq 21 \wedge Rating(m) \in [R, PG13, G]) \vee$$
$$(21 \geq Age(u) \geq 13 \wedge Rating(m) \in [PG13, G]) \vee$$
$$(Age(u) < 13 \wedge Rating(m) \in [G])$$

Advance Policy: Premium customer can view new releases.
For this, new rule R2 can be formed and can be aggregated with R1 to force both policies

$$R2: can_{access}(u, m, e) \leftarrow$$
$$(MemberType(u) = 'Premium') \vee$$
$$(MemberType(u) = 'Regular' \wedge$$
$$MovieType(m) \notin ['NewRelease'])$$
$$R3: can_{access}(u, m, e) \leftarrow R1 \wedge R2$$

ABAC

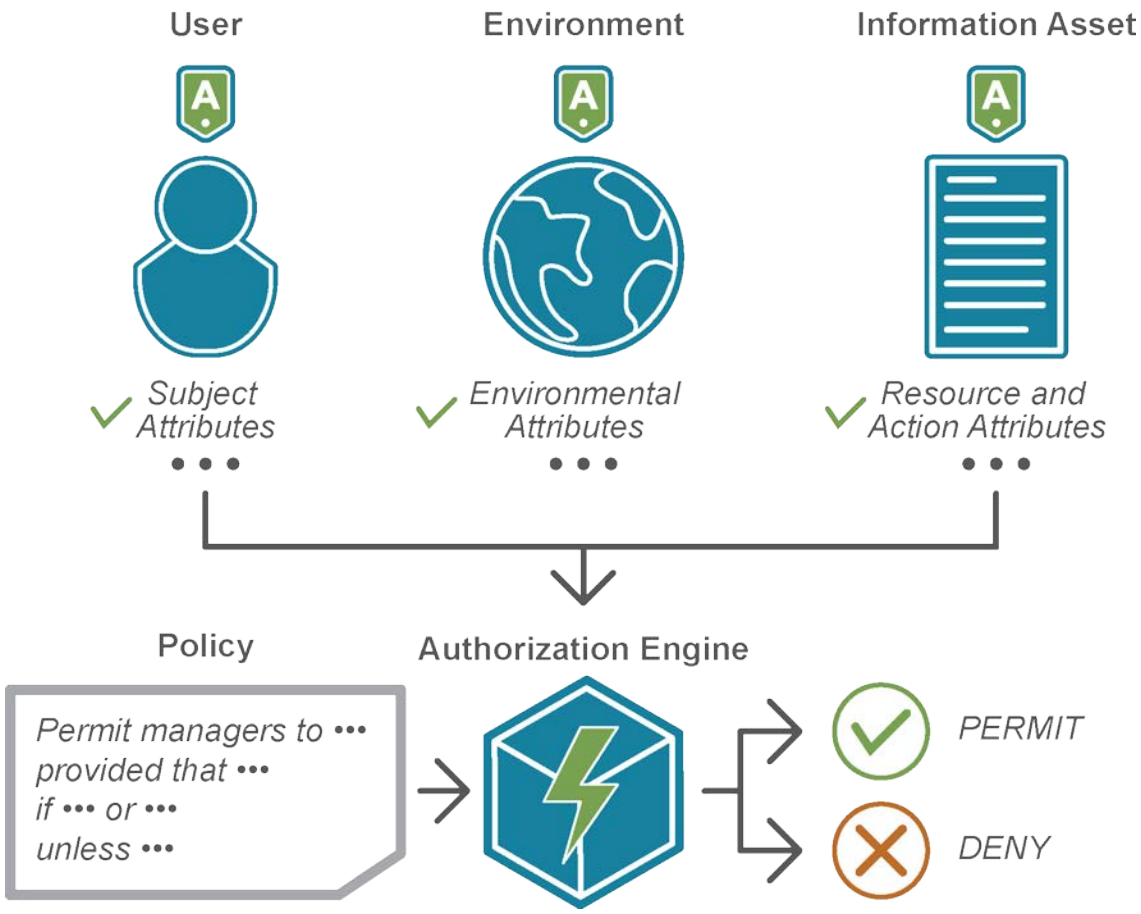


Policies bring attributes together to make it all work

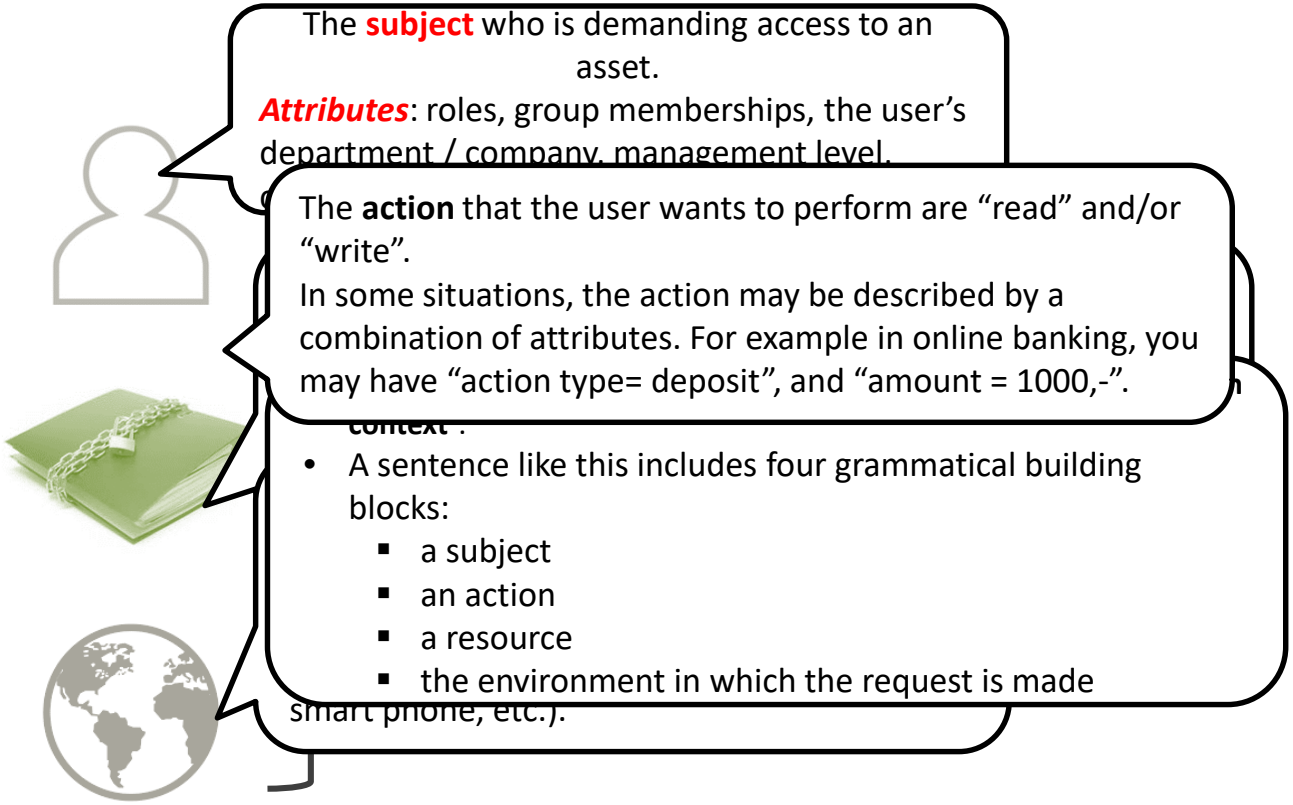
Policies grant&deny Access

Attributes

- ABAC can control access based on three different attribute types: **user attributes, attributes associated with the application or system to be accessed, and current environmental conditions**.
- It's also important to note that attributes can be about anything or anyone:
  - There can be **user or subject attributes** like name, ID, role, department, status
  - There can be **attributes to actions** like create, read, update, and delete (**CRUD**) and approve.
  - There can be **resource attributes** like the bank : bank account, bank balance, or resource clarification like top secret, public access
  - There can also be **attributes about the context** of an interaction: time, location, device



Attribute based access control



Attributes

- Subjects are principals who try to access resources to perform their tasks.
- Attributes are characteristics associated with the subject, resource or environment of the corresponding system.
- These attributes are classified as **subject attributes, resource attributes and environmental attributes**.
- **User Attributes**. Attributes of the subjects of the system. The **subject** who is demanding access to an information asset.
  - May include attributes like id, age, name, office number, job title, role, security clearance, home address, date hired, trust level (e.g. how trusted the user is by the system), etc.
  - General attributes describing the subject, for instance roles, group memberships, the department / company to which the user belongs, management level, certifications or competencies, user ID, etc. These attributes may also be available in the token received from the authentication service used during login, for instance a SAML token.
- The **action** that the user wants to perform. attributes that describe the action being attempted e.g. read, delete, view, approve...
  - Common action attributes in authorization requests are “read” and/or “write”. In more complex scenarios, the action may be described by a combination of attributes. When you access your online bank, the action may for instance be described by multiple attributes, such as “action type=transfer”, and “amount=$500”.
- **Object Attributes**. Attributes of the resources of the system.
  - May include attributes about the meta-data related to the object such as author, date created, last modified, size, file type, security level, etc., or the contents of the object such as patient name (e.g. for health records), student number (e.g. for student records), title of chapter 1, etc
  - The **resource** identifying the information asset or object impacted by the action. Resource attributes render the characteristics that identify the information asset. In eHealth applications the resource is probably a part of the health record identified by the patient ID etc.

Attributes

- **Contextual or Environmental Attributes:** Attributes derived from the current state of the system's environment. The **environment** identifying the context in which access is requested.
  - Attributes that deal with time, location or dynamic aspects of the access control scenario
    - For example, current time, day of the week, number of users logged in, free space, CPU usage, etc.
- **Connection Attributes**: Attributes that only apply to the current session of a user.
  - For example, IP address, physical location (e.g. for mobile systems), session start date/time, current session length, host name, number of access requests made, etc.
- **Administrative Attributes:** Configuration attributes that apply to the whole system and are either manually set by an administrator or by some automated process.
  - These could include a threat level (e.g. different policies could be used depending on whether or not the system was likely to be attacked), minimum trust level (e.g. the minimum amount of trust required for a user to access the system), maximum session length (e.g. the maximum allowable length of a session), etc.
- Ideally, these attributes are all properties of the elements in the system and do not need to be manually entered by administration (e.g. many of the attributes about an object come from it's meta-data).
- Access policies can be created using policy languages, limiting access to certain resources or objects, based on the result of a Boolean statement comparing attributes, for example "user.age >= 18 OR object.owner == user.id" or "TIME > 8:00AM AND TIME < 5:00PM".
- This allows for flexible enforcement of real world policies, while only requiring knowledge of some subset of attributes about a given user (as opposed to knowing their identity and to what roles or permissions they have been manually assigned).

Attribute-based Authorization

- Among these attributes, the **resource and action attributes are internal to the system** in the sense that these attributes can be defined within the system. There is no need for any entity to verify these attributes.
- On the other hand, **subject and environmental attributes are external to the system**, meaning that they are provided to the system by external entities.
- As such the authorization system requires that these attributes and their values be asserted by trusted entities.

Policies

- Policies are statements that bring together attributes to express what can happen and is not allowed. Policies in ABAC can be granting or denying policies. Policies can also be local or global and can be written in a way that they override other policies. Examples include:
  - Deny access to this document if user is not from a specific country
  - A user can edit a document if they are the owner and if the document is in draft mode
  - Deny access before 9am

Authorization Policy Languages

- Modern authorization systems are tasked to provide a large number of users secure authorization to a large number of resources which are often stored in distributed repositories, So the resultant system becomes very complex.
- If this **authorization system is developed as part of the application code**, then it **will be hard to analyze the system** and to verify if all the requirements are being met. Also, each system will require its **own customized authorization system developed for a specific application**.
- To avoid these complexities, specific languages have been developed to specify authorization requirements.
  - These languages have a special syntax for specifying the authorization requirements in an abstract fashion and their specific implementation can be customized to requirements of a single application.
  - Some of the common languages, like EPAL and XACML (Extensible Access Control Markup Language), are XML based, but other proposed languages like SecPAL are defined more in terms of formalism with a suggested XML schema . These XML based schemas make them especially suitable for use in web services.
  - XACML is an OASIS standard for specifying access control policies.
  - It is an attribute-based authorization language where access control rules are specified as a combination of subject, resource, action and environmental attributes. XACML is fast becoming one of the most popular standards in the industry as it supports flexible and ne-grained authorization policies.
  - These rules are easy to compose and can be comprehended by human beings, since they are XML based. The current recommendation is XACML v 3.0.

Policy Language Standards

- A critical component of ABAC, although not strictly part of the ABAC model, is the access control policy language used to define policy rules for a system.
- These languages, while not models in themselves (as is sometimes erroneously implied), are either generic access control language standards (such as XACML) or languages created specifically for use with a single model.
- XACML standard
  - eXtensible Access Control Markup Language (XACML) a standard created by the Organization for the Advancement of Structured Information Standards (OASIS), is one of the most frequently referenced works in ABAC literature.
  - XACML is an XMLbased access control policy language that is notable for it's support of attribute-based policies and used in multiple access control products.
  - Open source implementations available:
    - AuthZForce(Java)
    - Balana(Java)
    - OpenAZ(Java)
- Security Assertion Markup Language (SAML], also developed by OASIS, provides a standardized markup language and protocol for exchanging attribute based authorization and authentication information between service providers, identity/attribute providers and users.

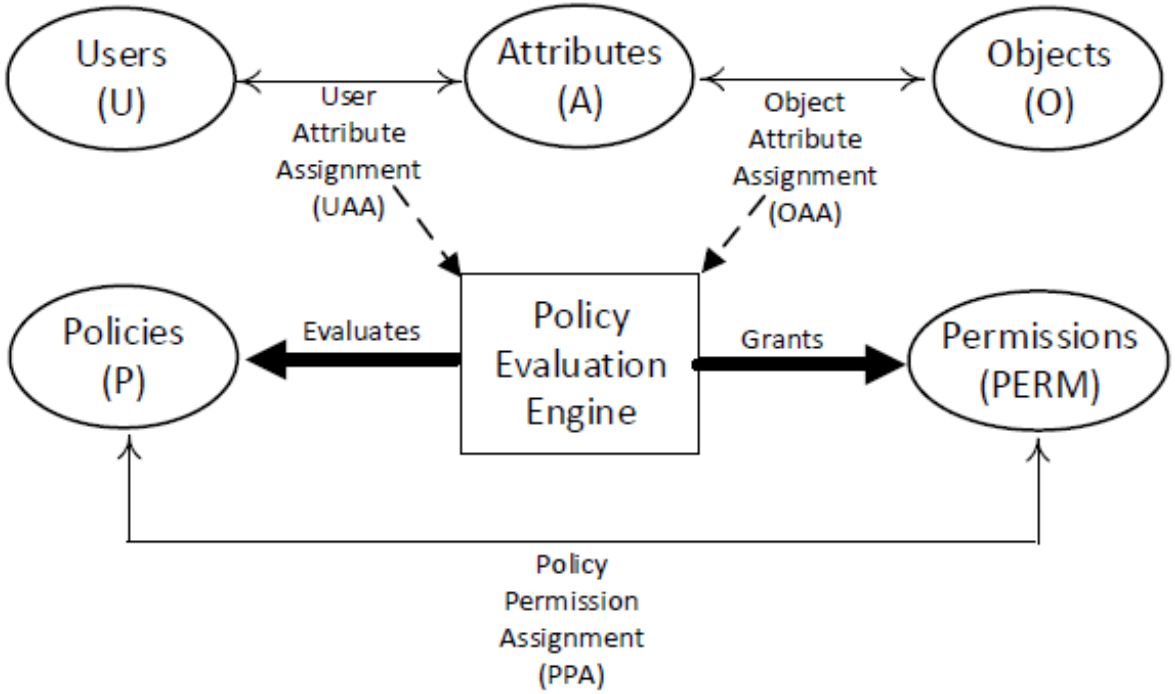
The XACML Standard

- The XACML policy language is as expressive as a natural language.
- For example, consider the following sentence: “A user wants to do something with an information asset in a given context”.
- A sentence like this includes four grammatical building blocks:
  - a subject
  - an action
  - a resource
    the environment in which the request is made

Core ABAC Model

- A simplified ABAC model based on common elements found in most ABAC models.
- The following are the most common elements of an ABAC system and are present in most models:
  - Users (U). The set of all users that may access the system.
  - Objects (O). The set of all objects protected by the system.
  - Attributes (A). The set of all attributes (given by a unique name) in the system.
  - Permissions (PERM). The set of all possible permissions that may be granted to users.
  - Policies (P). The set of all policies that govern access in the system. Normally these policies are written in a policy language and in some way related to permissions they grant.
- Users Attribute Assignment (UAA).
  - The assignment of attributes to users.
- Object Attribute Assignment (OAA).
  - The assignment of attributes to objects.
- Policy Permission Relation (PPR).
    - The relationship between policies and the permissions they grant.
- Policies in the P set are commonly Boolean statements involving attributes and constants such as “user.age >= 18” (grants access if the user is 18 or more years of age) or “user.id == object.author” (grants access if the user is the author of the file).
  - When an access request is made by a user it is evaluated against the set of policies (P) given the assigned attributes of the user making the request and the object being requested. In many models, access requests are not conducted directly by the user but indirectly through a session that may contain a subset of the user’s attributes

Core ABAC



Thin solid arrows denote many-to-many relations, thick solid lines denote relation with policy engine and doted lines denote information used by the policy engine to evaluate a given policy. Ovals represent ABAC model elements.

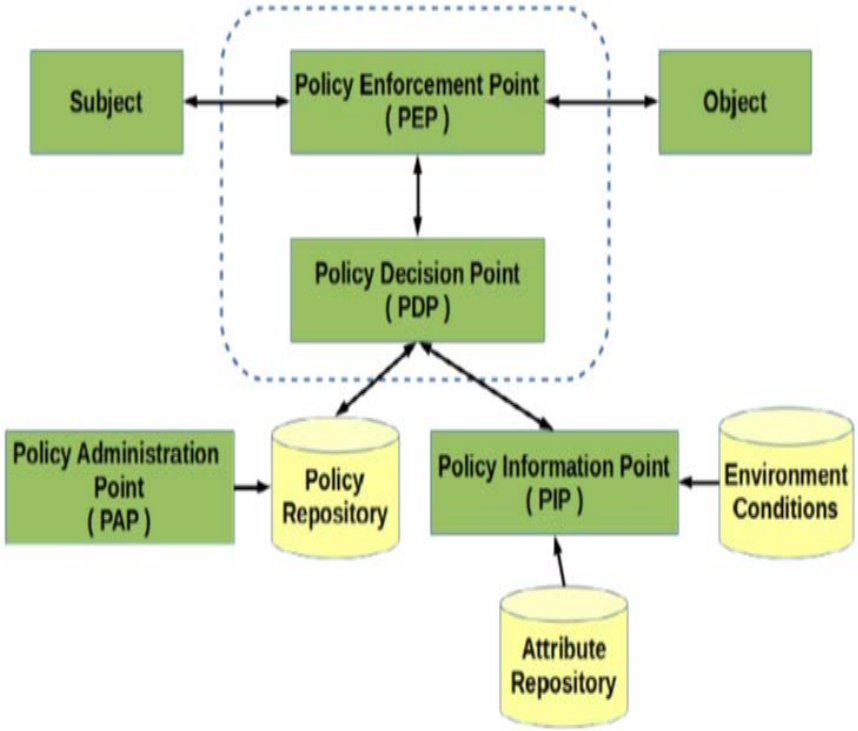
ABAC Architecture

The PEP or Policy Enforcement Point: it is responsible for protecting the apps & data you want to apply ABAC to. The PEP inspects the request and generates an authorization request from it which it sends to the PDP.

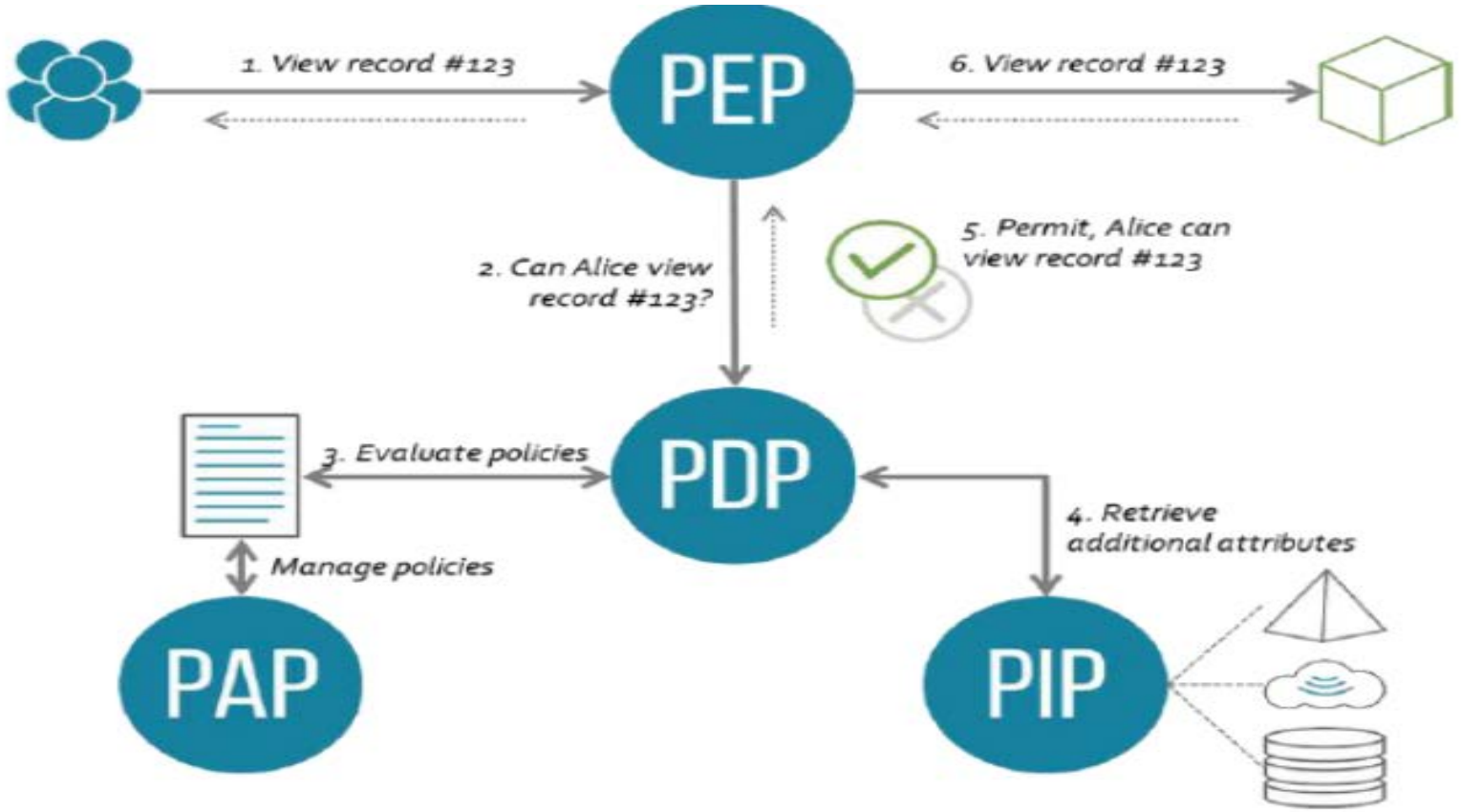
The PDP or Policy Decision Point: brain of the architecture. This is the piece which evaluates incoming requests against policies it has been configured with. The PDP returns a Permit / Deny decision. The PDP may also use PIPs to retrieve missing metadata.

The PIP or Policy Information Point: bridges the PDP to external sources of attributes e.g. LDAP or databases.

In ABAC, PEP, PDP, PAP & PAP may be on same machine or may be physically separated. Such distributed framework give rise to ABAC Enterprise.

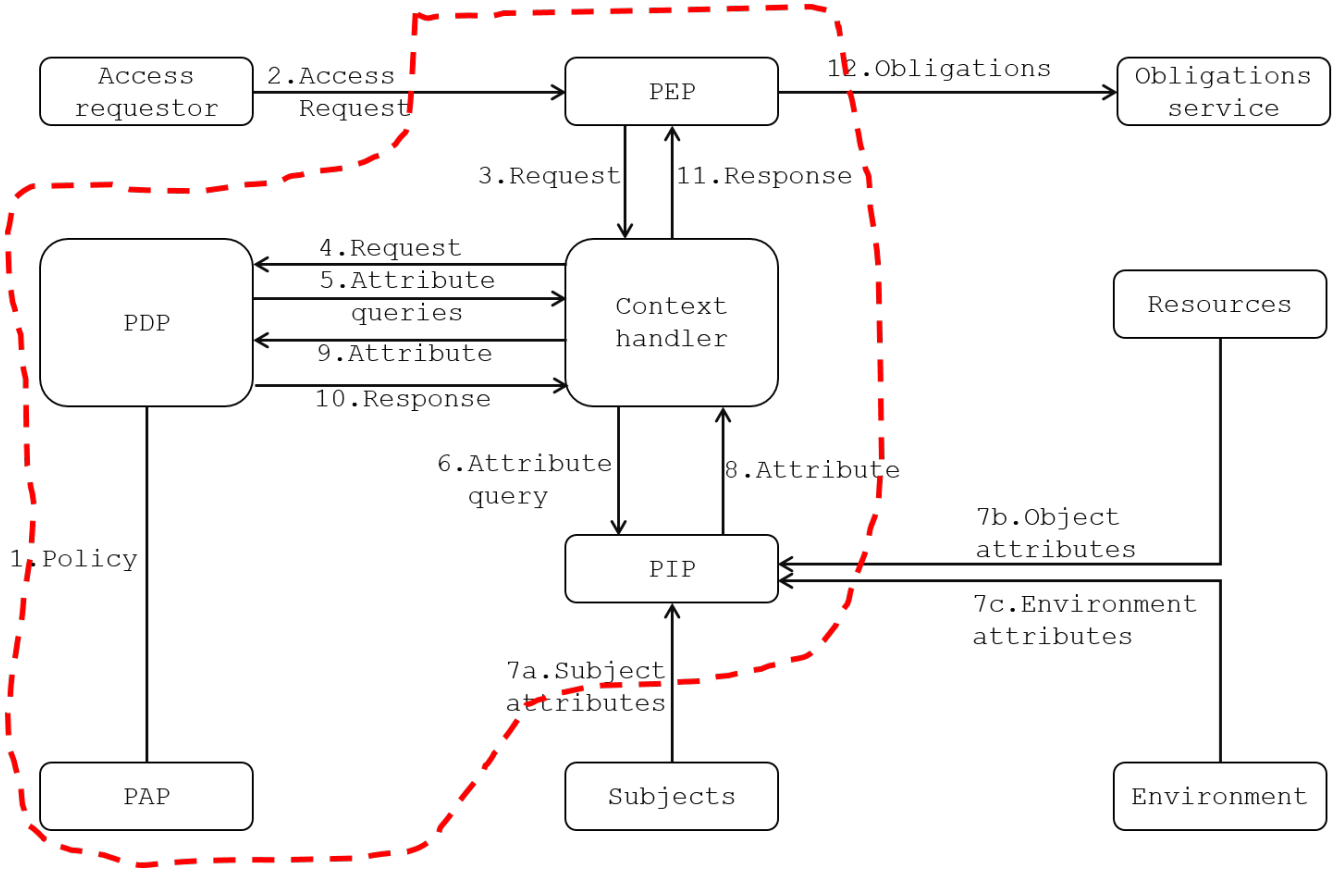


ABAC Model

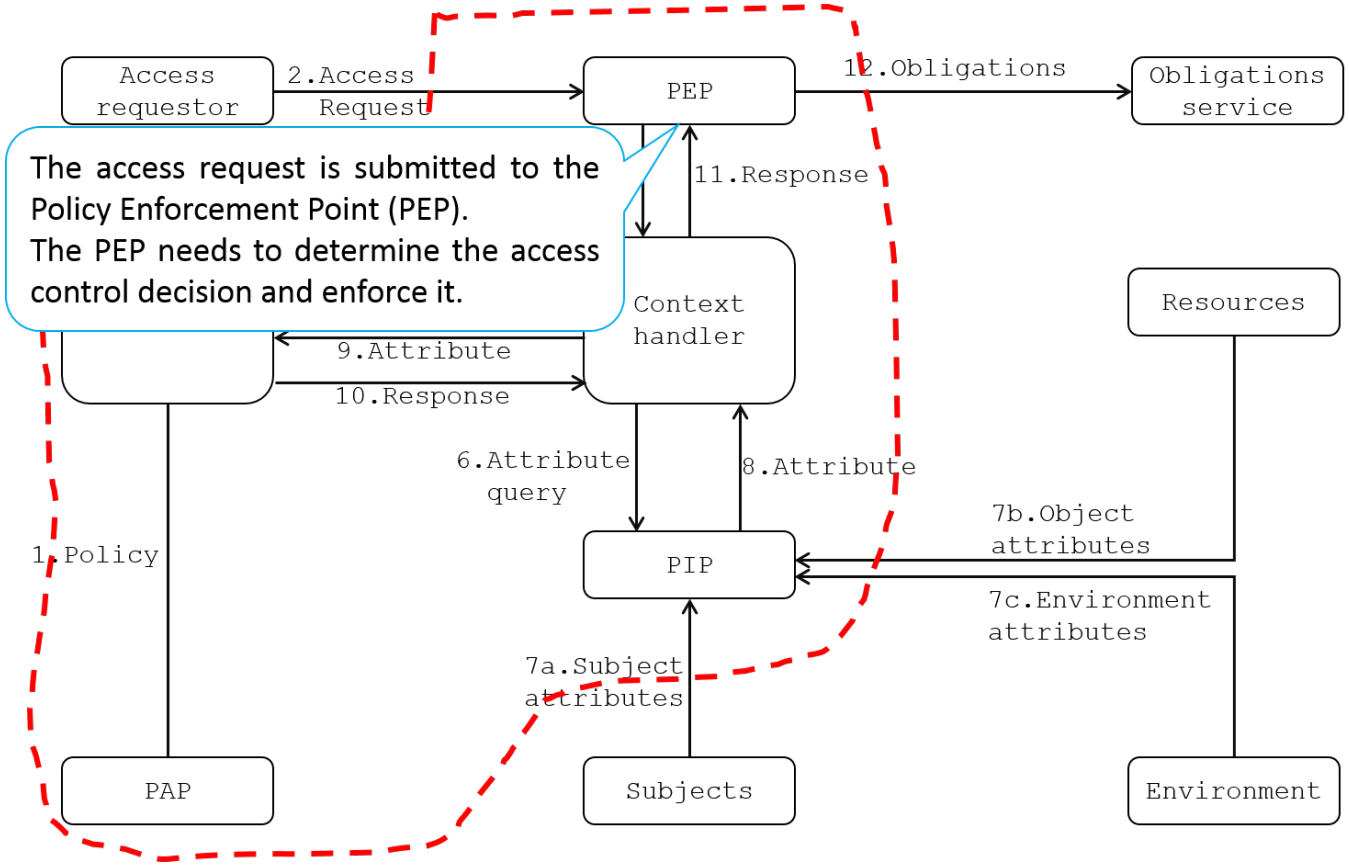


In ABAC a “Policy Decision Point” and “Policy Enforcement Point”, are the deciding factors when it comes to permitting or denying the access request.

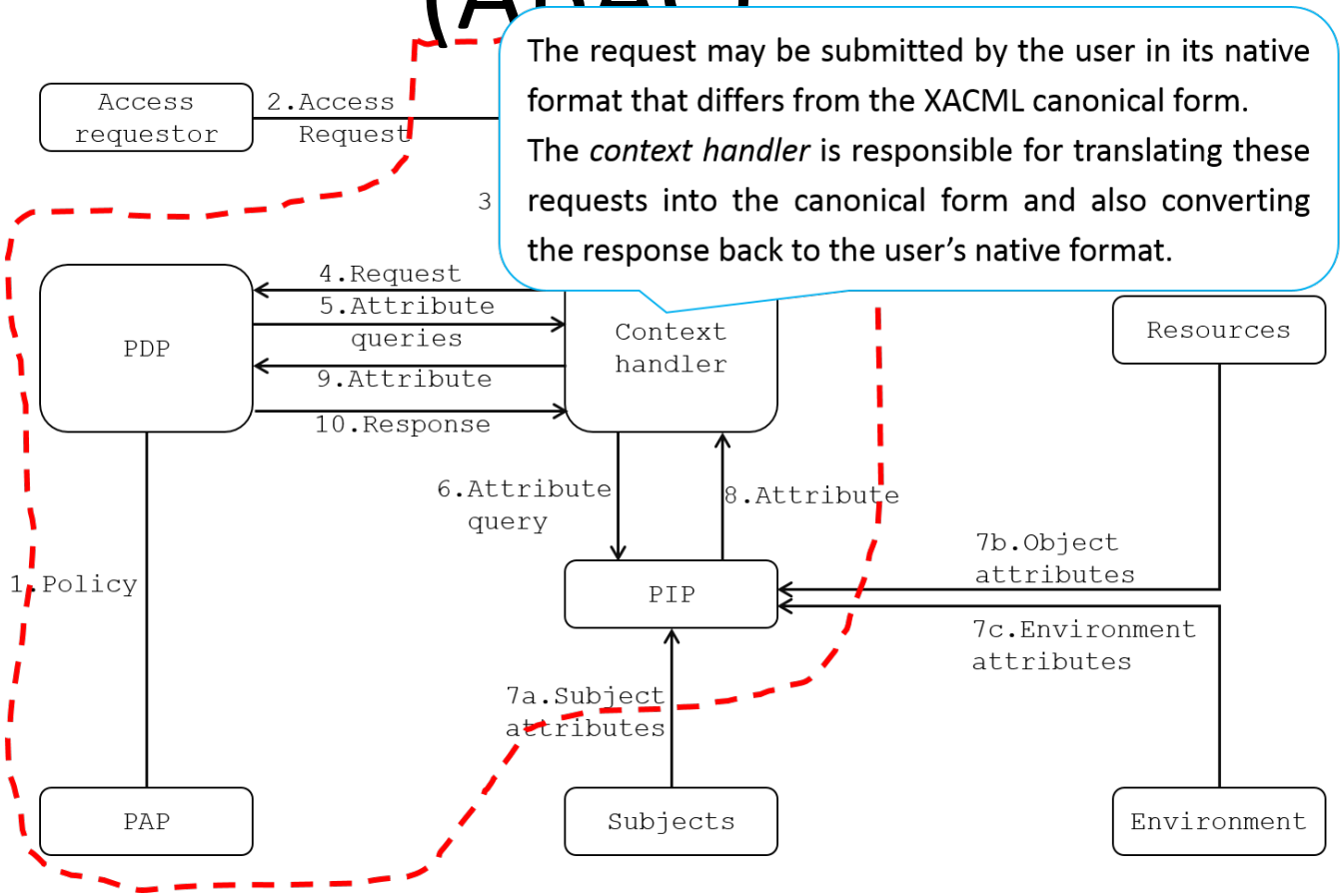
Attribute-Based Access Control (ABAC)



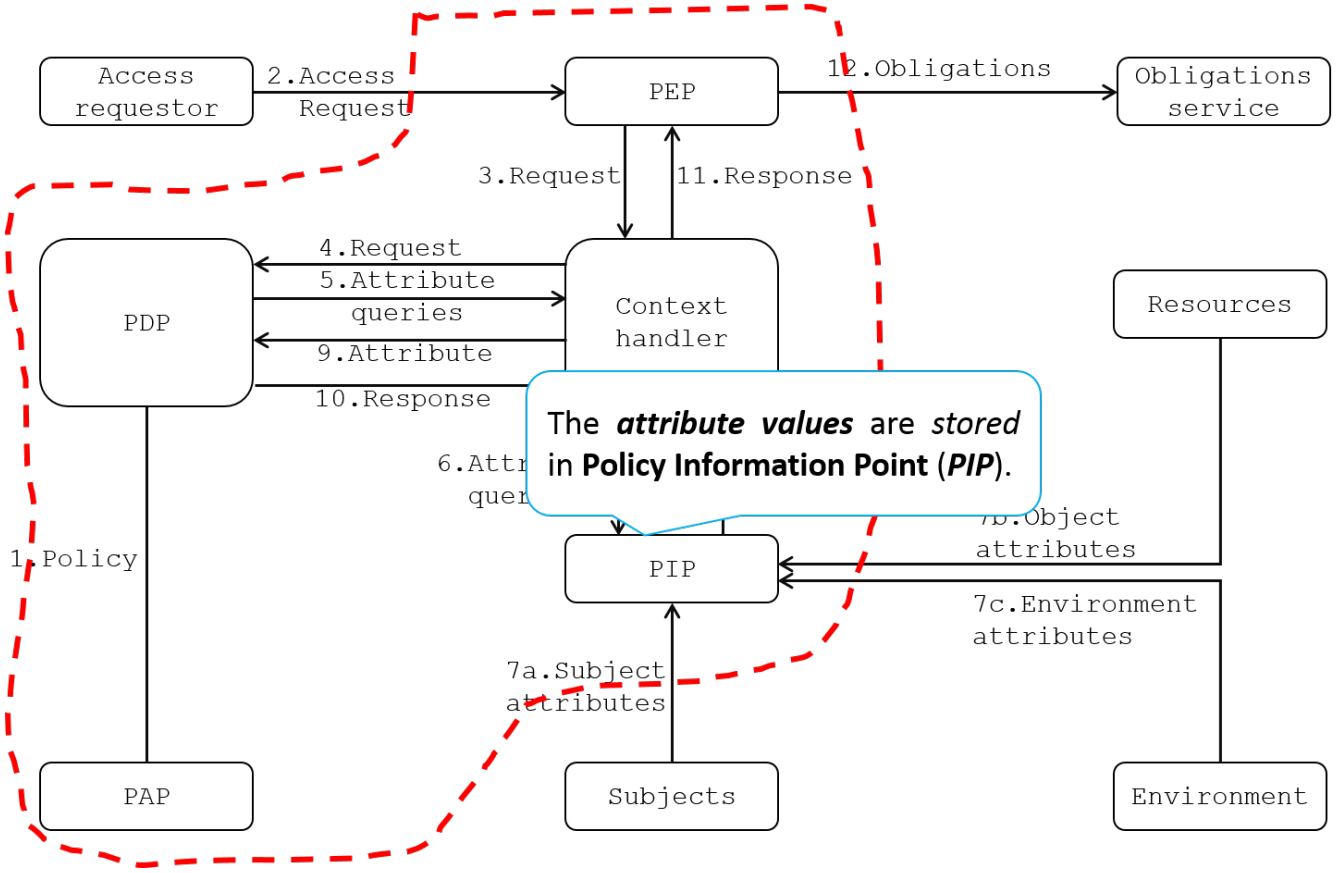
Attribute-Based Access Control (ABAC)



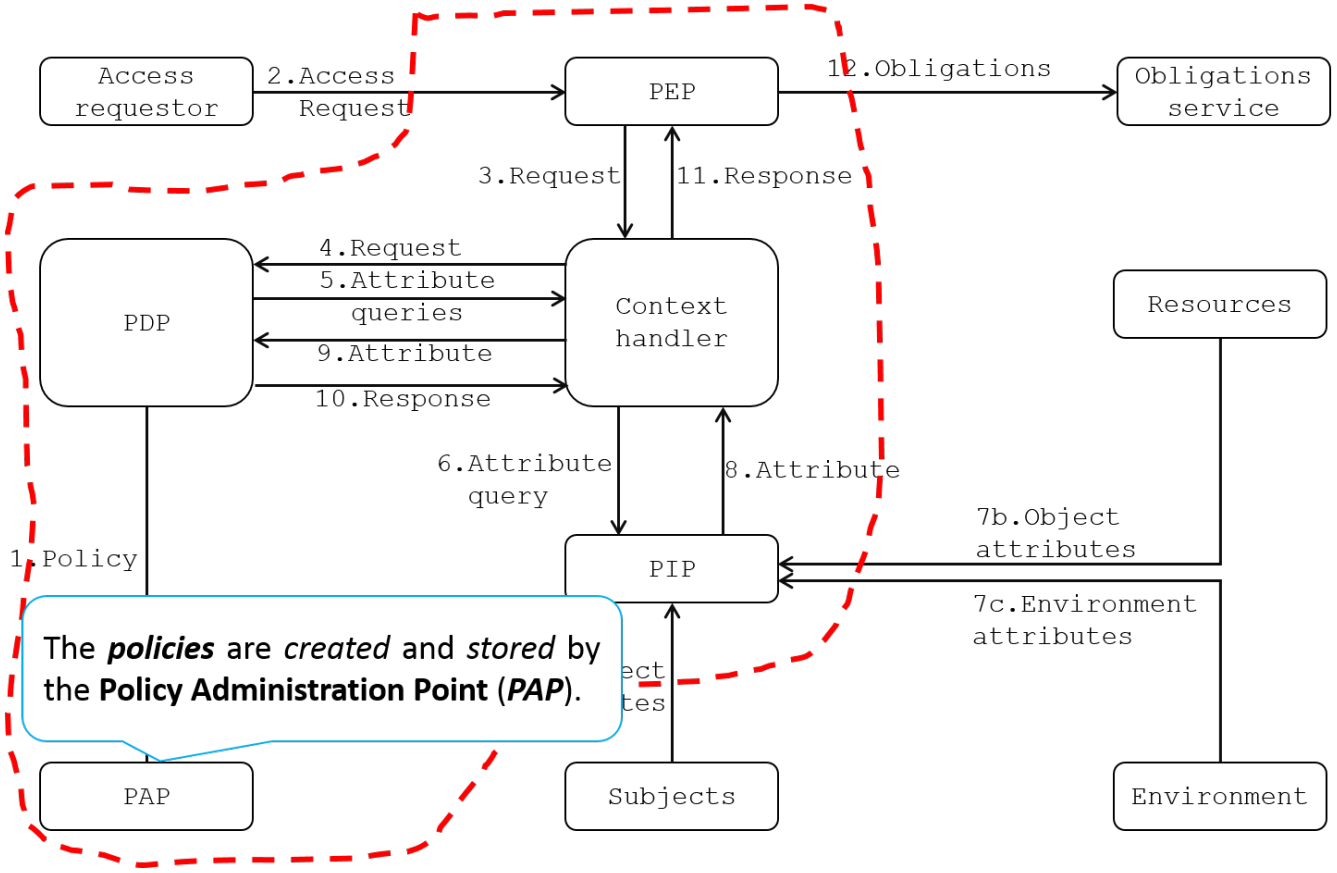
Attribute-Based Access Control (ABAC)



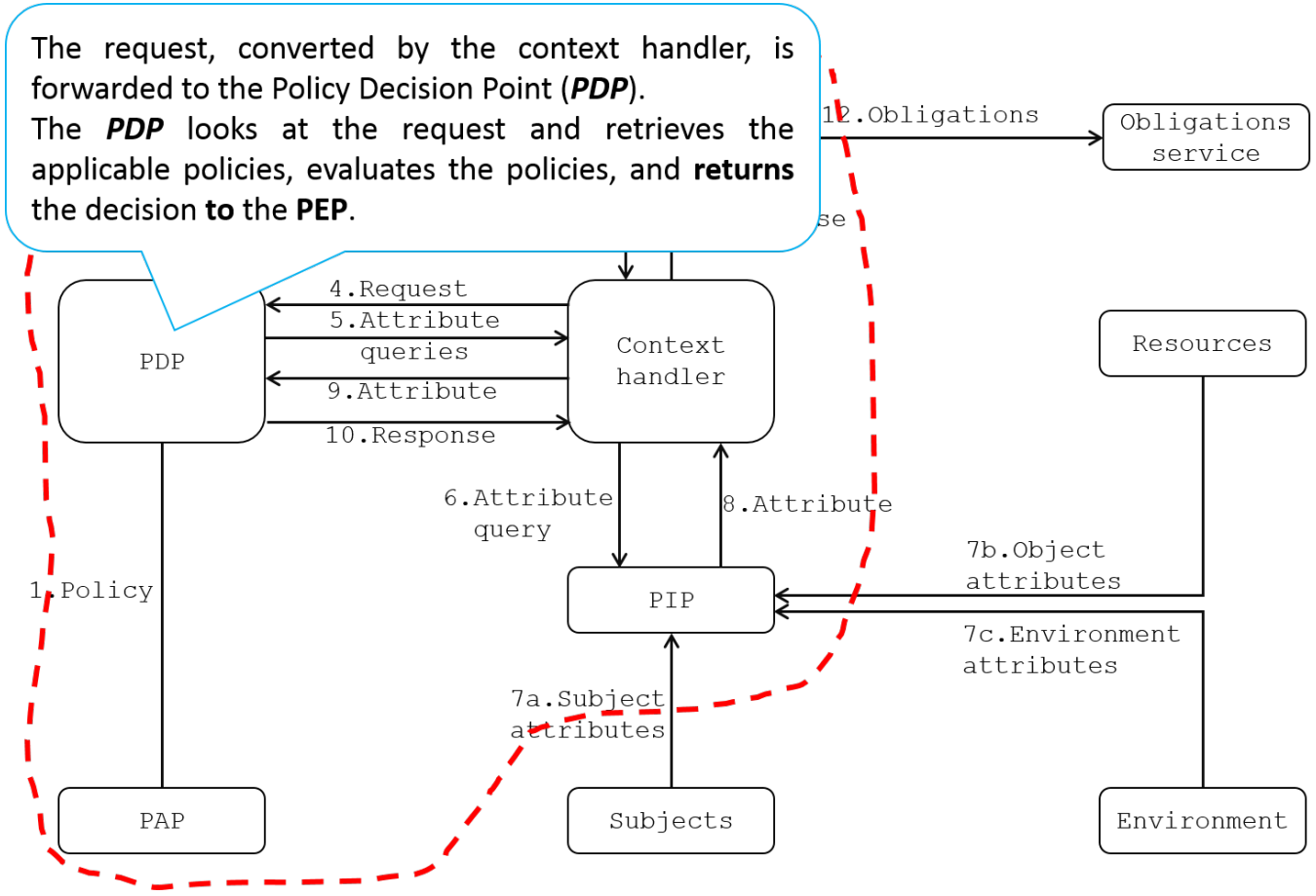
Attribute-Based Access Control (ABAC)



Attribute-Based Access Control (ABAC)



Attribute-Based Access Control (ABAC)



The benefits of an ABAC

- The benefits of an ABAC approach are significant:
  - Access decisions are centrally managed via policies rather than by individual application managers, which means that they are applied consistently across the organization, and administered by business managers rather than controlled by IT personnel;
  - Software development is simplified by the incorporation of “policy enforcement point” code which externalizes the access decision to a “policy decision point;”
  - Decisions are made at runtime based on attributes that may be combined to form fine-grained decisions; changes in access status are immediately recognized when attributes are updated.
- From a business viewpoint,
  - This translates to a vastly reduced risk profile; access rights will be modified as soon as the source system reflects the change.
  - No longer will “old permissions” stay in the system when an attribute changes or a staff member leaves the organization’s employ.
  - Access rights are no longer based on access control lists that require manual intervention in order to be updated. It also means lower costs.
  - Without manual intervention, there are no recurring costs for updating system access rights.
  - Another benefit is that software development is less expensive, removing the cost of developing sophisticated access control logic for applications.

Advantages of ABAC

- **ABAC provides the following advantages over the traditional RBAC model:**
  - **ABAC permissions scale with innovation.** It's no longer necessary for an administrator to update existing policies to allow access to new resources.
  - **ABAC requires fewer policies.** Because you don't have to create different policies for different job functions, you create fewer policies. Those policies are easier to manage. **ABAC requires fewer policies.** Because you don't have to create different policies for different job functions, you create fewer policies. Those policies are easier to manage.
  - **Using ABAC, teams can change and grow quickly.** This is because permissions for new resources are automatically granted based on attributes.
  - **Granular permissions are possible using ABAC.** When you create policies, it's a best practice to grant least privilege. Using traditional RBAC, you must write a policy that allows access to only specific resources. However, when you use ABAC, you can allow actions on all resources, but only if the resource tag matches the principal's tag.
  - **Use employee attributes from your corporate directory with ABAC.** You can use ABAC to allow or deny permissions based on those attributes.

ABAC Pros & Cons

Pros

- Dynamic & fine grained access control
- Scalable
- Consider environmental conditions
- Can be mapped to MAC & RBAC model
- Can easily adapt to Risk (RAdAC)
- Easy administration

Cons

- Attribute needs provisioning and maintenance
- Possibility of attribute explosion
- Complex to analyze

RBAC VS ABAC

Characteristic	RBAC	ABAC
Flexibility	(For small and medium-sized organizations)	
Scalability		
Simplicity	Easy to establish roles and permissions for a small company, hard to maintain the system for a big company	Hard to establish all the policies at the start, easy to maintain and support
Support for simple rules		
Support for complex rules		
Support for rules with dynamic parameters		
Customizing user permissions	(Every customization requires creating a new role)	
Granularity	Low	High

The Challenges of an ABAC

- Firstly, a mature identity and access management environment is a prerequisite because a data model that has defined the “authoritative source” for attributes, and has efficient “source of truth” repositories, is an ABAC requirement.
- Authentication is a mission-critical service; it must be highly available and it cannot tolerate excessive network latency. This means that in some cases, a synchronization business model might be required whereby remote data stores are synchronized to a local directory or database.
- Policy Enforcement Points can also be a challenge, particularly if there are many legacy applications to integrate. Old systems with built-in access control logic will need to be modified to externalize the authorization function and take advantage of the more fine-grained control that ABAC can provide. Vendors can assist this development with code and APIs that facilitate entitlement management.
- Policy Decision Point management can be complicated for distributed organizations or hybrid situations in which some applications are in the Cloud. While policy management must remain centralized, the decision points will need to be distributed and a mechanism to keep them current will be required.
- Finally, Policy Administration must be addressed. With the movement of policy management from IT to business, there is a need for business units to understand their requirements and have a capability to encode their requirements in policies. In some organizations this may be hard to achieve and a centralized policy management facility, servicing multiple business units, will be required.
- It is also vital that these ABAC capabilities be developed and deployed using a common foundation of concepts and functional requirements to ensure the greatest level of interoperability possible

Current Research

- The current research is focused on two areas in attribute-based authorization systems.
  - First, specifying **authorization policies in multi-authority systems** where there are multiple stakeholders in the disclosure of sensitive data. The research proposes to consider all the relevant policies related to authorization in real time upon the receipt of an access request and to resolve any differences that these individual policies may have in authorization.
  - Second, to enable a lot of entities to participate in the authorization process by asserting attributes on behalf of the principal accessing resources. Since it is required that these asserted attributes be trusted by the authorization system, it is necessary that t**hese entities are themselves trusted by the authorization system**.
- **Dynamic authorization**
  - In current authorization systems, the attribute-based policies are predefined and are therefore static. In systems where sensitive information like financial or medical data is shared, the access environment is dynamic. Some systems try to account for this dynamic behavior by including special conditions for unexpected or dynamic environment but a large majority of systems don't have any mechanisms to do even that. The problem with such mechanisms is that they make the authorization policies very difficult to understand; they neither have fallback mechanisms in case of emergencies or exceptions, nor do they have provisions for using these conditions to dene alternate policies for different access environments.

Research Issues

- **1. In attribute-based systems, the authorization system should have access to trusted attributes to evaluate the policies**. Access to these trusted attributes is a major challenge because there are very few entities which are pre-trusted by the authorization system and evaluating trust in attributes provided by unknown entities is an open question.
- **2. Using dynamic attributes for authorization and leveraging them to provide context aware authorization is a challenge in these systems**. This is primarily because the authorization system needs to have different policies for different contexts and it needs to dynamically switch the authorization context considering the relevant attributes.
- **3. Modern authorization systems that provide access to sensitive medical data typically have a number of stake holders in maintenance, storage and disclosure of this data, each of which denes an authorization policy**. One of the challenges in this context is to identify and combine all the relevant policies in runtime and to resolve the conflicts in the decisions arising from each of these individual policies.
- **4. Semantic-Based Access Control (SBAC)**
  - Idea: ABAC + semantic technologies
    - making decisions semantically as well as considering the semantic relationships for inferring implicit policies from explicit ones
    - Formally define entities and their attributes and relationships using an ontology
    - Describing relations for specific conditions using rule markup languages
    - Separation of ***ontology*** management from ***access*** management Two parts:
      - An ontology management system provides the extended user and resource attributes
      - An access control system uses the extended attributes for access evaluation

Summary

- ABAC can be see as authorization that is:
  - **Externalized**: Access control is externalized from the business logic
  - **Centralized**: Access control policies are maintained centrally
  - **Standardized**: Access control policies use XACML, the eXtensible Access Control Markup Language, the standard defined by OASIS and implemented by most ABAC solutions
  - **Flexible**: ABAC can be applied to APIs, databases, and more
  - **Dynamic**: Access decisions are made dynamically at runtime
  - **Context-based** / Risk-based: ABAC can take time, location, and other contextual attributes into account when reaching decisions.

Acknowledgements

- Slides of Prof. Ravi Sandhu on Attribute-Based Access Control Models and Beyond – 2015
- Slides of Axiomatics on What is Attribute Based Access Control?-OWASP Chicago October 2016
- Slides of Prof. Hamed Arshad on Semantic Attribute-Based Access Control-An overview of the existing approaches- Department of Informatics University of Oslo, March 2018
- **The above material is used only for academic purpose.**

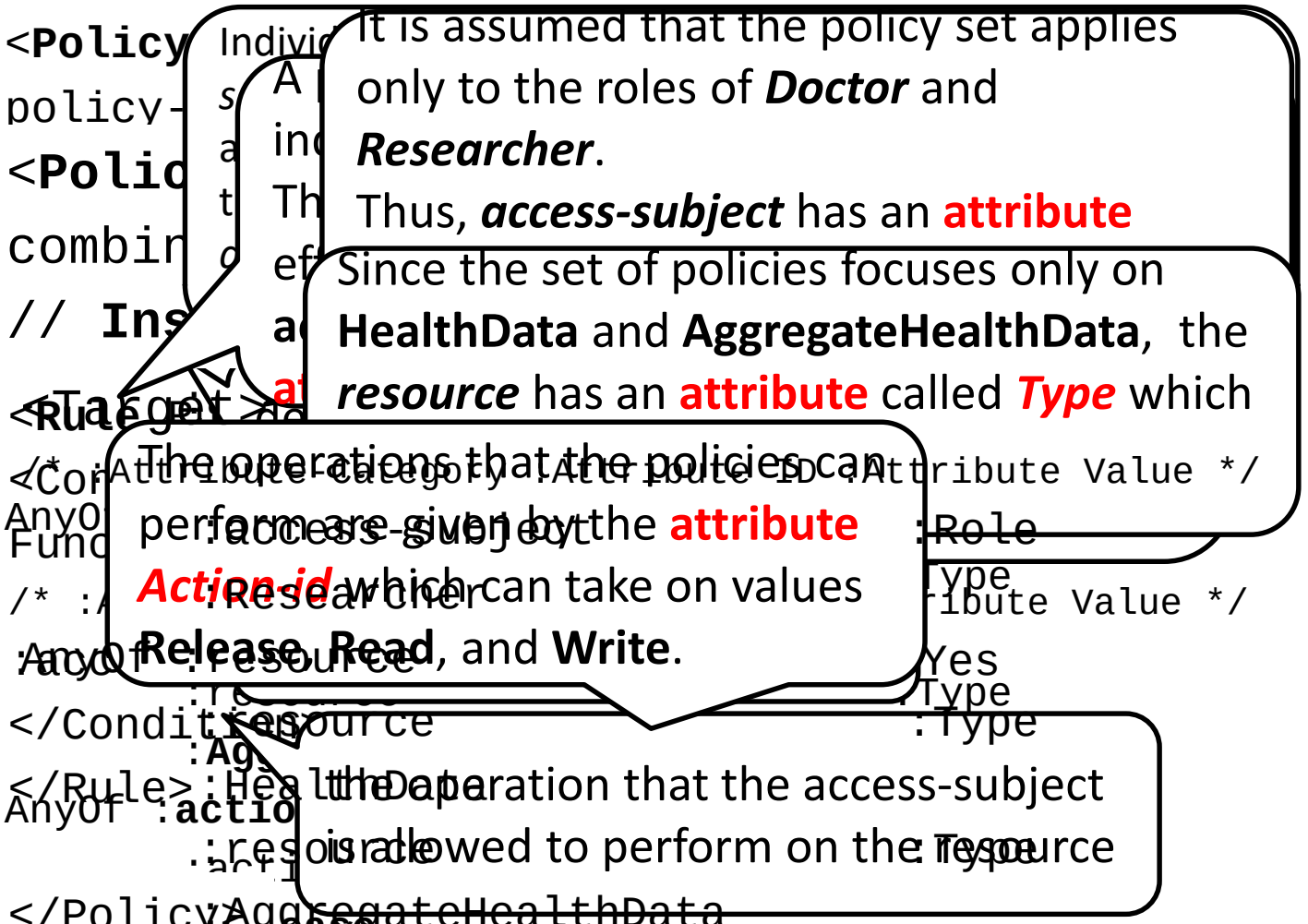
Example

Example XACML Policy Specification

- The example is that of a data store that gets clinical health data from various healthcare providers.
- Clinical data is received by the data store which must be processed as per the instructions of the patient and the legal rules before it is distributed to authorized researchers.
- The aim of this example is to show how the patient's privacy preferences can be expressed and enforced using the ABAC framework supported by XACML.
- The example uses ***Institute 1***, ***Institute 2***, ***Patient 1*** and ***Patient 2*** policies for the purpose of demonstration.

Policies of Institute 1

- ***Institute 1*** allows release of patient-level data to **researchers** as limited datasets as long as the requesters can assert HIPAA-compliance.


It is assumed that the policy set applies only to the roles of ***Doctor*** and ***Researcher***.
Thus, ***access-subject*** has an **attribute**
Since the set of policies focuses only on **HealthData** and **AggregateHealthData**, the ***resource*** has an **attribute** called ***Type*** which
The operations that the policies can perform are given by the **attribute** ***Action-id*** which can take on values **Release**, **Read**, and **Write**.
the operation that the access-subject is allowed to perform on the resource
:Action-id :Read
:Action-id :Write

```
<PolicySet PolicySetId = "PolicySetInstitute1“ policy-combining-algorithm="permit-overrides">
<Target>
/*:Attribute-Category :Attribute ID :Attribute Value */
AnyOf :access-subject
              :access-subject :Role        :Researcher
              :access-subject :Role        :Doctor
AnyOf :resource
              :resource      :Type         :HealthData
              :resource      :Type         :AggregateHealthData
AnyOf :action
              :action        :Action-id    :Release
              :action        :Action-id    :Read
              :action        :Action-id    :Write
</Target>
<Policy PolicyId ="Policy1“ rule-combining-algorithm="deny-overrides">
// Institute 1 Rules //
<Target>
/* :Attribute-Category :Attribute ID :Attribute Value */
:access-subject                              :Role         :Researcher
AnyOf :resource
              :resource      :Type         :HealthData
              :resource      :Type         :AggregateHealthData
:action                      :Action-id    :Release
</Target>
<Rule RuleId = "I1R1" Effect="Permit">
<Condition>
Function: string-equal
/* :Attribute-Category :Attribute ID :Attribute Value */
:access-subject             :HIPAA Comp    :Yes
</Condition>
</Rule>
</Policy>
</PolicySet>
```

Policies of Institute 2

- ***Institute 2*** only allows release of **aggregate statistics** (mean, count) and **no** release of patient-level data **to researchers**.
- Moreover, this institute **does not allow statistics** to be released that are **calculated** based on **less than 10 patients**.
- This institute **does not require** data receivers to be **HIPAA compliant**, **but denies** all **individual patient-level data** requests.

```
<PolicySet PolicySetId = "PolicySetInstitute2“
policy-combining-algorithm="permit-overrides">
<Target>
</Target>
<Policy PolicyId ="Policy2“
rule-combining-algorithm="deny-overrides">
<Rule RuleId = "I2R1" Effect="Permit">
<Condition>
Function: and
Function: string-equal
AnyOf :Doctor
AnyOf :Release
Function: greater than
AnyOf :Release
</Condition>
</Rule>
</Policy>
</PolicySet>
```

```
/*:Attribute Category :Attribute ID :Attribute Value */
AnyOf :access-subject
        :access-subject  :Role :Researcher
        :access-subject  :Role :Researcher
/* :Attribute-Category :Attribute ID :Attribute Value */
AnyOf :resource  :Type :AggregateHealthData
        :resource  :Type :AggregateHealthData
/* :Attribute-Category :Attribute ID :Attribute Value */
AnyOf :action    :Action-id :Release
        :action    :Action-id :Read
        :action    :Action-id :Write
/*:Attribute Category :Attribute ID :Attribute Value */
:AggregateHealthData  :PatientCount 10
```

```
<PolicySet PolicySetId = "PolicySetInstitute2“ policy-combining-algorithm="permit-
overrides">
<Target>
/*:Attribute-Category :Attribute ID :Attribute Value */
AnyOf :access-subject
            :access-subject          :Role       :Researcher
            :access-subject          :Role       :Doctor
AnyOf :access-resource
            :resource   :Type        :HealthData
            :resource                :Type       :AggregateHealthData
AnyOf :action
            :action           :Action-id        :Release
            :action           :Action-id        :Read
            :action           :Action-id        :Write
</Target>
<Policy PolicyId ="Policy2“ rule-combining-algorithm="deny-overrides">
<Target>
/*:Attribute-Category :Attribute ID :Attribute Value */
            :access-subject          :Role       :Researcher
AnyOf :resource
            :resource   :Type        :AggregateHealthData
            :action           :Action-id        :Release
</Target>
<Rule RuleId = "I2R1" Effect="Permit">
<Condition>
Function: and
Function: string-equal
/* :Attribute-Category :Attribute ID :Attribute Value */
            :resource   :Type        :AggregateHealthData
Function: greater than
/* :Attribute-Category :Attribute ID :Attribute Value */
            :resource   :PatientCount 10
</Condition>
</Rule>
</Policy>
</PolicySet>
```

Policies of *Patient 1*

- ***Patient 1***, a former drug addict, is **unwilling** to share his/her **past drug use information**.

```
<PolicySet PolicySetId = "PolicySetPatient1“
policy-combining-algorithm="permit-overrides">
 <Policy PolicyId = "PP1“ rule-combining-
 algorithm="deny-overrides">
  <Target>
   /* :Attribute-Category :Attribute ID :Attribute Value */
   AnyOf :access-subject
    :access-subject :Role
     :Researcher
    :access-subject :Role :Doctor
    :Researcher
   Function :string-equal
   /* :Attribute-Category :Attribute ID :Attribute Value */
    :resource :Type
     :HealthData :Type :Type
    :resource :Resource-owner
     :HealthData
   AnyOf :action
    :Patient1
    :action :Action-id :Action-id
   Function :string-not-equal
    :Release
    :resource :Content
    :purpose :Purpose-id
    :action :Action-id
    :DrugUsage
    :DrugEffect
    :Read
    :Write
  </Target>
  <Condition>
  </Condition>
 </Rule>
</Policy>
```

Patient 1 Policy for Researchers

```
<PolicySet PolicySetId = "PolicySetPatient1“ policy-combining-algorithm="permit-
overrides">
<Target>
/* :Attribute-Category :Attribute ID :Attribute Value */
AnyOf :access-subject
            :access-subject         :Role                   :Researcher
            :access-subject         :Role                   :Doctor
            :resource               :Type                   :HealthData
AnyOf :action
            :action                 :Action-id              :Release
            :action                 :Action-id  :Read
            :action                 :Action-id  :Write
</Target>
<Policy PolicyId = "PP1“ rule-combining-algorithm="deny-overrides">
// Patient 1 Policy for Researchers
<Target>
/* :Attribute-Category :Attribute ID :Attribute Value */
            :access-subject         :Role                   :Researcher
            :resource               :Type                   :HealthData
            :action                 :Action-id  :Release
            :purpose                :Purpose-id :DrugEffect
</Target>
<Rule RuleId = "P1R1" Effect="Permit">
<Condition>
Function: and
Function: string-equal
/* :Attribute-Category :Attribute ID :Attribute Value */
            :resource               :Resource-owner         :Patient1
Function: string-not-equal
            :resource               :Content                :Drug-usage
</Condition>
</Rule>
</Policy>
</PolicySet>
```

Policies of *Patient 2*

Patient 2 is a highly visible politician who **only approves** release of **treatment or visit data** at the **year level granularity**.

```
<PolicySet PolicySetId="PolicySetPatient2“
policy-combining-algorithm="permit-overrides">
<Target>
/* :Attribute-Category :Attribute ID :Attribute Value */
AnyOf :access-subject
  :access-subject   :Role
    :Researcher
  :access-subject   :Role
    :Doctor
AnyOf :Release
</Target>
```

```
<Policy PolicyId = "PP2“ rule-combining-
algorithm = "permit-overrides">
<Target>
Function: string-equal
    :resource   :Type
    :HealthData
    :resource   :Type
    :HealthData
Function: string-equal
    :purpose   :Purpose-id
    :DrugEffect
    :action        :Action-id :Read
    :action        :Action-id
    :Write
</Target>
```

```
<Rule RuleId = "P2R3" Effect="Deny">
<Condition>
Function: and
Function: string-equal
    :resource   :Resource-owner
    :Patient2
    :resource   :Duration
    :Release   :Yearly
</Condition>
</Rule>
</Policy>
```

```
<Rule RuleId = "P2R2" Effect="Permit">
<Condition>
/* :Attribute-Category :Attribute ID :Attribute Value */
Function: and
Function: string-equal
    :resource   :Resource-owner
    :Patient2
Function: string-equal
    :resource   :Duration
    :Yearly
    :action   :Action-id
</Condition>
</Rule>
<Obligations>
<Obligation Obligation Id="rewriteDate" FullfillOn ="Permit">
</Obligation>
</Obligations>
</PolicySet>
```

```
<PolicySet PolicySetId = "PolicySetPatient2“ policy-combining-algorithm="permit-overrides">
<Target>
/* :Attribute-Category :Attribute ID :Attribute Value */
AnyOf :access-subject
                :access-subject                :Role                          :Researcher
                :access-subject                :Role                          :Doctor
                :resource                      :Type                          :HealthData
AnyOf :action
                :action                        :Action-id                     :Release
                :action                        :Action-id                     :Read
                :action                        :Action-id                     :Write
</Target>
<Policy PolicyId = "PP2“ rule-combining-algorithm="permit-overrides">
<Target>
/* :Attribute-Category :Attribute ID :Attribute Value */
                :access-subject                :Role                          :Researcher
                :resource                      :Type                          :HealthData
                :action                        :Action-id                     :Release
                :purpose                       :Purpose-id     :DrugEffect
</Target>
<Rule RuleId = "P2R1" Effect="Permit">
<Condition>
Function: and
Function: string-equal
/* :Attribute-Category :Attribute ID :Attribute Value */
                :resource                      :Resource-owner                :Patient2
Function: string-equal
                :resource                      :Duration                      :Yearly
</Condition>
</Rule>
<Rule RuleId = "P2R2" Effect="Permit">
<Condition>
Function: and
Function: string-equal
/* :Attribute-Category :Attribute ID :Attribute Value */
                :resource                      :Resource-owner                :Patient2
Function: string-not-equal
                :resource                      :Duration                      :Yearly
</Condition>
</Rule>
<Obligations>
<Obligation Obligation Id="rewriteDate" FullfillOn ="Permit">
</Obligation>
</Obligations>
<Rule RuleId = "P2R3" Effect="Deny">
</Policy>
</PolicySet>
```

Access Request

- There exists a **researcher** who is requesting the data and he **is HIPAA compliant**.
- The ***researcher*** is conducting a study to assess the effectiveness of the experimental drug on hepatitis C symptoms each month in the first year after treatment.
- The ***researcher*** would **need access** to patient's **past drug history** and also **information** about the symptoms at the **month level**.

```
<Request REQ1>
<Attributes>
        :access-subject   :Subject-id     :Carol
        :access-subject   :Role           :Researcher
        :access-subject   :HIPAA Comp     :Yes
        :resource         :Type           :HealthData
/* Content */
        :resource        :Content          :Drug-usage
        :resource        :Content          :Symptoms
        :resource        :Content          :Disease
        :resource        :Content          :Duration
        :resource        :Content          :Date
/* Comparing Attribute Values */
        :content-selector :resource.Disease  :HepC
        :content-selector :resource.Duration :Monthly
        :content-selector :resource.Date :

        resource.Date+365>Current Date
        :action          :Action-id       :Release
        :purpose         :Purpose-id      :DrugEffect
</Attributes>
</Request REQ1>
```

Access Request

- The data store begins by matching the **Target** of the ***PolicySetInstitute1***.
- The ***access-subject*** **attribute Role matches**
- The ***resource*** **attribute Type matches**
- The ***action*** attribute **Action-Id** which is **Release matches**
- ***PolicySetInstitute1*** **applies** and Policy1 **must be checked**
- The **Target** of **Policy1 matches**,
  - so the **rule I1R1** is applied
- **I1R1** *has one condition* that checks whether the **access-subject** has the value **Yes** in the **attribute HIPAA Comp.**
- The condition of **I1R1** is **satisfied** and the effect is **Permit**.
  - Since this is the only rule, the net effect of **Policy 1** is **permit**.
  - So the effect of **PolicySetInstitute1** is **permit**

Access Request

- The data store then applies the policy set **PolicySetInstitute2** corresponding to **Institute 2**.
- The Target of **PolicySetInstitute2** is successfully matched and ***Policy2 is applied***.
- The request **fails** to **match Target** of Policy 2 and so this **policy** is **inapplicable**.
- Thus, **no policy is applicable** for this request from **Institute 2**.

Access Request

- **PolicySetPatient 1** belonging to **Patient 1** is then considered.
- The **Target** of ***PolicySet1*** **matches** the **attributes** of the ***researcher*** and **his access request** so the policy **PP1** is **applied**.
- **Target** of **PP1** **matches** and so rule **P1R1** is **applied.**
- The condition in **P1R1** is ***unsatisfied***
  - as the *researcher requests the drug-usage*, so the **rule is not applied**.
- Thus, **no policies** of ***Patient 1*** is able to **satisfy** the **request**.

Access Request

- ***PolicySetPatient2*** is then considered.
- The **Target** of ***PolicySetPatient2*** **matches** and so the **policy PP2** is **applied**.
- The **Target** of **PP2** **matches** and so the **rules P2R1** and **P2R2** are **checked**.
- The condition of **P2R2** **matches** and the effect is **Permit**.
- **P2R2** has an ***obligation*** that requires the response be rewritten to transform the dates.
- **P2R3** is the **default** rule and its effect is **Deny**.
- **P2R2** and **P2R3** provide **contradictory** effects,
  - but the ***rule-combining algorithm*** of **PP2** says that **permit-overrides**.
- ***Researcher*** will **have access** to the date **rewritten** records of Patient 3 that are found in ***Institute 1.***

The goal

- We want to model various policies imposed by the different stakeholders and ensure that they are enforced, keeping in mind that the policies by the various agencies or the preferences of the user may change any time.

Security requirements

- **Maintain patient privacy**
  - Prevent illegal disclosure and improper usage of private health data
- **Resistant to attribute collusion**
  - Multiple users should not be able to collude their attributes and gain access to health records
- **On demand user revocation**
  - When a particular attribute of a user is no longer valid, the user should not be able use the revoked attribute for accessing EHRs.
- **Selective disclosure of attributes**
  - Users should be able to gain access by disclosing only the minimum set of attributes satisfying the associated policy.
- **Flexible access**
  - Users from foreign domains should be able to gain access without registration in the local domain.

Sample System representation of the attribute-based authorization system