BROADCOM®
SOFTWARE

5 OAuth Essentials for API Access Control

Introduction: How a Web Standard Enters the Enterprise

OAuth's Roots in the Social Web

OAuth puts the user in control of delegating access to an API. This allows one service to integrate with another service on behalf of that user. The same social Web providers who popularized the pattern of exposing an API to enable third-party developers to enrich their platforms were the first ones to apply such delegated authorization mechanisms. OAuth was defined in 2006, to standardize mechanisms of this kind.

OAuth Comes to the Enterprise

Riding the popularity of APIs on the Web and social media, enterprises increasingly use APIs to expose their data, application functionality and other information assets to external departments, partners, customers and the general public. Access to enterprise APIs must be controlled and although some of these APIs rely on proprietary authentication schemes, OAuth provides standard patterns upon which API providers can base their token issuing.

OAuth has moved beyond its Web origins and is now a common component of enterprise APIs. Consequently, the OAuth standard has grown to accommodate an expanded range of use cases and requirements. The original OAuth 1.0/1.0a standard was mainly designed for third-party authorization on the Web. The recently-formalized OAuth 2.0 standard introduces new "grant types", which address a wider range of use cases, relevant to API access control in general.

Social Media Example: Letting Twitter Post Tweets to Your Facebook Wall

OAuth is used to automatically syndicate a user's Twitter posts to that user's Facebook wall. To do this, Twitter must be allowed to act on behalf of the user on Facebook. Exposing the user's Facebook login credentials to Twitter would create privacy and security issues. Instead, the user is redirected between Twitter and Facebook, in order to express consent. The output of this "handshake" is a token, which is used by Twitter when calling the Facebook API—allowing Twitter to effectively write on the user's Facebook wall.

Introduction: How a Web Standard Enters the Enterprise (Continued)

Enterprise API Example: Charging an Ecommerce Purchase to Your Mobile Bill

A mobile carrier creates an API that allows ecommerce sites to give customers the option of charging purchases to their monthly mobile bills. OAuth is used to enable communication between the carrier, the retailer and the customer. Any customer who chooses the "bill to my mobile account" option on the ecommerce site is redirected to the carrier's server in order to authorize the ecommerce store to temporarily access the customer's account there and make the charge.

The Challenges of Leveraging OAuth for Enterprise APIs

OAuth standardizes a number of important access-related challenges for API publishers. However, deploying OAuth as an authorization mechanism for enterprise APIs raises challenges around scalability, correct usage and integration. To make matters worse, OAuth is a relative newcomer to the enterprise; it is not supported by existing infrastructure and is poorly understood by many enterprise architects.

In this eBook, we outline five key considerations for organizations that are thinking of using OAuth as a mechanism for controlling access to enterprise APIs. These considerations will give you the basic knowledge and framework you will need in order to address the complex challenges associated with implementing OAuth and managing an OAuth provider. We also provide links to additional resources that deliver more in-depth knowledge.

Overview: 5 OAuth Essentials for API Access Control

Design Your Solution for Optimal Performance & Scaling

ARCHITECT YOUR TOKEN ISSUING AND VALIDATION TO ENSURE EFFICIENT PERFORMANCE AND SCALING

Deploy as Much Security as You Want & Need

OAUTH BY ITSELF IS NEITHER SECURE NOR INSECURE, SO YOU MUST TAILOR IT TO YOUR SPECIFIC NEEDS

Integrate with Existing Identity & Access Infrastructure

LEVERAGE EXISTING INVESTMENTS – DON'T REINVENT THE WHEEL OR CREATE IDENTITY SILOS

Design & Enforce Token Governance Policies

MANAGE THE TOKEN LIFECYCLE AND ENABLE TOKEN REVOCATION TO CONTROL SECURITY CENTRALLY

Use the Appropriate Grant Type for Your Use Case

CHOOSE THE OAUTH 2.0 PATTERN THAT MEETS THE REQUIREMENTS OF YOUR APPLICATION

Design Your Solution for Optimal Performance & Scaling

ARCHITECT YOUR TOKEN ISSUING AND VALIDATION TO ENSURE EFFICIENT PERFORMANCE AND SCALING

WHAT

Your OAuth solution should be designed to ensure that the process of OAuth-based authentication does not impact the performance of applications built against your API.

In particular, your solution must be scalable, so that app performance is not impacted as the popularity of your service increases and API calls multiply.

To achieve optimal performance and scalability, you will need to architect distributed enforcement points for your API access rules as well as a central authorization server.

WHY

The kinds of token issuing and token verification operations associated with OAuth-based access control can be highly resource-intensive.

As your API becomes more popular, its ability to support more demand will be tied to your ability to issue and verify more tokens.

With your OAuth solution coming under ever-greater strain, latency is likely to increase – and latency is a user experience killer.

HOW

The most efficient way to architect an OAuth solution is to use API Gateways, deployed either

as hardware networking appliances or virtually, in the cloud.

To maximize the scalability of your solution and increase fault tolerance, these API Gateways should be quickly and easily clusterable.

A full-functioned API Gateway should also be able to optimize performance by caching token validity at the perimeter and accelerating the issuing process.

Deploy as Much Security as You Want & Need

OAUTH BY ITSELF IS NEITHER SECURE NOR INSECURE, SO YOU MUST TAILOR IT TO YOUR SPECIFIC NEEDS

WHAT

Strong authorization control is essential for protecting APIs against attack and misuse. OAuth has emerged as the leading authorization technology for API security.

One of the great benefits of OAuth (especially OAuth 2.0) is its flexibility. You should leverage this to create a solution tailored to your specific needs and requirements.

It is important to remember that OAuth is not inherently secure or insecure and that OAuth-based authorization will only be one element of a complete API security solution.

WHY

APIs can significantly increase your organization's attack surface. Therefore, they must have strong authentication controls to protect them against unauthorized access.

Each API will require a unique level and type of security. Some API calls are more sensitive than others; some involve private information or financial transactions.

OAuth does not dictate how identity is validated; it simply provides guidelines for token issuing patterns. It is up to you to deploy patterns appropriate for you use case.

HOW

Your OAuth authorization server should be integrated with strong, multi-factor authentication, wherever this is applicable.

Ideally, your API Gateway should have templates that will simplify the process of designing token governance policies and OAuth patterns appropriate to your use case.

The Gateway technology should also include a runtime policy enforcement layer that will make it easier to enforce and manage the policies and patterns for multiple APIs.

Integrate with Existing Identity & Access Infrastructure

LEVERAGE EXISTING INVESTMENTS—DON'T REINVENT THE WHEEL OR CREATE IDENTITY SILOS

WHAT

Typically, an enterprise will already have an identity and access management (IAM) infrastructure, consisting of technology solutions, identity stores and policies.

Wherever possible, these existing IAM investments should be integrated with your OAuth architecture for reuse in the API access control process.

In particular, you should aim to provide Single Sign-On (SSO) access for individuals who already have identity attributes with your organization (e.g. internal developers).

WHY

The chances are your organization has already invested significant time, money and expertise in mature IAM systems. It would make no sense to replicate all this effort.

Requiring existing users to create and manage whole new sets of login credentials to access your APIs will frustrate developers, reducing engagement with your APIs.

From your perspective, having users with different identity attributes located across multiple identity silos will inevitably become a security and management nightmare.

HOW

The API Gateway you choose must be specifically designed to fully integrate with your existing IAM technologies and identity stores.

This will allow you to link existing identity providers to your authorization server policies and configure SSO agents in order to verify incoming cookies at handshake time.

You will also be able to look up identity attributes and group memberships in authorization and resource server policies to enhance runtime authorization decisions.

Design & Enforce Token Governance Policies

MANAGE THE TOKEN LIFECYCLE AND ENABLE TOKEN REVOCATION TO CONTROL SECURITY CENTRALLY

WHAT

OAuth works by issuing tokens that provide temporary access to resources exposed by APIs. It is vital to ensure these tokens provide appropriate levels of access.

To achieve this, you must design and enforce policies that define the lifespan of the tokens you issue as well as a range of other lifecycle considerations.

Other aspects of managing the token lifecycle include creating an audit trail of token session information and deciding whether to support token revocation and refresh.

WHY

Because every API (and every app built against every API) has its own security requirements, each will need a unique set of token lifecycle policies.

API publishers, third-party application providers and app users all need ways to manage their live permissions and revoke their tokens.

With all these elements affecting the token lifecycle across a distributed architecture, centralized token governance becomes crucial to retaining control of API security.

HOW

An OAuth-enabled API Gateway will allow you to decouple token management implementation from the perimeter OAuth endpoints.

Tokens should be indexed in the Gateway according to key properties (such as client ID, user ID and scope) to make retrieval quicker and easier.

Additionally, the API Gateway will make it possible to connect subscriber portals, administrative tooling and business intelligence components to the correct tokens.

Use the Appropriate Grant Type for Your Use Case

CHOOSE THE OAUTH 2.0 PATTERN THAT MEETS THE REQUIREMENTS OF YOUR APPLICATION

WHAT

In OAuth-based access, the API publisher provides an authorization grant to the client app, allowing the app to issue the user a token.

The relatively-new OAuth 2.0 specification defines four core grant types: authorization code, implicit, password and client credentials.

Additionally, a number of extension grant types have become available (e.g. SAML bearer, JWT bearer) and the list of these extensions is growing.

WHY

Different grant types address the needs of different scenarios e.g. user or application authentication, confidential or public clients, Web or mobile app.

The grant type will affect the user experience. Users expect different experiences in different scenarios and the choice of grant type should reflect this.

For example, although users will provide credentials directly to applications they trust, they expect to be redirected in order to authorize third-party applications.

HOW

Your API Gateway should provide implementation templates for all the core grant types plus select extension grant types.

The Gateway should also allow you to set policies defining specific patterns allowed for each grant type and to disable grant types not used in authorization server policies.

It is important to give developers comprehensive documentation detailing which grant types will work with your APIs.

Conclusion: Deploying an OAuth-Capable API Management Infrastructure

OAuth's Roots in the Social Web

In enterprise use cases, OAuth should be applied as a component of a complete API Management infrastructure. As part of this infrastructure, an API Gateway should be used to:

- Expose OAuth endpoints (e.g. the authorization server, the resource server)
- Provide OAuth templates, to which APIs and existing identity infrastructure are attached
- Manage tokens to control and distribute session information

The Layer7 API Gateway deliver all this functionality. The Layer7 API Gateway includes an OAuth Toolkit, which features a range of easily-customizable implementation templates. The OAuth Toolkit can easily be integrated with common enterprise IAM technologies like Active Directory and Oracle Access Manager. Layer7 API Gateways offer the simplest and most powerful way to leverage OAuth as part of a complete enterprise API Management program.

LEARN MORE | **About Securing Your APIs**
Click here to learn more about securing your API

Layer7 API Management provides enterprises with a comprehensive set of solutions that externalize APIs in a secure, reliable and manageable way.

PLEASE VISIT **THE LAYER7 PRODUCT PAGE AT BROADCOM.COM**

BROADCOM® SOFTWARE

For product information please visit our website at: broadcom.com

BC-XXXXEN August 2021