University of Science and Technology
Faculty of Computer Science and Information Technology
Information Security Master Program
Course Title: Cloud Computing Security

Lecture (4): Identity and Access Management

Reference: Cloud Computing Security and Privacy, by Tim Mather, Subra and Lattif, -------Chapter (5)

Instructor: Prof. Noureldien Abdelrahman

THIS lecture presents the practice of identity and access management (IAM) and IAM features that aid in Authentication, Authorization, and Auditing (AAA) of users accessing cloud services.

4.1 Trust Boundaries and IAM

In a typical organization where applications are deployed within the organization's perimeter the "trust boundary" is mostly static and is monitored and controlled by the IT department. In that traditional model, the trust boundary encompasses the network, systems, and applications hosted in a private data center managed by the IT department. And access to the network, systems, and applications is secured via network security controls including firewalls, virtual private networks (VPNs), intrusion detection systems (IDSs), intrusion prevention systems (IPSs), and multifactor authentication.

With the adoption of cloud services, the organization's trust boundary will become dynamic and will move beyond the control of IT. With cloud computing, the network, system, and application boundary of an organization will extend into the service provider domain. This loss of control continues to challenge the established trusted governance and control model, and, if not managed properly, will impede cloud service adoption within an organization.

4.2 Why IAM?

Traditionally, organizations invest in IAM practices to:

1- Improve operational efficiency

Properly architected IAM technology and processes can improve efficiency by automating user on-boarding and other repetitive tasks (e.g., self-service for users requesting password resets that otherwise will require the intervention of system administrators using a help desk ticketing system).

2- Regulatory security compliance management

Regulatory security compliance means to protect systems, applications, and information from internal and external threats and to comply with various regulatory, privacy, and data protection requirements. **Almost, organizations implement an "IT general and application-level controls" framework derived from industry standard frameworks such as ISO 27002 and Information Technology Infrastructure Library (ITIL).**

Cloud Cases Sported by IAM

Cloud Service Providers require IAM support from many cases that include:

• Employees and on-site contractors of an organization accessing a SaaS service using identities and credentials.

• IT administrators accessing the CSP management console to provision resources and access for users using a corporate identity.

• Developers creating accounts for partner users in a PaaS platform.

• End users accessing storage service in the cloud (e.g., Amazon S3) and sharing files and objects.

• An application residing in a cloud service provider (e.g., Amazon EC2) accessing storage from another cloud service.

4.3 IAM Definitions

To start, we'll present the basic concepts and definitions of IAM functions for any service:

Authentication

Authentication is the process of verifying the identity of a user or system . Authentication usually connotes a more robust form of identification.

Authorization

Authorization is the process of determining the privileges the user or system is entitled to once the identity is established. In the context of digital services, authorization usually follows the authentication step and is used to determine whether the user or service has the necessary privileges to perform certain operations—in other words, authorization is the process of enforcing policies.

Auditing

Auditing is the process of review and examination of authentication, authorization records, and activities to:

- determine the adequacy of IAM system controls,

- verify compliance with established security policies and procedures (e.g., separation of duties)

- detect breaches in security services (e.g., privilege escalation), and

- recommend any changes that are indicated for countermeasures.

4.4 IAM Architecture and Practice

IAM is not a monolithic solution that can be easily deployed to gain capabilities immediately. It **is as much an aspect of architecture** (see Figure 5-1) as **it is a collection of technology components, processes, and standard practices.**

Standard enterprise IAM architecture encompasses several layers of technology, services, and processes. At the core of the *deployment architecture is a directory service (such as LDAP or Active Directory) that acts as a repository for the identity, credential, and user attributes of the organization's user pool*. The directory interacts with IAM technology components such as authentication, user management, provisioning, and identity services that support the standard IAM practice and processes within the organization.

4.4.1 IAM Processes

The IAM processes that support the business can be broadly categorized as follows:

User management

These are activities for the effective governance and management of identity life cycles.

Authentication management

These are activities for the effective governance and management of the process for determining that an entity is who or what it claims to be.

Authorization management

These are activities for the effective governance and management of the process for determining entitlement rights that decide what resources an entity is permitted to access in accordance with the organization's policies.

Access management

These are activities that **enforce policies** ***for access control*** in response to a request from an entity (user, services) wanting to access an IT resource within the organization.

Data management and provisioning

These are activities that propagate identity and authorization data to an IT resource via automated or manual processes.

Monitoring and auditing

Monitoring, auditing, and reporting compliance by users regarding access to resources within the organization based on the defined policies.

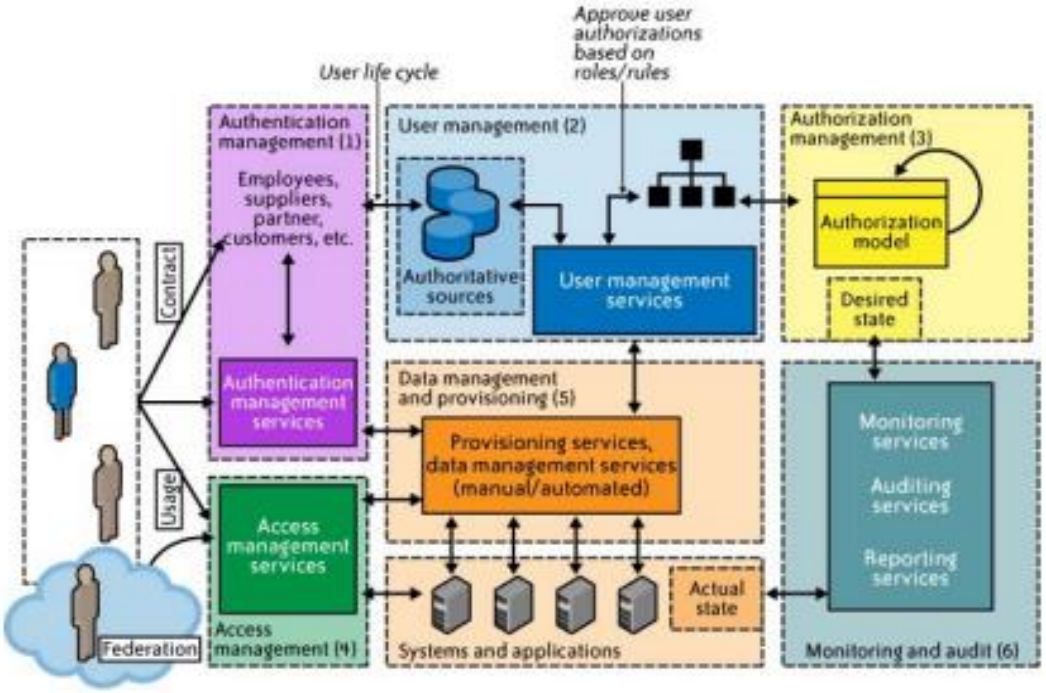


FIGURE 5-1. Enterprise IAM functional architecture

4.4.2 Operational Activities Supported by IAM

IAM processes support the following operational activities:

1- Provisioning

This is the process of on-boarding users to systems and applications. These processes provide users with necessary access to data and technology resources. Provisioning can be thought of as a combination of the duties of the human resources and IT departments, where users are given access to data repositories or systems, applications, and databases based on a unique user identity. ***Deprovisioning*** works in the opposite manner, resulting in the deletion or deactivation of an identity or of privileges assigned to the user identity.

*2- Credential and **attribute** management*

These processes are designed to manage the life cycle of credentials and user attributes— ***create issue, manage, revoke***—to minimize the business risk **associated with identity impersonation** and inappropriate account use.

Credentials are usually bound to an individual and are verified during the authentication process. The processes include provisioning of attributes, static (e.g., standard text password) and dynamic (e.g., one-time password) credentials that comply with a password standard (e.g., passwords resistant to dictionary attacks), handling password expiration, and encryption management of credentials during transit and at rest, and access policies of user attributes (privacy and handling of attributes for various regulatory reasons).

3- Entitlement management

Entitlements are also referred to **as *authorization policies*.** The processes in this domain address the ***provisioning and deprovisioning privileges*** needed for the user to access resources including systems, applications, and databases.

Proper entitlement management ensures that users are assigned only the required privileges (least privileges) that match with their job functions. Entitlement management can be used to strengthen the security of web services, web applications, legacy applications, documents and files, and physical security systems.

4- Compliance management

This process implies **that access rights and privileges are monitored and tracked to ensure the security of an enterprise's resources**. The process also helps auditors verify compliance to various internal access control policies, and standards that include practices such as segregation of duties, access monitoring, periodic auditing, and reporting. An example is a user certification process that allows application owners to certify that only authorized users have the privileges necessary to access business-sensitive information.

5- Identity Federation management

Federation is the process of managing the trust relationships established beyond the internal network boundaries or administrative domain boundaries among distinct organizations. A federation is an association of organizations that come together to exchange information about their users and resources to enable collaborations and transactions (e.g., sharing user information with the organizations' benefits systems managed by a third-party provider).

6- Centralization of authentication and authorization

A central authentication and authorization **infrastructure alleviates the need for** application developers to build custom authentication and authorization features into their applications. Furthermore, it promotes a loose coupling architecture where applications become agnostic to the authentication methods and policies.

Figure 5-2 illustrates the identity life cycle management phases.

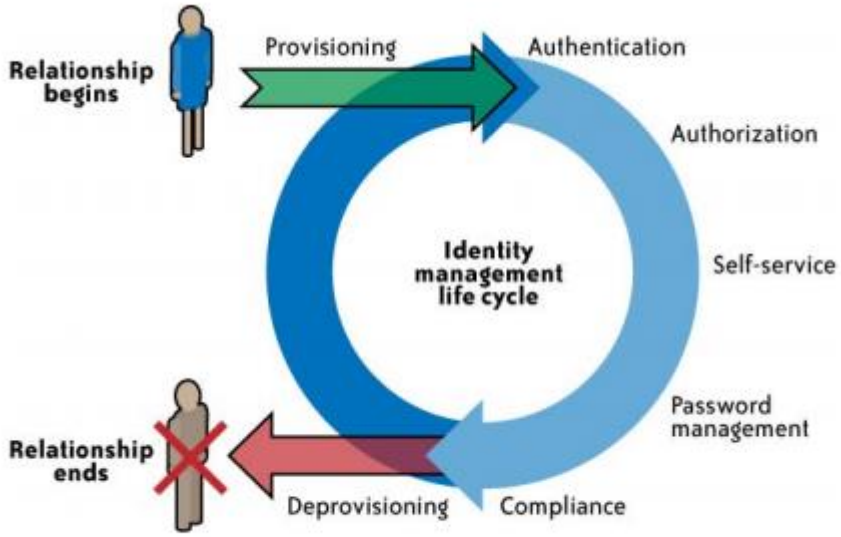


FIGURE 5-2. Identity life cycle

4.5 IAM and the Cloud

As a first step, organizations planning for cloud services must plan for basic user management functions such as user account provisioning and ongoing user account management, including timely deprovisioning of users when they no longer need access to the cloud service.

Enterprises that haven't established procedures for identity and access management can use cloud-based solutions from a number of vendors that offer identity management services (examples include Symplified, Ping Identity, Conformity, and TriCipher).

4.5.1 Relevant IAM Standards and Protocols for Cloud Services

In this section, we will discuss the relevant IAM standards that act as catalysts for organizations adopting cloud services.

The following IAM standards and specifications will help organizations implement effective and efficient user access management practices **and processes in the cloud**. These sections are ordered by four major challenges in user and access management faced by cloud users:

1. How can I avoid **duplication of identity, attributes, and credentials and provide a single sign-on (SSO) user experience for my users?**

The answer is: Use SAML.

2. How can I automatically provision user accounts with **cloud services and automate the process** of provisoning and deprovisioning?

The answer is: Use SPML.

3. How can I provision user accounts with appropriate privileges and manage entitlements for my users? The answer is: Use XACML.

4. How can I authorize cloud service X to access my data in cloud service Y without disclosing credentials? The answer is: Use OAuth.

4.5.1.1 What is SAML?

SAML, Security Assertion Markup Language (**SAML**, pronounced *sam-el*) is an XML-based, open-standard data format for exchanging authentication and authorization data between parties, in particular, between an **identity provider** (IdP) and a **service provider** (SP).

In computing, an **Identity provider** (IdP), also known as **Identity Assertion Provider**, can:

1. provide identifiers for users looking to interact with a system
2. assert to such a system that such an identifier presented by a user is known to the provider
3. possibly provide other information about the user that is known to the provider

A **service provider** (**SP**) provides organizations with consulting, legal, real estate, communications, storage, processing. Although a service provider can be an organizational sub-unit, it is usually a third party or outsourced supplier

SAML is a product of the **OASIS** Security Services Technical Committee. SAML dates from 2001; the most recent major update of SAML was published in 2005, but protocol enhancements have steadily been added through additional, optional standards.

The **Organization for the Advancement of Structured Information Standards** (**OASIS**) is a global nonprofit consortium that works on the development, convergence, and adoption of standards for security, Internet of Things, energy, content technologies, emergency management, and other areas.

Principles

The SAML specification defines three roles: the principal (typically a user), the Identity provider (IdP), and the service provider (SP).

In the use case addressed by SAML, the principal requests a service from the service provider. The service provider requests and obtains an identity assertion from the identity provider. On the basis of this assertion, the service provider can make an access control decision – in other words it can decide whether to perform some service for the connected principal.

Before delivering the identity assertion to the SP, the IdP may request some information from the principal – such as a user name and password – in order to authenticate the principal.

SAML specifies the **assertions between the** three parties: in particular, the messages that assert identity that are passed from the IdP to the SP.

In SAML, one identity provider may provide SAML assertions to many service providers. Similarly, one SP may rely on and trust assertions from many independent IdPs.

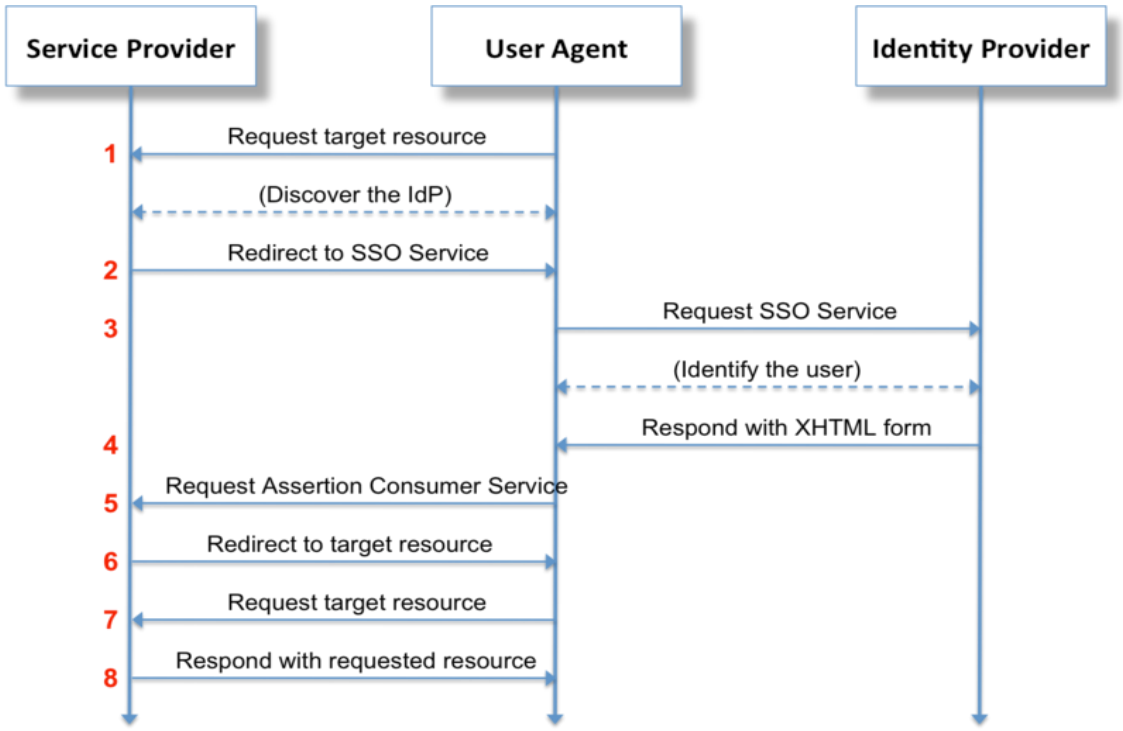
SAML does not specify the method of authentication at the identity provider; it may use a username and password, or other form of authentication, including multi-factor authentication. A directory service such as LDAP, RADIUS, or Active Directory that allows users to log in with a user name and password is a typical source of authentication tokens at an identity provider.

Use

The primary SAML use case is called *Web Browser Single Sign-On (SSO)*, where a user using a *user agent* (usually a web browser) requests a web resource protected by a SAML *service provider*.

> **Single sign-on** (**SSO**) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to a connected system or systems without using different usernames or passwords, or in some configurations seamlessly sign on at each system. **This is typically accomplished using the Lightweight Directory Access Protocol (LDAP) and stored LDAP databases on (directory) servers**. A simple version of single sign-on can be achieved over IP networks using cookies but only if the sites share a common DNS parent domain

The service provider, who is wishing to know the identity of the requesting user, issues an authentication request to a SAML *identity provider* through the user agent. The resulting protocol flow is depicted in the following diagram.



1. Request the target resource at the SP

   The principal (via an HTTP user agent) requests a target resource at the service provider:
   `https://sp.example.com/myresource`

The service provider performs a security check on behalf of the target resource. If a valid security context at the service provider already exists, skip steps 2–7.

2. Redirect to the SSO Service at the IdP

   The service provider determines the user's preferred identity provider and redirects the user agent to the SSO Service at the identity provider:
   `https://idp.example.org/SAML2/SSO/`**`Redirect?SAMLRequest=request`**
   The value of the `SAMLRequest` parameter is the encoding of a deflated `<samlp:AuthnRequest>` element.

3. Request the SSO Service at the IdP

   The user agent issues a GET request to the SSO service at the identity provider where the value of **the `SAMLRequest` parameter** is taken from the URL query string at step 2. The SSO service processes the `AuthnRequest` and performs a security check. If the user does not have a valid security context, the identity provider identifies the user (details omitted).

4. Respond with an XHTML form

   The SSO service validates the request and responds with a document containing an XHTML form:

```
  <form method="post" action="https://sp.example.com/SAML2/SSO/POST"
...>
    <input type="hidden" name="SAMLResponse" value="response" />
    ...
    <input type="submit" value="Submit" />
  </form>
```

   The value of the `SAMLResponse` parameter is the encoding of **a `<samlp:Response>`** element.

5. Request the Assertion Consumer Service at the SP

   The user agent issues a POST request to the assertion consumer service at the service provider. The value of the `SAMLResponse` parameter is taken from the XHTML form at step 4.

6. Redirect to the target resource

   The assertion consumer service processes the response, creates a security context at the service provider and redirects the user agent to the target resource.

7. Request the target resource at the SP again

   The user agent requests the target resource at the service provider (again):
   `https://sp.example.com/myresource`

8. Respond with requested resource

   Since a security context exists, the service provider returns the resource to the user agent.

In the example flow above, all depicted exchanges are *front-channel exchanges*, that is, an HTTP user agent (browser) communicates with a SAML entity at each step. In particular, there are no ***back-channel exchanges*** **or direct communications between the service provider and the identity provider**. Front-channel exchanges lead to simple protocol flows where all messages are passed *by value* using a simple HTTP binding (GET or POST). Indeed, the flow outlined in the previous section is sometimes called **the *Lightweight Web Browser SSO Profile***.

Alternatively, for increased security or privacy, messages may be passed *by reference*. For example, an identity provider may supply a reference to a SAML assertion (called an *artifact*)

instead of transmitting the assertion directly through the user agent. Subsequently, the service provider requests the actual assertion via a back channel. Such a back-channel exchange is specified as a **SOAP** message exchange (SAML over SOAP over HTTP). In general, any SAML exchange over a secure back channel is conducted as a SOAP message exchange.

SOAP (originally **Simple Object Access Protocol**) is a protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to induce extensibility, neutrality and independence. It uses XML Information Set for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission. SOAP allows processes running on disparate operating systems (such as Windows and Linux) to communicate using Extensible Markup Language (XML).

4.4.1.2 Service Provisioning Markup Language (SPML)

SPML is an XML-based framework being developed by OASIS for exchanging user, resource, and service **provisioning information among cooperating** organizations. SPML is an emerging standard that can help organizations automate provisioning of user identities for cloud.

When SPML is available, organizations should use it to **provision user accounts and profiles with the cloud service.**

If SPML is supported, software-as-a-service (SaaS) providers can enable **"just-in-time provisioning" to create accounts for new users in real time** (as opposed to preregistering users). In this case, the CSP extracts attributes from the SAML token of a new user, creates an SPML message on the fly, **and hands the request to a provisioning service which in turn adds the user identity to the cloud user database**.

OASIS uses the following definition of "provisioning": *Provisioning is the automation of all the steps required to manage (setup, amend and revoke) user or system access entitlements or data relative to electronically* published services.

1. Goal of SPML

The goal of SPML is to allow organizations to securely and quickly set up user interfaces for Web services and applications, by letting enterprise platforms such as Web portals, application servers, and service centers generate provisioning requests within and across organizations.

This can lead to automation of user or system access and entitlement rights to electronic services across diverse IT infrastructures.

SPML aims to achieve a couple of things:

• **Automated IT provisioning tasks**: By standardizing the job of provisioning and making it easier to encapsulate the security and auditing requirements of provisioning systems.

• Interoperability between different provisioning systems: Different provisioning systems can now expose standard SPML interfaces to each other and interoperate with each other.

2. Components

A provisioning system is made up of three essential components: The Requesting Authority (RA), the Provisioning Service Point (PSP), and the Provisioning Service Target (PST).

- **Requesting Authority (RA):**

This is the client in the SPML scheme. It creates well-**formed SPML massages and sends them as requests to the SPML service point.** These requests describe an operation to be performed at specific provisioning service points (PSP).
For an RA to issue a request to a PSP, **a trust relationship must exist between** the RA and the SPML service point (PSP). *Even an SPML service point can act as an RA when* it issues an SPML request to another service point.

- **Provisioning Service Point (PSP):**

This is the component that listens to the request from the RA, processes it, and returns a response to the RA. Any component that listens and processes well-formed SPML documents is called a Provisioning Service Point.

- **Provisioning Service Target (PST):**

This is **the actual resource on which the action is taken**. For example, it could be an LDAP directory that stores all of an organization's user accounts, or it could be a ticketing system that is used to issue access tickets.
So as you can see, the architecture is essentially a client (RA), a server (PSP), and resources (PSTs) that SPML manages.

The figure below illustrates an SPML use case in which an HR system (RA) is requesting a provisioning system in the cloud with the SPML request. In the figure, HR System of Record (requesting authority) is an SPML web services client interacting with the SPML provisioning service provider (PSP) at the cloud service provider, which is responsible for provisioning user accounts on the cloud services (provisioning service target).

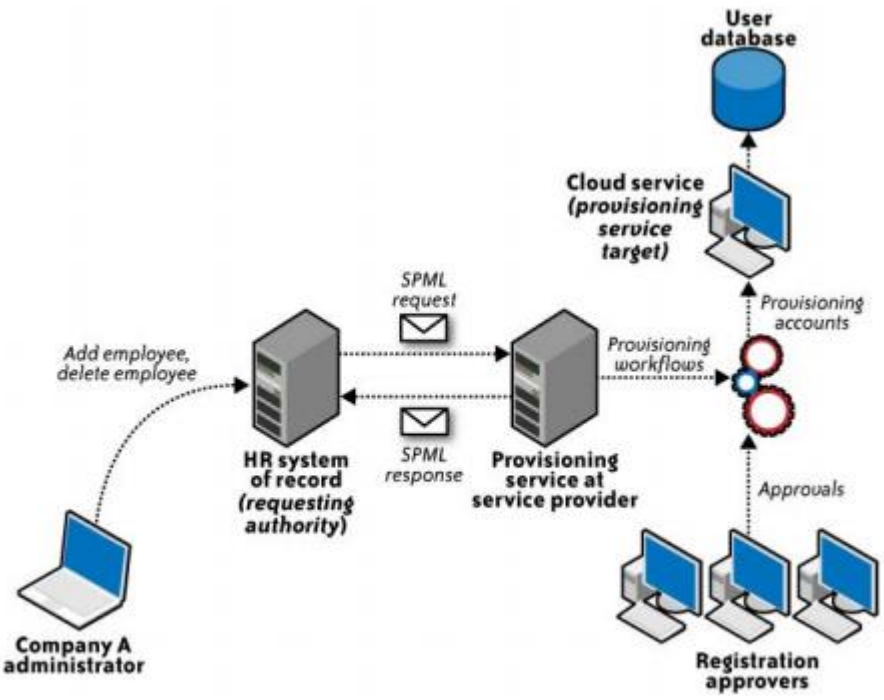


FIGURE 5-4. SPML use case

3. SPML Features

In SPML, there is:

- A core set of operations
- Request response protocol

a- SPML core operations

In the SPML 1.0 specifications, the following operations are ***defined as core: Add, Modify, Delete, and Search.*** This essentially means that all PSTs make these four operations available for manipulation as SPML-defined interfaces.

b- SPML request response protocol

SPML request response protocol covers how various SPML components talk to each other -- primarily the client (RA) and the server (PSP). This protocol dictates that the request issued by the client describes one or more operations to be performed on a specific service point. *You can submit these requests individually or in a batch, and select either synchronous or asynchronous submission for both request types.*

To complete your understanding of the request response protocol, you need to dig deeper into two different aspects of it:

- ***Request execution model (synchronous or asynchronous***)
- ***Grouping of requests (individual or batch)***

b.1- Request execution model

How SPML dictates support for synchronous and asynchronous execution of requests?

Requests processed synchronously are blocking calls. An RA keeps its execution call open until it receives a response on that call.
In the case of asynchronously executed requests, the RA issues each request with a unique request ID and does not wait for a response on the same execution call. The request ID is maintained by the PSP for the duration of the request execution, and is returned to the client along with the response when the response is ready.

While the request is processed, the client may be interested in knowing the status of the request. This is where the request ID comes in handy. The request ID is also used to control and manage the pending and executing SPML requests at the server.

SPML provides two operations to manage and control asynchronous request execution:

• **StatusRequest**: -- This operation allows clients to get the current status of the request that's being executed asynchronously.
• **CancelRequest**: -- This operation allows the RA to cancel the execution of a pending asynchronous request.

b.2- Grouping (Batch) of requests

The RA can issue a request either individually or in a batch. In either case, requests can be executed synchronously or asynchronously. Individual requests are handled in a standard manner, but the batch requests have a kink that you need to understand.
A batch request is multiple requests grouped together. With batch processing, it's important to understand three things:
• *How results are sent back for each of the requests* in a batch
• *How multiple requests are processed (in parallel or sequentially*?)
• *How errors are handled when some batch requests fail*

Result processing:

Multiple SPML operations are collected together and issued as a single BatchRequest. The results are processed by the PSP and sent as a BatchResponse based on positional correspondence -- which essentially means that the first response in a BatchResponse corresponds to the first request of the BatchRequest, the second response corresponds to the second request, and so on.

Processing types:

SPML supports both sequential and parallel processing of requests. The RA can ***determine how the processing is done.*** In the case of sequential processing, requests are processed in the order in which they are mentioned in the batch request. In case of parallel processing, the server can

execute the batched requests in any order. In both the cases, however, the response is determined by the positional correspondence.

Error handling:

SPML provides two options -- resume and exit -- for handling cases where some of the requests in a batch fail.
The RA specifies one of these as the preferred option at the time a batch request is issued. When the instruction to the server is to resume, the failure of an individual operation does not affect execution of the remaining requests, and the positional correspondence is maintained. When the instruction is to exit, the server terminates the execution of the remaining requests as soon as an error is encountered, and all the requests that do not execute are marked as failed.

3. How can I provision user accounts with appropriate privileges and manage entitlements for my users? The answer is: Use XACML.

4.4.1.3 eXensible Access Control Markup Language (XACML)

XACML is an OASIS, general-purpose, XML-based **access control language for policy management and access decisions**. It provides an XML schema for a general policy language which is used to protect any kind of resource and make access decisions over these resources.

The XACML context also specifies the request/response protocol that the **application** environment can use to communicate with the **decision point**. The response to an access request is also specified using XML.

Most applications (web or otherwise) have **a built-in authorization module that** grants or denies access **to certain application functions or resources based on entitlements assigned to the user.**

In a **centrally managed IAM architecture, application-specific authorization models make it difficult to state the access rights of individual users across all applications**.

XACML Goal:

The goal of XACML is to provide a standardized language, a method of access control, and policy enforcement **across all applications that implement a common authorization standard**. These authorization decisions are based on various authorization policies and rules centered **on the user role and job function**. In short, XACML allows for unified authorization policies (i.e**., the use of one consistent XACML policy for multiple services**).

The figure below illustrates the interaction among various health care participants with unique roles (authorization privileges) accessing sensitive patient records stored in a health care application.

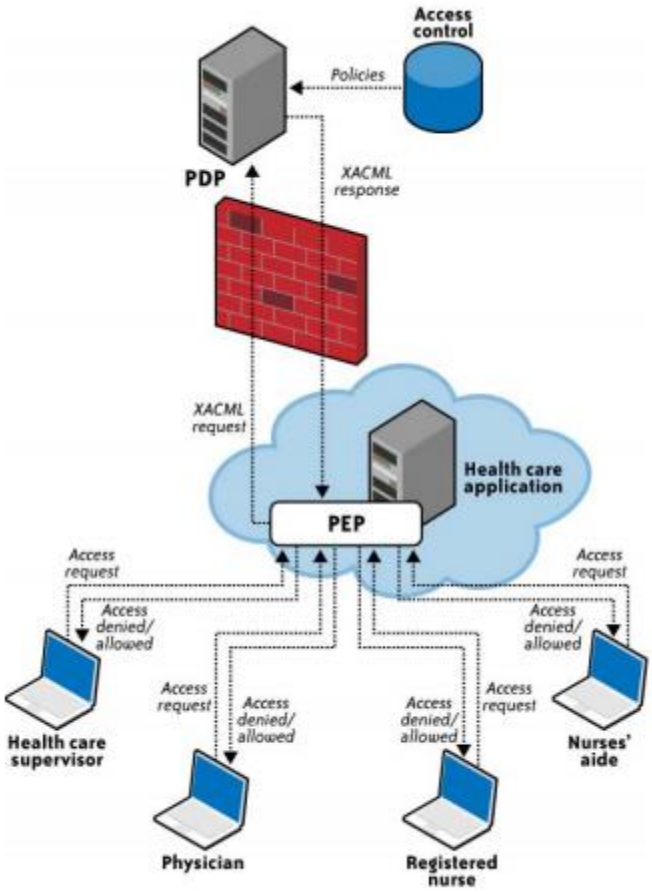


FIGURE 5-5. XACML use case

The figure illustrates the following steps involved in the XACML process:

1. The health care application manages various hospital associates (the physician, registered nurse, nurses' aide, and health care supervisor) *accessing various elements* of the patient record. This application relies on the **policy enforcement point (PEP)** and forwards the request to the PEP.
2. The PEP is actually the interface of the application environment. It receives the access requests and evaluates them with the help of the **policy decision point (PDP).** It then permits or denies access to the resource (the health care record).
3. **The PEP then sends the request to the PDP.** The PDP is the main decision point for access requests. It collects all the necessary information from available information sources and concludes with a decision on what access to grant. *The PDP should be located in a trusted network with strong access control policies, e.g., in a corporate trusted network protected by a corporate firewall.*

4. After evaluation, the PDP sends the XACML response to the PEP.
5. The PEP fulfills the obligations by enforcing the PDP's authorization decision.

The interaction takes place using a **request-response protocol with the XACML message as the payload**. In this way, XACML is used to convey the evaluation of policies against access decision requests.

4. How can I authorize cloud service X to access my data in cloud service Y without disclosing credentials? The answer is: Use OAuth.

4.4.1.4 Open Authentication (OAuth)

OAuth is an open standard for token-based authentication on the Internet. OAuth, which is pronounced "oh-auth," allows an end user's account information to be used by third-party services, such as Facebook, without exposing the user's password.

OAuth is an emerging authentication standard that allows consumers to share their private resources (e.g., photos, videos, contact lists, bank accounts) stored on one CSP with another CSP without having to disclose the authentication information (e.g., username and password).

OAuth and Login

OAuth and login must be separately understood. Assume we have a company where employees gain access to its building using their employee ID card. Now assume that an external guest needs to visit the company. If **login** stands for *an employee accessing the building*, **OAuth** stands for ***a guest receiving a visitor card and accessing the building***.

See the following example.

- An external **Guest A** says to the reception desk that he wants to meet **Employee B** for business purposes.
- The **reception desk** notifies **Employee B** that **Guest A** has come to visit him.
- **Employee B** comes to the reception desk and identifies **Guest A**.
- **Employee B** records the business purpose and identity of **Guest A** at the reception desk.
- The **reception desk** issues a visitor card to **Guest A**.
- **Employee B** and **Guest A** go to the **specified room to discuss** their business.

This example help you understand the procedure of issuing OAuth and the authorization. The *visitor card* allows visitors to access **pre-determined places only** which means that a person with a "visitor card" *cannot* access *all* the places that a person with an "employee ID card" can access. In that way, a user who has directly logged into the service can do more than a user who has been authorized by OAuth.

In OAuth, "***Auth***" means "**Authorization**" as well as "Authentication". Therefore, when authentication by OAuth is performed, the service provider (*reception desk*) asks whether a user (*employee*) wants to authorize the request of the third-party application (*guest*).

Understanding OAuth requires getting to know OAuth terminology in advance. The following table summarizes the key OAuth terminology.

Table 2. Key OAuth 1.0 Terminology.

Terminology	Description
User	A user who has **an account with the Service Provider** and tries to use the Consumer. (*An employee in our example.*)
Service Provider	Service that provides Open API that uses OAuth. (*The reception desk in our example.*)
Consumer	An application or web service that wants to use functions of the Service Provider through OAuth authentication. (*A guest*)
Request Token	A value that a Consumer uses to be authorized by the Service Provider. After completing authorization, *it is exchanged for an Access Token*. (*The identity of the guest.*)
Access Token	A value that contains a key for the Consumer to access the resource of the Service Provider. (*A visitor card.*)

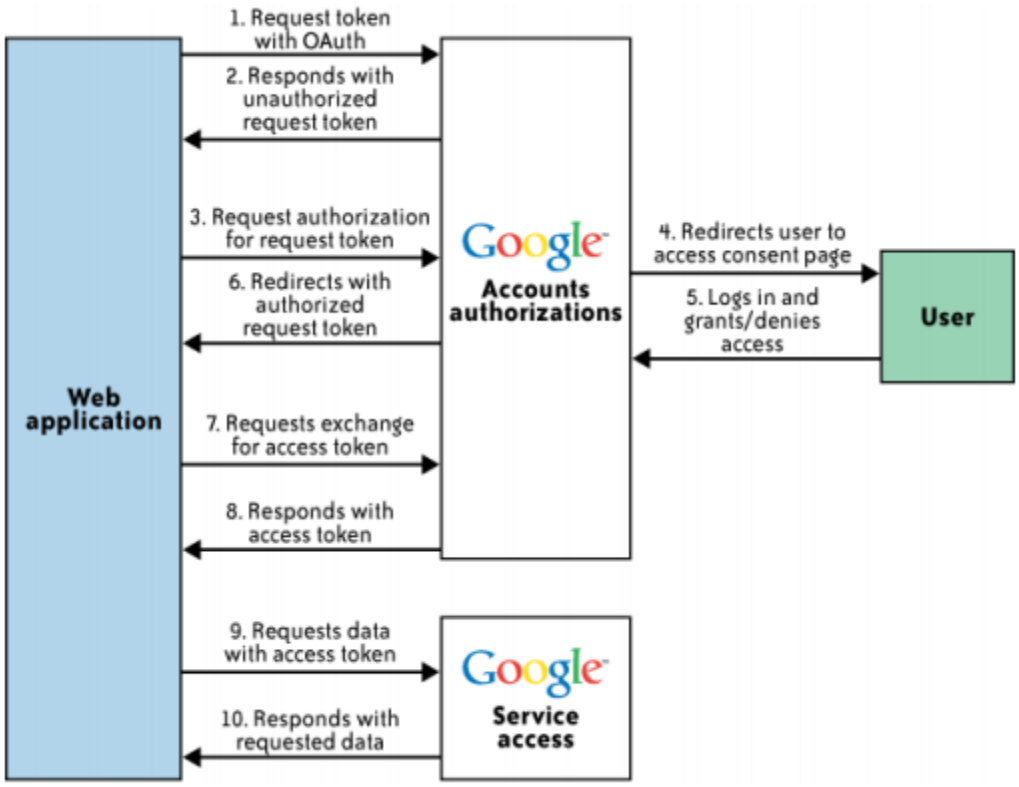
The following table shows the OAuth authentication process by comparing to the company visit process described before.

Table 3. Company Visit Process and OAuth Authentication Process.

Company Visit Process	OAuth Authentication Process
1. Guest A (an external guest) says to the reception desk that he wants to meet Employee B (an employee) for a business purpose.	Requesting for and issuing Request Token
2. The reception desk notifies Employee B that Guest A has come to visit him.	Calling user authentication page
3. Employee B comes to the reception desk and identifies Guest A.	User login completed
4. Employee B records the business purpose and identity of Guest A at the reception desk.	Requesting for user authority and accepting the request
5. The reception desk issues a visitor card for Guest A.	Issuing Access Token
6. Employee B and Guest A go to the specified room for the business.	Requesting service information by using Access Token

From this table you can understand the **Access Token** as a *visitor card*. With the visitor card, a visitor can access the permitted space. Likewise, a **Consumer** with an **Access Token** can call the available Open API of the Service Provider.

The figure below illustrates the sequence of interactions between customer or partner web application, Google services, and end user.



1-Customer web application contacts the Google Authorization service, asking for a request token for one or more Google service.
2. Google verifies that the web application is registered and responds **with an unauthorized request token.**
3. The web application directs the end user to a Google authorization page, referencing the request token.
4. On the Google authorization page, the user is prompted to log into his account (for verification) and then either grant or deny limited access to his Google service data by the web application.
5. The user decides whether to grant or deny access to the web application. If the user denies access, he is directed to a Google page and not back to the web application.
6. If the user grants access, the Authorization service redirects him back to a page designated with the web application that was registered with Google. The redirect includes the now-authorized request token.

7. The web application sends a request to the Google Authorization service to exchange the authorized request token for an access token.
8. Google verifies the request and returns a valid access token.
9. The web application sends a request to the Google service in question. The request is signed and includes the access token.
10. If the Google service recognizes the token, it supplies the requested data.