Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 01 January 2024

Central Authentication for Api Consumers on Multiple Api Gateways

Vishal Vikas Javalkar
IT Dept.,Cencora Dallas, US

Abstract **– Application Programming Interface (APIs) unlocks access to a business's critical data and processes which allows organizations to more efficiently reuse capabilities, share assets, and innovate with partners. Typically, organizations have hundreds of applications and unlocking and sharing data from these applications is critical capability for organizations. It allows organizations to make the best use of its data assets and ensure business processes run efficiently and seamlessly. API are created out of these applications to allow integration with other internal or external applications and often API Gateways are used to govern, manage and protect these APIs. Large organizations end up using multiple API Gateways depending on where the application workloads are running either on-premises, cloud , hybrid and depending on geographies due to data residency and processing laws. Each API Gateway comes with its own local identity management component to authenticate the API consumer, with multiple API gateways it can easily lead to an inconsistent adoption of API authentication standards, policies, and duplication of API consumers identities. This further leads to weakened security, additional maintenance overhead, and lack of visibility and central governance of APIs and API consumers. This paper recommends and outlines an approach for organizations to effectively address this challenge.**

Keywords—API Gateway, Application Programming Interface (API), API Management, API Authentication

I. INTRODUCTION

APIs unlock critical business data therefore its imperative to securely manage, administer the access and systematically govern the API consumers. API Gateway allows organizations to manage and govern the API and its API consumers. There are many API Management products out in the market, however large organizations end up having multiple API Management gateways depending where the applications and applications workload are running, this could be on on-premises, cloud , hybrid and even depending on geographies due to data residency and processing laws. Enabling APIs through various API Gateways and using its own local Identity Management leads to duplication of consumers, non-standardized authentication methods and lack of central governance and visibility. This paper recommends an approach on how to effectively introduce and manage central authentication to multiple API Gateways in an organization.

A. Approach for Central Authentication for API Gateways

There are different methods to authenticate API consumers on API Gateways while consuming an API. Below are some of the popular methods.

1. Basic Authentication Basic and Bearer authentication scemes are widely used API authentication methods. They both use HTTP headers to authenticate API users, and they can be used in combination with API Keys for added security.
   HTTP Basic Authentication: API consumers send API requests with a username and password in an HTTP header. API providers then validate the credentials to authenticate API users. This simple API authentication method lacks security, as API requests can be intercepted easily.
   HTTP Bearer Authentication: API consumers send API requests with a unique API access token in an HTTP header. API providers then validate the API access token to authenticate API users. This API authentication method is more secure than Basic, as API requests cannot be intercepted easily.
2. API Key API key-based authentication involves sending an API key along with a request. An API key is a unique identifier that is issued by the API provider to authorized users or applications and is used to identify and track API usage.
   To use an API that requires key-based authentication, the user or application includes the API key as a parameter in the request, typically as a query parameter or in a header. The API provider verifies the key and then allows or denies access to the API based on the user's permissions.
3. OAuth 2.0 OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification and its extensions are being developed within the IETF OAuth Working Group.
   OAuth 2.0 is an authorization framework that allows users or applications to access resources from an API without giving the API access to their credentials, such as a username and password. It is widely used to grant third-party applications access to resources on behalf of a user, without requiring the user to disclose their credentials to the third party.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 01 January 2024

OAuth 2.0 is widely used by social media platforms, cloud service providers, and other web applications to provide a secure and standardized way of granting access to resources. It provides security advantages over traditional authentication schemes, including the ability to revoke access to a specific application or user, the ability to grant limited access to specific resources, and the ability to delegate authentication to a trusted third party.

4. JWT Based Authentication JWTs (JSON Web Tokens) are a compact and URL-safe means of representing claims to be transferred between parties. JWTs consist of three parts separated by dots: a header, a payload, and a signature. The header specifies the algorithm used to sign the token, the payload contains the claims, and the signature is used to verify the integrity of the token.

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.

JWTs are often used for authentication and authorization in web applications. When a user logs in, the server generates a JWT that contains information about the user, such as the user ID and permissions. The JWT is then signed using a secret key that only the server knows. The server sends the JWT to the client, which can then use it to access protected resources on the server. When the client sends a request to the server, it includes the

JWT in an authorization header. The server verifies the signature of the JWT using the secret key, and if the signature is valid, it extracts the claims and uses them to authorize the request.

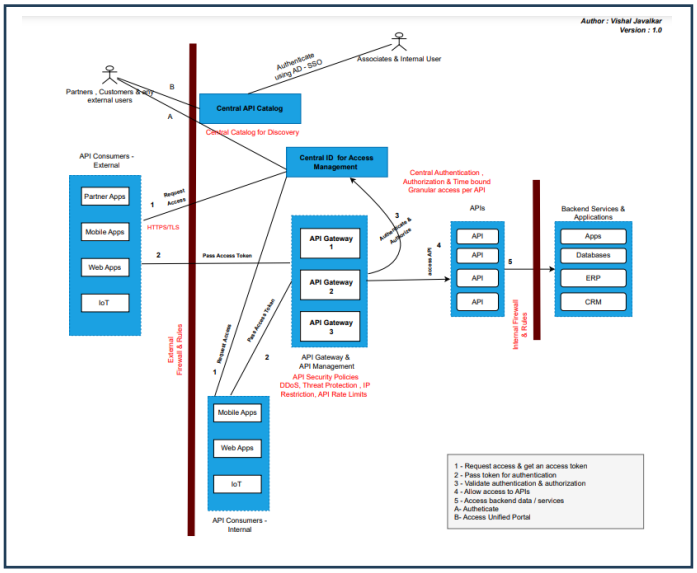
JWTs are widely used in REST APIs, as they allow the stateless transmission of authentication and authorization data between the client and the server. They are also portable since they can be easily shared between different services and systems.

JWTs are widely used and most suitable for REST APIs. It is recommended to use JWT based authentication over the other methods.

Although all API Gateway provides a way to generate JWT token often their capabilities are limited and are designed to work as a central API Identify and Access Solution, hence its critical to create a capability dedicated for API Access & Identity Management. A centralized API Access & Identify Management solution will allow organizations to govern and manage the consumer identities centrally where the access is controlled based on the tokens which are short lived, provides role based granular access and can be used to easily provision or revoke access to any API.

B. Reference Architecture for Central Authentication of API Consumers on API Gateways

Below diagram depicts the reference architecture for central API Authentication for API Consumer on API Gateway. It describes various components and entities involved to achieve a central API Authentication capability in an organization.



Components or Entities involved in the reference architecture

1) API Consumers API consumers are the applications which will consuming the API which are exposed by the API Gateway and Backed Services or applications. API Consumers can be either external application of parterns , customers , vendors or it could internal applications with the organization.

2) API Gateway: API Gateway are the specialized API Management solutions which are designed to publish, manage and govern the APIs. It acts as proxy to your actual backend APIs. This provides a way to secure and systematically document and discover APIs.

3) Central Identity for Access Management : Central Identify for Access Management will act as a centralized solution for API Authentication & authorization where are the API consumers are registered and are centrally governed and managed.There will be a trust established between API Gateways and the Central API Authentication Servers. This will allow centralized access management of all the API consumers. There are additional advantages of having such architecture.

a. Central Governance, Visibility & Access Management of API
b. Role based Authorization of API
c. Short lived tokens for enhanced security
d. Security Standards Compliance
e. Rule based API access
f. Easy provisioning and de-provisioning of API access

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 01 January 2024

4) Central API Catalog : If your organization is using multiple API Gateways it become difficult to discover and reuse APIs hence considering a solution where all the APIs irrespective which API Gateways there are exposed from are listed on the Central API Catalog.

5) Backend APIs and Backend Applications : These are your API which unlock your data or opens up business processes for integrations.

6) Partners ,Customers , Any External User, Employees or any Internal Users : These are the developers either internal or external who wish to consume API from the API Gateways.

7) External and Internal Firewalls : Ensure your API Gateways and backed APIs are behind the External and Internal Firewalls respectively for enhanced security. Also ensure all the communication is on HTTPS (TLS 1.2+) to data over transmission.

API Consumer Authentication Flow using Central Authentication Server and API Gateway:

Step 1: API Consumer (External /Internal) requests access to Central ID server for the JWT access token. Central ID server issues a JWT token to the API consumer.

Step 2: API Consumer passes the token to API Gateway, where API Gateway validates the authenticity of token and the authorization with the Central ID.

Step 3: Central ID validates the authenticity and authorization of the consumer.

Step 4: Upon successful validation API Gateway allows access to backend APIs.

Step 5: Backend APIs performs any of CRUD operation on the backend application or execute any other function with API.

This process allows a standardized and secure way to access API and backed application data. Thereby enhancing reuse, avoiding consumer application identify duplication and brings in central governance and visibility to all the API consumers.

II. CONCLUSION

The white paper discusses the challenge of managing authentication for API consumers on multiple API gateways within an organization. It highlights the need for central authentication to ensure consistent adoption of authentication & authorization standards, reduce security risks, and improve governance and visibility.

The paper describes various authentication methods such as Basic Authentication, API Key, OAuth 2.0, and JWT Based Authentication. It recommends that JWT-based authentication is the most suitable option due to its portability and ability to provide role-based granular access.

To implement central authentication, the paper presents a reference architecture consisting of several components:

API Consumers: Applications that consume APIs exposed by the API gateway.

API Gateway: Specialized solutions used to publish, manage, and govern APIs.

Central Identity for Access Management: A centralized solution for registering and governing API consumers' identities.

Central API Catalog: A catalog listing all APIs regardless of the API gateway they are exposed from.

Backend APIs and Applications: The APIs that unlock data or enable integrations with business processes.

Partners, Customers, External/Internal Users: Developers who wish to consume APIs from the API gateways.

External/Internal Firewalls: Security measures to protect the API gateways and backend APIs.

The paper also outlines an authentication flow using a central authentication server and an API gateway. The steps include requesting a JWT access token from the central ID server, validating the token at the API gateway, validating consumer authenticity and authorization at the central ID server, and granting access to backend APIs upon successful validation.

Overall, this white paper provides insights into implementing central authentication for API consumers on multiple API gateways within an organization.

REFERENCES

[1] Authentication
https://en.wikipedia.org/wiki/Authentication
[2] OAuth 2
https://oauth.net/2/
[3] JWT Authentication
https://jwt.io/introduction
[4] API Authentication methods
https://blog.hubspot.com/website/api-authentication