IAM

Basics

Introduction

Lexicon

NAME	DEFINITION	DESCRIPTION	MORE
SAML	Security assertion markup language	Standardized way to tell external applications and services that a **user is who they say they are**. SAML is a standard for transferring identity data between **two parties**: an **identity provider** (IdP) and a **service provider** (SP).	SAML makes **single sign-on** (SSO) technology possible by providing a way to authenticate a user once and then communicate that **authentication** to multiple applications.
IdP	Identity provider	An SAML indentity provider is an entity that **manages and stores user credentials**.	Performs authentication and **passes the user's identity and authorization level to the service provider**.
SP	Service provider	Trusts the identity provider and **authorizes the given user to access the requested resource**.	
IB	Identity borker	**Provide** a broad selection of **identity providers** and **authentication methods**.	Facilitates auth between service providers and their configured Identity Providers.
Auth0	Organization	Identity and service provider. Supports Social providers, MFA and biometric auths.	Auth0: Secure access for everyone. But not just anyone. You can also read my Auth0 tutorial.
OAuth 2.0	Protocol	**Standardized authorization protocol** that allows a user to grant access to their resources on one site, to another site, without having to expose their credentials.	
OIDC	OpenID Connect	**Protocol** based on Auth2.0 framework specifications.	
IAM	Identity and Access Management	IAM is a set of practices that **encompasses user authentication, authorization, and access control**.	
MFA	Multi-factor authentication		Auth with password and phone number.

Comparisons

Differences between OAuth and OAuth2.0

Most of the people developing the new version have working 1.0 implementations. So they all made sure it would be **trivial to upgrade**.

The providers with early 2.0 support use **Bearer tokens**, which are, send over **HTTPS** and do not include any cryptography on their own.

The main difference and where the transition can be more complex is when dealing with large **scale**. 2.0 handles scale significantly **better than 1.0**.

* Source: What's the difference between OAuth and OAuth 2.0? - Stack Overflow

Differences between OAuth2.0 and OpenID Connect

Main difference

OpenID is an extra identity layer **on top of the OAuth 2.0** security stack.

Authorization vs Authentication

OAuth only **AUTHORIZES** devices, APIs, servers with access tokens...

OpenID transaction procedure is the same as OAuth 2.0 authorization workflow. The significant difference is an '**id-token**' instead of an **access token** that allows the user **AUTHENTICATION**.

Id token vs access token

Since OpenID Connect is based on OAuth 2.0, OpenID Connect will **also** provide access tokens.

An id token is represented as a JSON Web Token (JWT).
An id token is a **security token** that contains **Claims** (information about user in jwt).
An id token is a JWT.

Access tokens are used as bearer tokens.
A bearer token means that the bearer (who hold the access token) can access authorized resources without further identification. They often have a **short lifespan**.

Process differences

In OAuth 2.0, at any time when a user wants to log in, he will be redirected to the login page, or a new pop-up page will appear for the authorization, unlike OpenID.

In OpenID, whenever a user wants to log in to a third-party app, he should enter his OpenID credentials to the 3rd-party applications.
After that, the 3rd-party app will redirect the user to the OpenID provider to confirm the login process.

State vs Nonce

OAuth2.0 uses only a **state parameter.** It's there to protect the end user from cross-site request forgery (CSRF) attacks by binding request and response.

OpenID inherits state from Auth2.0 but also uses **nonce parameters.** It binds the tokens with the original client requests.

* Source: Detailed difference OAuth 2.0 vs. OpenID (hitechnectar.com)
* Source: State and nonce in oidc - Stack Overflow
* Source: Tin Isles: How Does OpenID Work?
* Source: How Does OpenID Work? (windley.com)
* Source: Nonce and state differences - Stack Overflow
* Source: Id token Vs access token. ID Token | Medium

OAuth 2.0 vs OpenID vs SAML

OAuth 2.0

If you've ever signed up to a new application and agreed to let it **automatically source new contacts** via Facebook or your phone contacts, then you've likely used OAuth 2.0. This standard provides secure delegated access.
That means an application **can take actions or access resources** from a server on behalf of the user, without them having to share their credentials.
It does this by allowing the identity provider to **issue tokens to third-party applications** with the user's approval.

OpenID Connect

If you've used your **Google account to sign in to applications** like YouTube, or Facebook to log into an online shopping cart, then you're familiar with this authentication option.
OpenID Connect is an **open standard** that organizations use to **authenticate users**. IdPs use this so that users can sign in to the IdP, and then access other websites and apps without having to log in or share their sign-in information.

SAML

You've more likely experienced SAML authentication in action in the work environment. For example, it enables you to **log into your corporate intranet or IdP and then access numerous additional services**, such as Salesforce, Box, or Workday, **without having to re-enter your credentials**. SAML is an XML-based **standard for exchanging authentication and authorization data between Identity providers and service providers** to verify the user's identity and permissions, then grant or deny their access to services.

* Source: Nonce and state differences - Stack Overflow

Summary

OAuth 2.0
Used for user authorizations.

OpenID Connect
Used for authentication.

SAML
Used for exchanging authorization and authentication data between IdPs and SP.

Id-token
JWT security token containing user information.

Access Token
Is a bearer token.

Identity Provider
Manages and store credentials.

Service Provider
Trust and authorize Identity Providers.

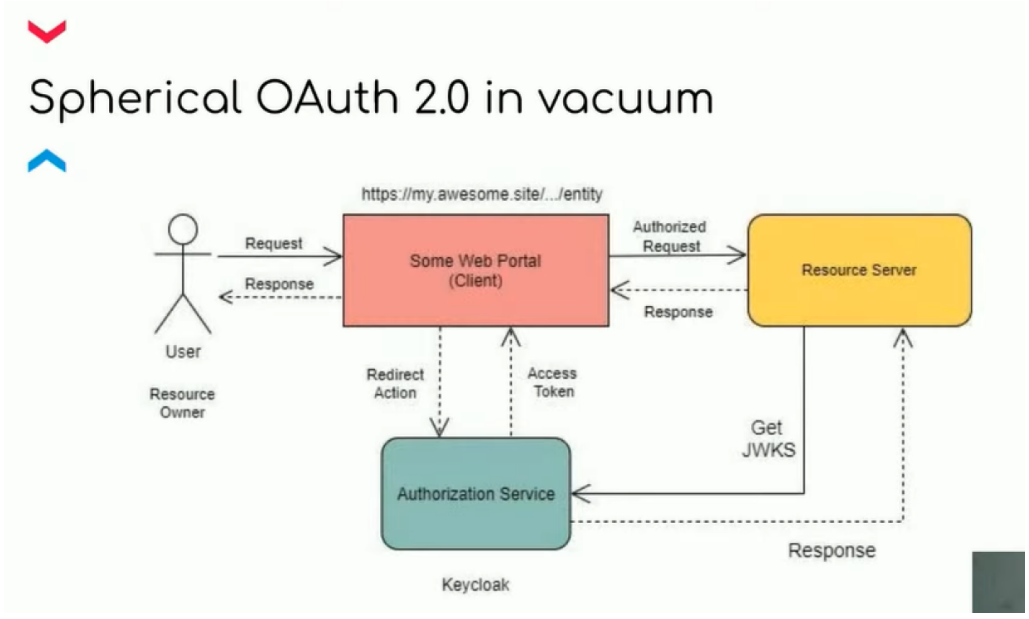
Keycloak

Introduction

Keycloak is an open source tool that helps build stable, convenient identity and access management (IAM) services into your applications.

Read redhat tutorial here.

It includes **OAuth2.0 OpenID and SAML** standard protocols.
This means that keycloak can manage **SSO, authentication and authorization** processes.



Lexicon

NAME	JOB
User	Resource owner
Client application	Web portal
keycloak	Authorization service (and more)
Resource server	A set of microservices

JSON Web Key Set

To perform a request to an API, you will **call that microservice's API**. But the microservice shouldn't give the API to just anyone, even if the request came with a token. It **must verify that the token is signed** by your authorization service (keycloak here).

Therefore, the microservice makes a **request for a JSON Web Key Set (JWKS)**, which is a **set of keys used to validate a token signature using a Key ID**. If the signature is valid, the process returns a response.

Definitions (The mall analogy)

Keycloak has realms, users, groups, clients and roles.
Keycloak SDK has built-in methods that allow you to perform HTTP requests from your microservices.

Keycloack

Consider Keycloak as a **shopping mall** with departments, which contains stores.

Clients

Consider **any department** in the mall as a client.
When you log into keycloak you log into a certain department.

Realms

Consider **any store** in the mall as a realm.

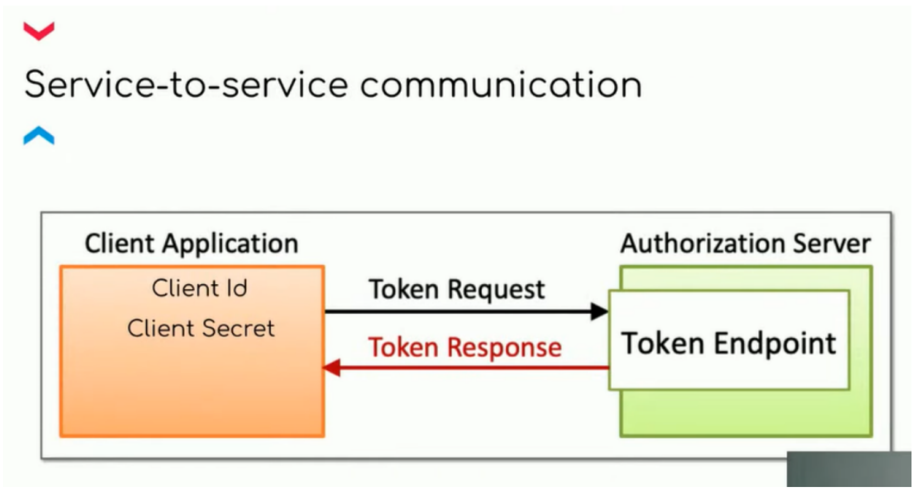
Roles

Consider **customers, cashiers, people who serve customers …** as roles.
Cashiers have one role, and customers have another.
Therefore, you need to **differentiate users by department and by shop**.
For example, if someone can take things free in shop one, it doesn't mean that they can take things for free in shop two.

Login URL

A standard keycloak URLs looks like this:
https://someDomain/auth/realms/myRealm/protocol/openid-connect/auth?client_id=myClient…

Authorization and authentication



To allow communication between your microservices, you'd use a Client id and a client secret authentication allowing them to communicate safely.

*Source : How to architect OAuth 2.0 authorization using Keycloak | Enable Architect (redhat.com)

Kerberos

Kerberos is a protocol that serves for **network authentication**.
Used for **authenticating clients/servers** in a network **using a secret cryptography key**.
Designed for providing strong authentication while communicating to applications.

Introduction

Mutual authentication

Mutual authentication is a fundamental concept in Kerberos.
It means that not only does the **user verify the identity of the service** they want to access, but the **service also verifies the identity of the user**.

Key distribution Center

The Key Distribution Center (KDC) is the heart of Kerberos.
It's a **trusted server** responsible for **managing authentication and issuing encryption keys**.
The KDC consists of two main components:

- **The Authentication Server (AS) authenticates users** and provides a Ticket Granting Ticket (TGT).
- **The Ticket Granting Server (TGS)** takes the TGT and **issues service tickets**, which allow users to access specific services.

Ticket-based authentication

Kerberos relies on a ticket-based authentication system.

1. When a **user logs in**, they **request a Ticket Granting Ticket (TGT)** from the AS.
   The TGT is a special ticket that can be used to request service tickets **without entering a password again**.
2. These service tickets are **used to access various services**, and they expire after a set time, enhancing security.

Security and Use case

Kerberos offers robust protection against common network attacks.

Kerberos also simplifies access control:
It's used to ensure that only authorized employees can access company resources like servers and databases.

Redhat – SSO

Red Hat SSO is considered being the leader in IAM solutions.
Of course it uses SAML, OIDC and OAuth2, but on top of that, adds a **lot of different functionalities**, like:

- **Tracking user**
- **Monitoring user accounts**
- **High availability** by clustering with cloud providers to comply with a **hybrid cloud philosophy**.
- **Kerberos**-based authentication

….

Redhat SSO solution is based on keycloak open-source project.

Introduction

Lexicon

NAME	DESCRIPTION
High Availability	Means the program is always available by having **running copies**.
Cloud Providers	Companies that offers space, power and more for your program to run.
Hybrid Cloud	This term indicates that you could have **multiple cloud providers and/or local data centers**.
Kerberos	Kerberos is an authentication **protocol that verifies the identity** of a user/host.

OpenShift Container Platform & Service accounts

OpenShift is a **Kubernetes-based container platform** developed by Red Hat. It includes several products and components to enable container orchestration and more.

Introduction

The core product, providing enterprise-level Kubernetes for **container orchestration**. It includes features for **scaling applications, monitoring, and managing containerized workloads**.

Containers

Containers are a lightweight way to package and run applications and their dependencies, making it easier to develop, deploy, and maintain software.

CI/CD

OpenShift includes CI/CD (Continuous Integration/Continuous Deployment) capabilities through OpenShift Pipelines.

Monitoring and logging

OpenShift offers built-in monitoring and logging solutions, including integration with **Prometheus for monitoring** and **Elasticsearch/Fluentd/Kibana (EFK) for log management**.
This helps operators and developers track the health and performance of applications and the platform.

Introduction

A service account is an **identity used by applications, pods (containers), or processes** running **within a Kubernetes or OpenShift cluster to interact with the Kubernetes API server**.
It allows these **entities to authenticate themselves** and gain specific permissions to access resources.
Note that Service Accounts don't use traditional IAM recommendations since they operate only within the cluster.

Kubernetes and OpenShift support **two types of service accounts**:

- **User Service Accounts**:
  Applications and pods within the cluster to **interact** with the Kubernetes API typically use these.
- **Service Account Tokens**:
  These are tokens associated with service accounts and are used for **authentication** when making requests to the Kubernetes API server.

Roles

Service accounts are typically **associated with roles** and **role bindings** to define their permissions.
A role specifies what actions can be performed on specific resources (e.g., pods, services), and a **role binding associates a service account with a role**, granting it those permissions.

Tokens

When you **create a service account**, a **unique service account token is automatically generated** and stored as a **secret** within the same namespace.
Applications and pods can use this token to **authenticate** themselves when making requests to the Kubernetes API.

Exactly like Keycloak access tokens they are used as bearer tokens ☜(ﾟヮﾟ☜).

By default, **every pod** in Kubernetes and OpenShift **is associated with a service account called "default."** This is used when no specific service account is specified for a pod.

Made by :

Matthias Brat

Socials:

github.com/matthiasbrat

stackoverflow.com/users/17921879/matthias-b

matthiasbrat.dev