

# AUTORIZZAZIONE

Marco Console  
[console@dis.uniroma1.it](mailto:console@dis.uniroma1.it)

# NOTA PRELIMINARE

- Il seguente materiale didattico è basato su un insieme di lucidi preparato dal **Prof. Leonardo Querzoni**.
- Ringrazio il **Prof. Leonardo Querzoni** per avermi concesso di usare il materiale da lui prodotto.
- Tutti i diritti sull'utilizzo di questo materiale sono riservati ai rispettivi autori.
- Buona parte di questo set di slide è stato estratto da:
  - <http://www.slideshare.net/MarkNiebergall/access-control-models-controlling-resource-authorization>

# CONTROLLO DEGLI ACCESSI

Flusso tipico per il controllo degli accessi



# CONTROLLO DEGLI ACCESSI

Attori:

- **SUBJECT** : è l'elemento che richiede l'accesso ad una risorsa
  - Umano
  - Processo
- **RESOURCE**: è l'oggetto della richiesta
  - Dati
  - Periferiche
  - Servizi
  - ...

# AUTORIZZAZIONE

Il processo di autorizzazione garantisce l'accesso ad una risorsa da parte di un subject.

- La risposta può essere positiva (*Allow*) o negativa (*Deny*)
- Il processo di autorizzazione si svolge seguendo un modello di controllo degli accesso (*access control model*)
  - Definisce un framework per l'autorizzazione
  - Decide chi può fare cosa

# MODELLI

- Modelli per l'accesso
  - **DAC:** Discretionary Access Control
  - **MAC:** Mandatory Access Control
  - **RBAC:** Role Based Access Control
  - **ABAC:** Attribute Based Access Control

DAC

# DAC – ESEMPI

- Accesso ad una abitazione tramite un mazzo di chiavi.
  - Il proprietario (o un gestore) dà una copia delle chiavi a coloro che possono entrare
  - Solo chi ha le chiavi può entrare.



# DAC – ESEMPI

- **Unix file system.**
  - Il sistema riconosce tre categorie di utente per ogni file
    - User (proprietario)
    - Group
    - Others
  - Il sistema riconosce tre tipi di permessi
    - Lettura
    - Scrittura
    - Esecuzione
  - Il proprietario può concedere o negare i permessi a group e others.

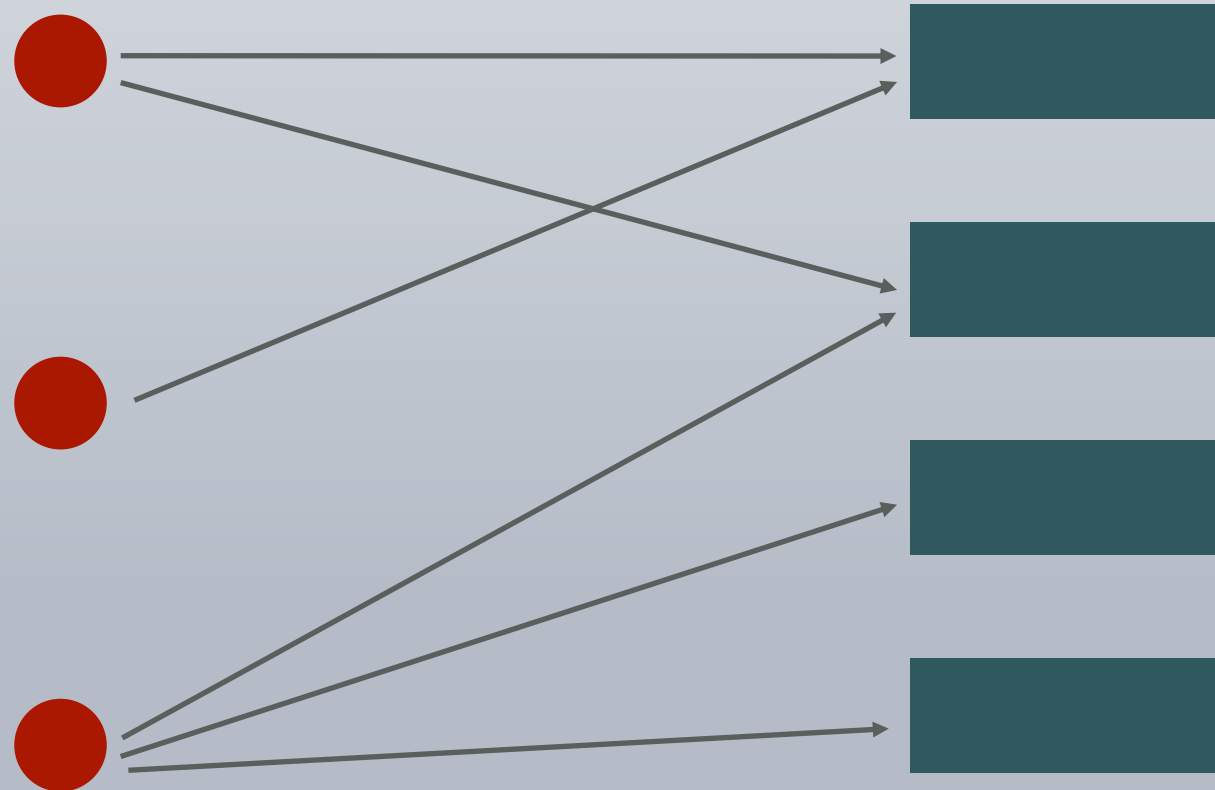
# MODELLI - DAC

## Discretionary Access Control (DAC)

- Il controllo è nelle mani del gestore della risorsa

Subjects




















Resources



# MODELLI - DAC

## Discretionary Access Control (DAC)

- È semplicemente implementabile tramite Access Control Lists (ACL)

	Page/Namespace	User/Group	Permissions <sup>1)</sup>
#1	 *	 @ALL	<input type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input checked="" type="radio"/> Create <input type="radio"/> Upload <input type="radio"/> Delete
#2	 *	 bigboss	<input type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input checked="" type="radio"/> Delete
#3	 devel: *	 @ALL	<input checked="" type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input type="radio"/> Delete
#4	 devel: *	 @devel	<input type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input checked="" type="radio"/> Upload <input type="radio"/> Delete
#5	 devel: *	 bigboss	<input type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input checked="" type="radio"/> Delete
#6	 devel: *	 @marketing	<input type="radio"/> None <input checked="" type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input type="radio"/> Delete
#7	 devel:funstuff	 bigboss	<input checked="" type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input type="radio"/> Delete
#8	 devel:marketing	 @marketing	<input type="radio"/> None <input type="radio"/> Read <input checked="" type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input type="radio"/> Delete
#9	 marketing: *	 @marketing	<input type="radio"/> None <input type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input checked="" type="radio"/> Upload <input type="radio"/> Delete
#10	 start	 @ALL	<input type="radio"/> None <input checked="" type="radio"/> Read <input type="radio"/> Edit <input type="radio"/> Create <input type="radio"/> Upload <input type="radio"/> Delete

- Quando un utente richiede una risorsa, il Sistema controlla la ACL e consente o nega l'azione dell'utente.

# MODELLI - DAC

## Discretionary Access Control (DAC)

- È semplicemente implementabile tramite Access Control Lists (ACL)

```
SELECT subject, resource
FROM acl
WHERE subject = 'Alice'
AND resource = 'Report'
LIMIT 1
```

# MODELLI - DAC

## Discretionary Access Control (DAC)

- PRO:

- Semplice implementazione
- Modello intuitivo
- Completamente decentralizzato (una ACL per ogni risorsa)

- CONTRO:

- Overhead enorme a runtime
  - Aumento incontrollato delle entries nelle ACL
- Controllo dell'accesso nelle mani del proprietario della risorsa
  - Le policy di governance non verranno eseguite in automatico dal sistema.

MAC

# MAC – ESEMPI

- Documenti confidenziali (classificazione UE)
  1. RESTRICTED.
  2. CONFIDENTIAL
  3. SECRET
  4. TOP SECRET
- Un individuo che può accedere a documenti di livello  $X$  può anche accedere a tutti i documenti di livello  $Y < X$

# MAC – ESEMPI

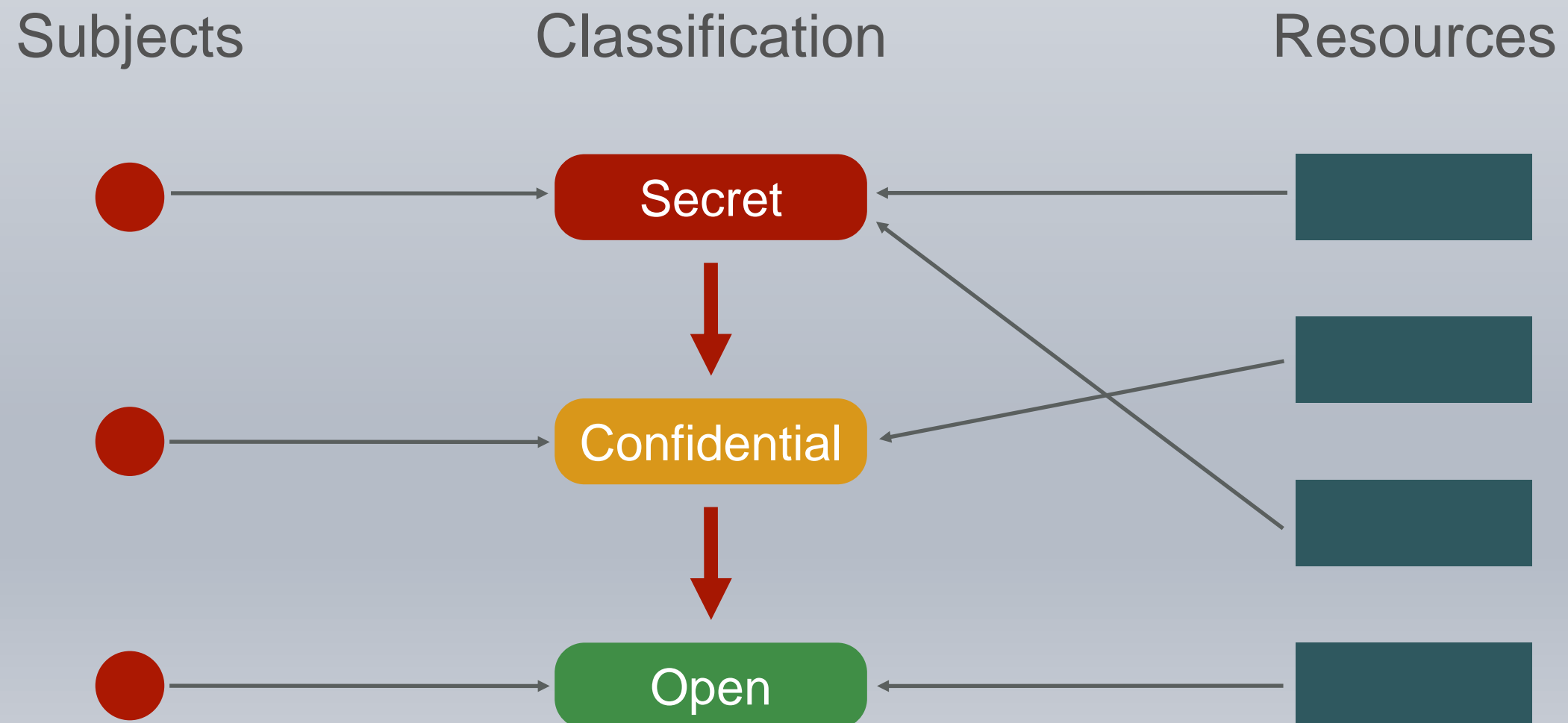
- Online gaming, per esempio un MORPG
  1. Area per personaggi di livelli 1-10
  2. Area per personaggi di livelli 11-20
  3. Area per personaggi di livelli 21-30
  4. Area per personaggi di livelli 31-40
  5. ....
- Un giocatore può accedere a tutte le aree del suo livello o inferiore.



# MODELLI - MAC

## Mandatory Access Control (MAC)

- Il controllo è basato su proprietà di riservatezza associate alle risorse e al livello di riservatezza a cui può accedere l'utente.
- I livelli di riservatezza sono organizzati in una gerarchia



# MODELLI - MAC

## Mandatory Access Control (MAC)

- Il controllo è basato su proprietà di accesso associate a subject e resources
- La riservatezza della risorsa è definita dal suo proprietario
- Il livello di accesso del subject è definito dal sistema

Subject security level	Object label		
	Secret	Confidential	Open
Secret	Allow	Allow	Allow
Confidential	Deny	Allow	Allow
Open	Deny	Deny	Allow

# MODELLI - MAC

## Mandatory Access Control (MAC)

- Implementazione semplice

```
SELECT s.security_level
FROM subject s
JOIN security_level sl_s
  ON sl_s.name = s.name
JOIN resource r
  ON r.resource = 'Report'
JOIN security_level sl_r
  ON sl_r.name = r.name
  AND sl_r.level <= sl_s.level
WHERE s.subject = 'Alice'
LIMIT 1
```

# MODELLI - MAC

## Mandatory Access Control (MAC)

- PRO:

- Sicurezza multilivello (con ereditarietà fra i livelli)
- L'accesso alle risorse è definito da due enti separati
  - Proprietari delle risorse e il sistema che assegna i livelli agli utenti.

- CONTRO:

- Scarsa flessibilità
  - Le classi delle risorse sono spesso hard-coded nel sistema
- Moderato overhead

RBAC

# RBAC – ESEMPI

- Utenti di tipo premium in uno specifico servizio.
  - Nel ruolo di utente normale posso acquistare prodotti.
  - Nel ruolo di utente premium ottengo maggiori servizi
    - Sconti
    - Spedizione gratuita

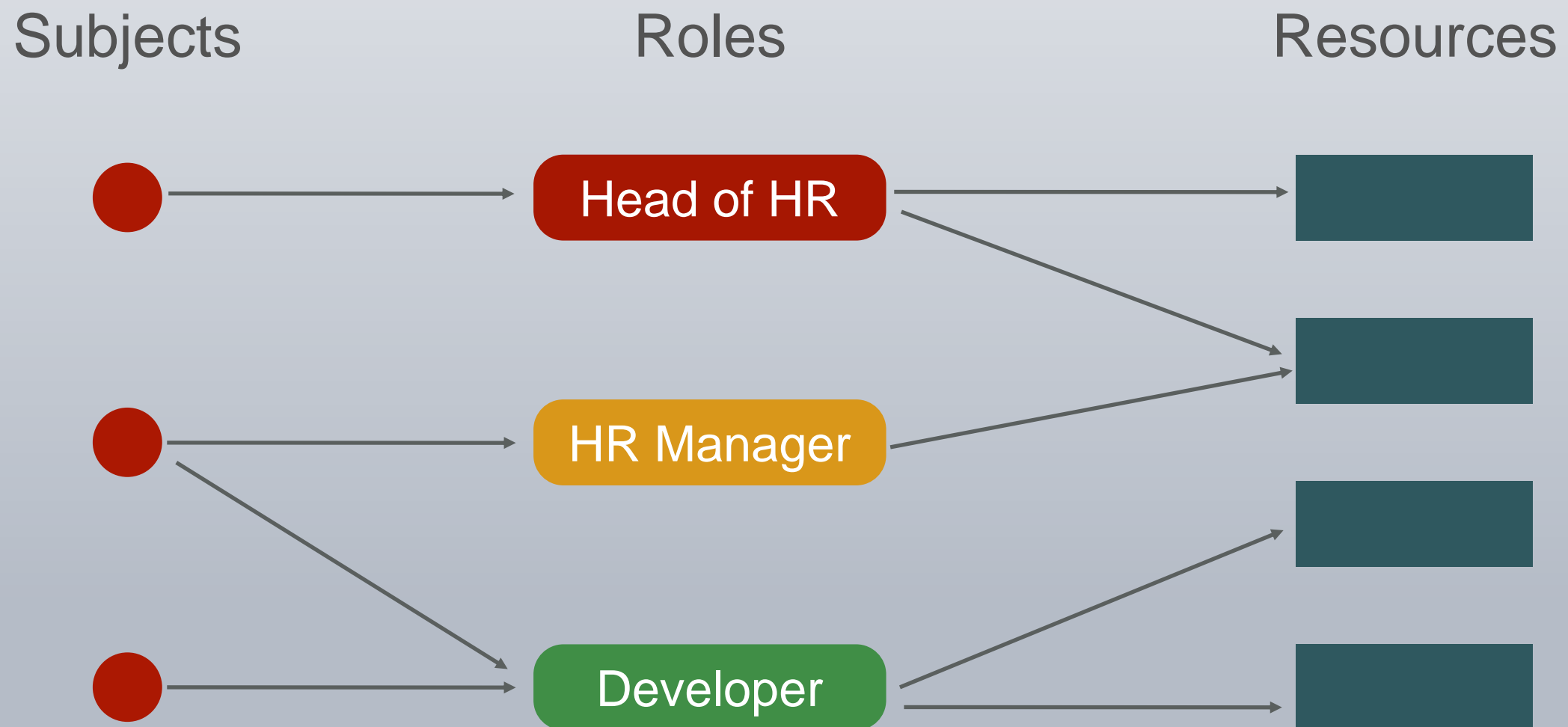
# RBAC – ESEMPI

- Utenti di un sistema operativo
  - L'utente semplice può accedere alle sue risorse e a specifiche funzionalità.
  - Nel ruolo di amministratore l'utente ottiene privilegi aggiuntivi

# MODELLI - RBAC

## Role-Based Access Control (RBAC)

- Il controllo è basato sulla definizione di ruoli e permessi sui ruoli.



- Più vicino al linguaggio naturale usato nelle policy aziendali.



# MODELLI - RBAC

## Quattro tipologie di modello

- **RBAC<sub>0</sub>**: subjects + roles + resources
- **RBAC<sub>1</sub>**: aggiunge la possibilità di definire un ordine gerarchico tra i ruoli
- **RBAC<sub>2</sub>**: aggiunge la possibilità di definire condizioni associate ai ruoli
- **RBAC<sub>3</sub>**: RBAC<sub>0</sub> + RBAC<sub>1</sub> + RBAC<sub>2</sub>

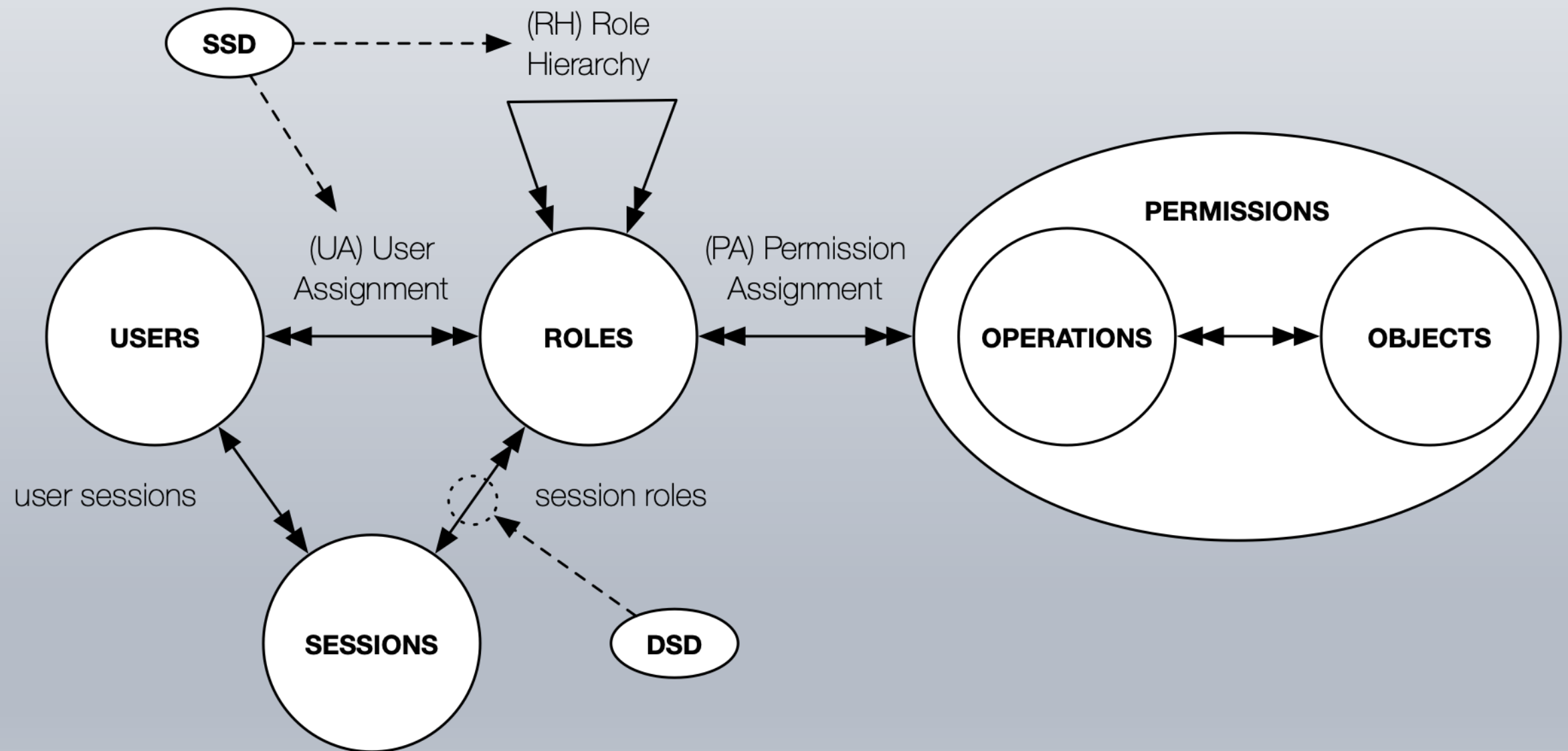
# MODELLI - RBAC

**RBAC<sub>3</sub>**: Prevede cinque concetti:

- **User** → Subject
- **Role** → Una funzione specifica che un utente può assumere all'interno di un'organizzazione. Es:
  - Studente, Direttore, Responsabile della sicurezza, Revisore dei conti
- **Object** → Resource
- **Permission/Operation** → Un permesso viene associato ad un ruolo per permettere l'esecuzione di una operazione. Le operazioni vengono svolte sugli objects.
- **Session** → Mappatura dei ruoli di un utente rispetto ad una sua specifica sessione d'uso

# MODELLI - RBAC

## Modello



# SEPARATION OF DUTIES -- EX

- Passi per pubblicare un articolo in una testata giornalistica
  1. Genera un nuovo articolo
  2. Controlla l'articolo e lo approva per la pubblicazione
  3. Posiziona l'articolo all'interno della testata
  4. Approva la pubblicazione online
- Possono essere eseguiti dalla stessa persona?

# SEPARATION OF DUTIES

- Approccio orientato a:
  - Ridurre il potere associato ad un singolo ruolo
  - Eliminare ed evitare possibili conflitti di interesse nell'uso di una risorsa
- Idea: per completare un task complesso sono necessarie operazioni i cui permessi sono associati a ruoli diversi
- Ex: Pubblicare un articolo su una testata giornalistica
  - Autore: genera un nuovo articolo
  - Revisore: controlla l'articolo e lo approva per la pubblicazione
  - Impaginatore: posiziona l'articolo all'interno della testata
  - Capo redattore: approva la pubblicazione online

# MODELLI - RBAC

RBAC implementa due tipologie di separation of duties

- **Static Separation of Duties**

- Vengono imposte condizioni sull'assegnazione statica dei ruoli agli utenti
- Queste condizioni lavorano a livello amministrativo
- Es: un utente non può scrivere un articolo e pubblicarlo online
- Es: un solo utente può essere capo-redattore

- **Dynamic Separation of Duties**

- Vengono imposte condizioni sulla attivazione di diversi ruoli da parte dell'utente
- Queste condizioni lavorano a runtime sulla sessione utente
- Es: Se ho richiesto un pagamento in una sessione non posso approvarlo nella stessa sessione
- In realtà DSD risponde più ad un principio di privilegio minimo

# MODELLI - RBAC

## Principle of Least Privilege

- È un principio che deve essere applicato ogni volta che il modello di accesso viene istanziato
- **Principio:** ogni utente deve interagire con il sistema utilizzando i privilegi minimi necessari a completare il proprio task
- Es: se devo usare una semplice applicazione sul mio computer (Internet browsing) non ho bisogno avviarla con i privilegi di amministrazione
- Vantaggi
  - Maggiore stabilità dei sistemi (meno interferenze possibili tra utenti)
  - Maggiore sicurezza (un utenza violata può causare danni limitati)
  - Maggiore semplicità nel controllo

# MODELLI - RBAC

Tipica implementazione di RBAC<sub>0</sub>: Access Control Matrix

Users	Roles			
	R <sub>1</sub>	R <sub>2</sub>	...	R <sub>m</sub>
	U <sub>1</sub>	X		
	U <sub>2</sub>	X	X	
	...			
	U <sub>n</sub>	X		X

		Objects			
		O <sub>1</sub>	O <sub>2</sub>	...	O <sub>p</sub>
Roles	R <sub>1</sub>	owner	control		owner
	R <sub>2</sub>		read		delete
	...				
	R <sub>m</sub>	read			execute



# MODELLI - RBAC

## Role-Based Access Control (RBAC)

- PRO:
  - Flessibile
  - Gestibile e scalabile
  - Sicuro
  - Mappa chiaramente il contesto organizzativo
  
- CONTRO:
  - Esplosione dei ruoli
  - Se non sono chiaramente definite delle policy di sicurezza è difficilmente applicabile

# DOMANDA

- Qual è la differenza fra i ruoli di RBAC e i livelli di riservatezza nel modello MAC?
- Possiamo implementare MAC e DAC con RBAC?

# DOMANDA

- Qual è la differenza fra i ruoli di RBAC e i livelli di riservatezza nel modello MAC?
  - I livelli di riservatezza sono una proprietà delle risorse. Gli utenti sono poi associati al livello massimo di riservatezza a cui possono accedere.
  - I ruoli di RBAC sono una proprietà degli utenti. I permessi (operazione + risorsa) sono assegnate ai singoli ruoli.
- Possiamo implementare MAC e DAC con RBAC?
  - Possiamo implementare MAC utilizzando i ruoli diversi per i livelli di riservatezza
  - Possiamo implementare DAC associando un singolo ruolo ad ogni utente.

ABAC

# ABAC – ESEMPI

- Accesso all'area di imbarco di un aeroporto
  - Abbiamo un biglietto per un volo nelle prossime ore.
  - Non trasportiamo nessun oggetto pericoloso.
  - Il bagaglio ha dimensione e peso corretti
  - ....

# ABAC – ESEMPI

- Accesso al codice dei progetti di una start-up
  - Alice lavora ai progetti A e B
    - Come sviluppatore in A
    - Come tester in B
  - Bob lavora ai progetti B e C
    - Come sviluppatore in B
    - Come tester in C
  - Charlene lavora ai progetti A, B, C
    - Come sviluppatore in C
    - Come tester in A
    - Come user in B
- Quali ruoli? Quanti ruoli?

# MODELLI - ABAC

Nella sua essenza, Access Control si riduce alle seguenti domande

- Who ?
  - Chi sta richiedendo l'accesso?
- What ?
  - Cosa vuole fare?
- When ?
  - Quando si vuole farlo?
- Where ?
  - Su quale risorsa?
- Why ?
  - In base a quale criteri?
- How ?
  - Come verrà effettuato l'accesso?

Decision (PERMIT/DENY)

# MODELLI - ABAC

## Attribute-Based Access Control

Un metodo di access control in cui le richieste dei **soggetti** di eseguire **operazioni** sugli **oggetti** sono permesse o negate in base agli **attributi** assegnati ai soggetti, agli oggetti, alle condizioni del sistema e a un insieme di policies specificate su questi attributi.



# ABAC -- CARATTERISTICHE

- Il controllo d'accesso è *esternalizzato* rispetto alla business logic.
  - Il controllo può essere definito al di fuori della risorsa.
- Le regole che definiscono il controllo di accesso sono gestite in modo centralizzato.
- Il controllo di accesso è svolto in modo flessibile
  - Può essere applicato a servizi, dati, etc.
- Le decisioni sull'accesso vengono prese dinamicamente a *runtime*.
  - A seconda del contesto in cui avvengono

# MODELLI - ABAC

Esempio: corporate policy

*Solo i dipendenti dell'agenzia di Milano*

*possono vedere*

*i saldi dei conti-corrente bancari*

*dei clienti di Milano*

# MODELLI - ABAC

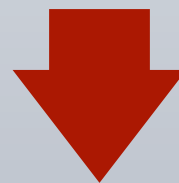
Esempio: corporate policy

*Solo i dipendenti dell'agenzia di Milano*

*possono vedere*

*i saldi dei conti-corrente bancari*

*dei clienti di Milano*



Ruolo: dipendente	Risorsa: saldo
Posizione: Milano	Tipo risorsa: conto corrente bancario
Operazione: lettura	Posizione risorsa: Milano

# MODELLI - ABAC

Esempio: corporate policy

*Solo i dipendenti dell'agenzia di Milano  
possono vedere  
i saldi dei conti-corrente bancari  
dei clienti di Milano*



Un utente con ruolo == dipendente  
può eseguire l'operazione == SELECT  
sulla colonna = SALDI  
della tabella = CONTI\_CORRENTI  
se CONTI\_CORRENTI.posizione ==  
utente.posizione

# MODELLI - ABAC

## Attribute-Based Access Control (ABAC)

- PRO:

- Semplifica la gestione delle autorizzazioni perché più vicino al linguaggio naturale.
- Riduce il rischio di accessi non autorizzati per la sua semplicità.
- La gestione centralizzata facilita notevolmente l'auditing

- CONTRO:

- Nessuna nota, probabilmente per il grado di maturità.

*By 2020, 70% of all businesses will use attribute based access control (ABAC) as the dominant mechanism to protect critical assets, up from 5% today (Gartner, 2013)*

REVIEW

# REVIEW

- **DAC:** Discretionary Access Control
  - Facile da implementare tramite ACL, Non automatizzato, Overhead
- **MAC:** Mandatory Access Control
  - Facile da implementare, Overhead Moderato, Poco Flessibile
- **RBAC:** Role Based Access Control
  - Flessibile e poco overhead in generale. Il numero dei ruoli potrebbe esplodere.
- **ABAC:** Attribute Based Access Control
  - Flessibile e con overhead limitato. Ad oggi il sistema più avanzato.

XACML



# XACML

## XACML: eXtensible Access Control Markup Language

- È uno standard per l'espressione di regole di gestione dell'accesso
- È basato su ABAC
- Permette di esprimere le policy con un linguaggio naturale

### Definisce:

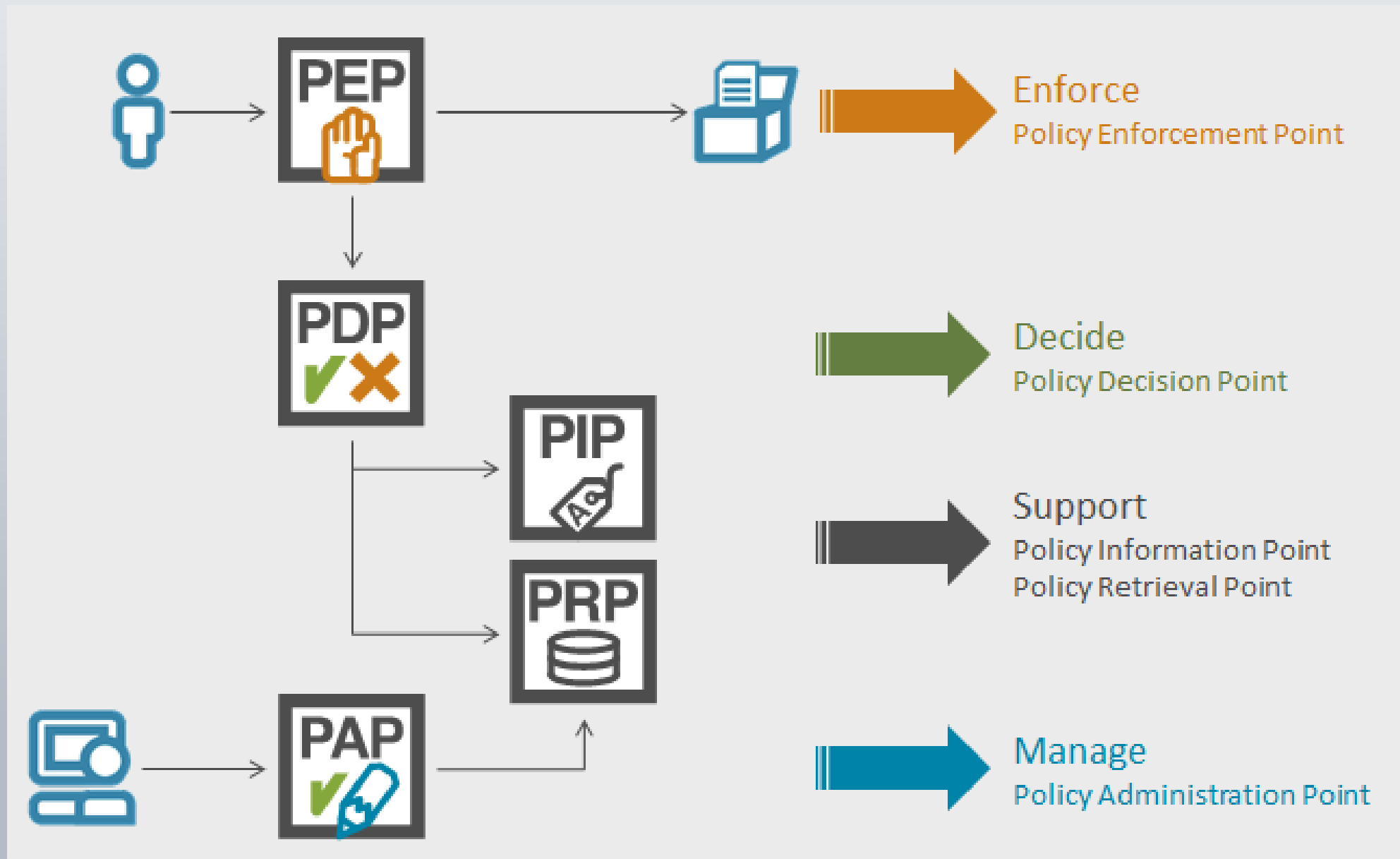
- Un linguaggio per esprimere policy
- Un protocollo request/response
- Una architettura per l'implementazione

# XACML – FRAMEWORK

- Concetti base
  - **Subject:** l'entità che richiede l'accesso
  - **Object:** la risorsa alla quale il subject richiede di accedere
- Elementi logici del framework.
  - Policy Enforcement Point
    - Intercetta le richieste dei Subject e agisce secondo la decisione presa dal PDP.
  - Policy Decision Point
    - Valuta le richieste di accesso e restituisce una decisione (concede\nega l'accesso).
  - Policy Information Point
    - Dove gli attributi dei Subject sono conservati
  - Policy Retrieval Point:
    - Dove le policy di accesso sono salvate.
  - Policy Administration Point:
    - Dove le policy di accesso vengono gestite (dagli amministratori)

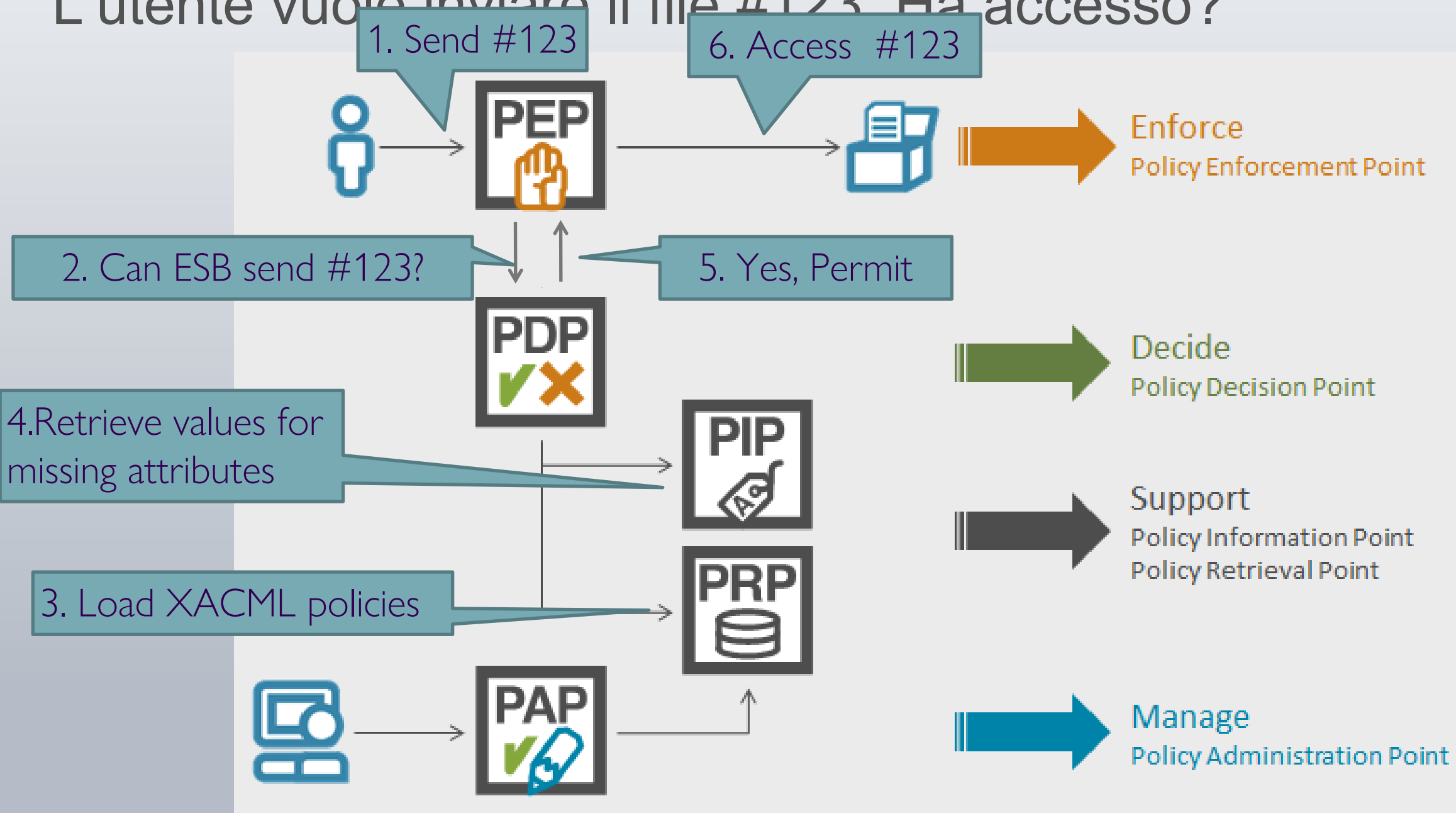
# XACML

## Architettura:



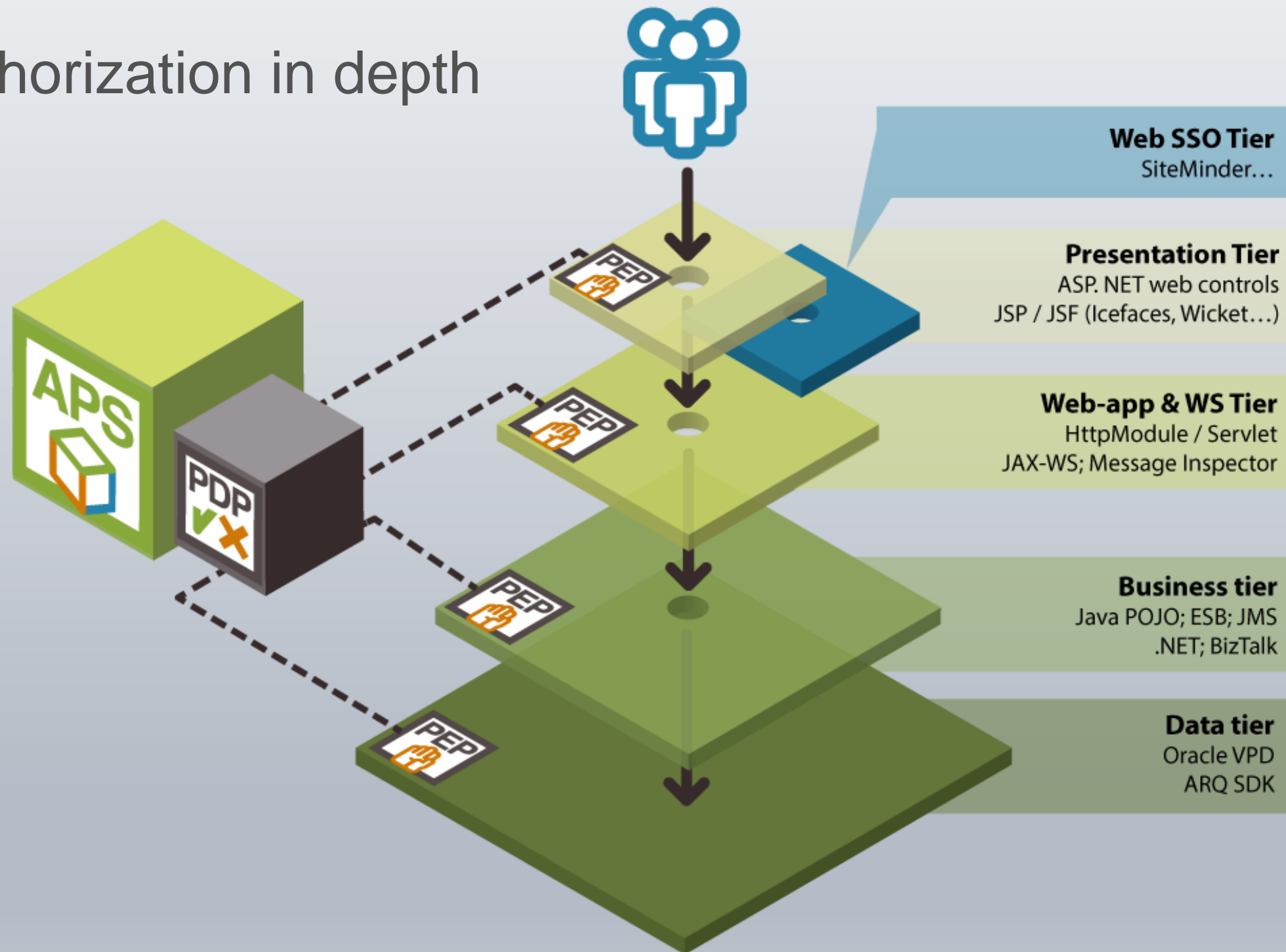
# XACML

L'utente vuole inviare il file #123. Ha accesso?



# XACML

## Authorization in depth



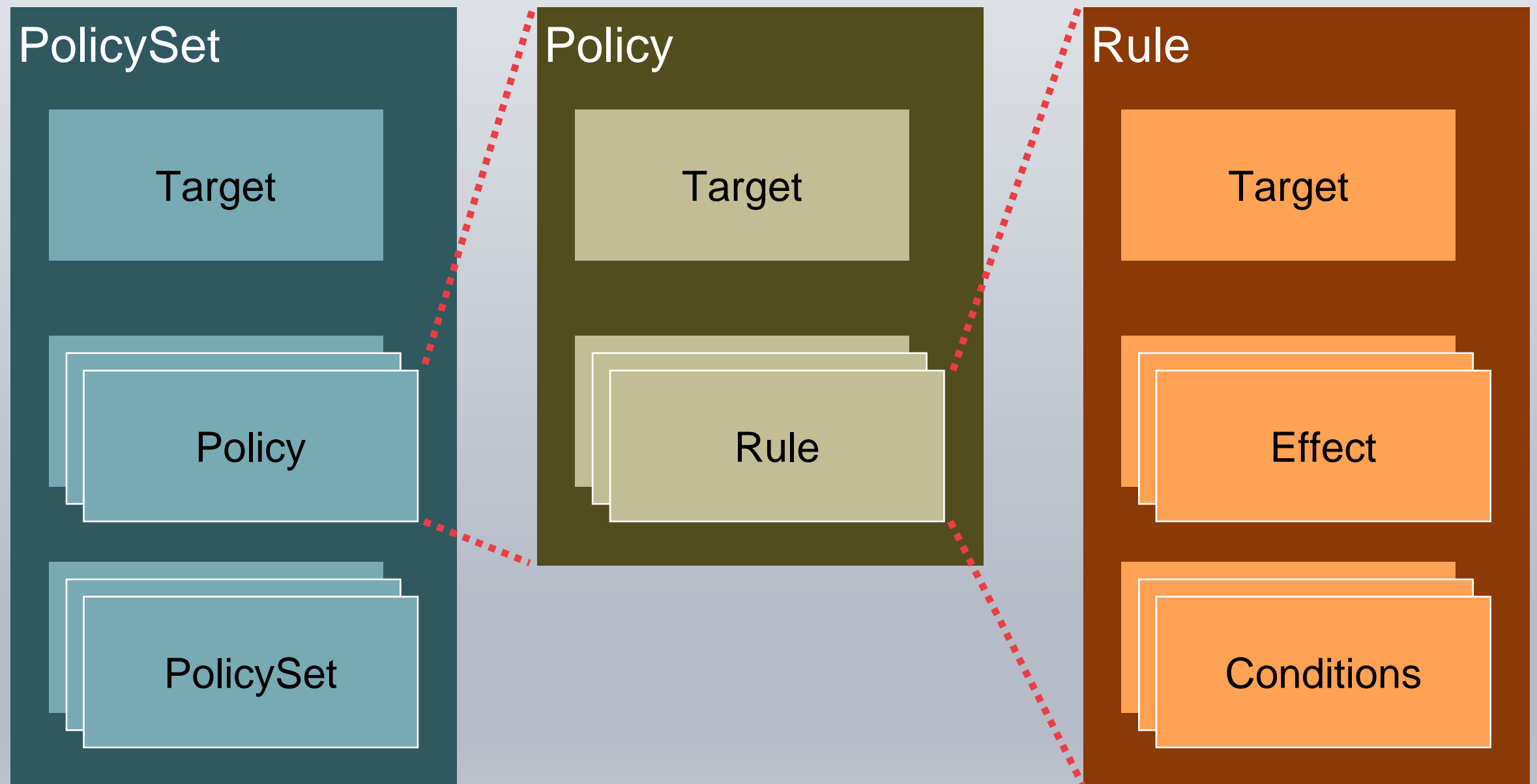
# XACML – RIECHIESTE

```
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>admin</AttributeValue>
    </Attribute>
    <Attribute AttributeId="department" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>sysadmin</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>http://localhost:8280/services/echo/echoString</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue>
    </Attribute>
  </Action>
  <Environment/>
</Request>
```

# XACML – POLICY LANGUAGE

- A *<Policy>* contains a set of *<Rule>* elements, and a rule-combining algorithm
- A *<Rule>* contains:
  - a target (the set of subjects, resources, actions and environments to which it applies)
  - an effect ("Permit" and "Deny")
  - a condition (refines the applicability of the rule beyond the predicates implied by its target)

# XACML – POLICY LANGUAGE





# XACML – POLICY LANGUAGE

```
<PolicySet PolicySetId="PPS:HR:role" RuleCombiningAlgId=
  <Policy PolicyId="Permissions.for:HR:role" RuleCombiningAlgId=
    "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
    <Description>Anybody in HR can view any record</Description>
    <Rule RuleId="Permission.to:view:HR:record" Effect="Permit">
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId=
              "urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType=
                "http://www.w3.org/2001/XMLSchema#string">
                hr-record
              </AttributeValue>
              <ResourceAttributeDesignator>
                urn:emc:edn:samples:xamcl:resource:resource-type
              </ResourceAttributeDesignator>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId=
              "urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType=
```