International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 6, Issue 4, 91-95, 2025
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V6I4P113

Original Article

Securing Broadband Subscriptions on SaaS with OAuth2.0 in REST API

Amey Deshpande
Sr. Cloud Delivery Manager, Calix Inc., McKinney, TX, USA.

Received On: 16/09/2025 ***Revised On: 20/10/2025*** ***Accepted On: 27/10/2025*** ***Published On: 16/11/2025***

Abstract - *The rapid evolution of cloud-native technologies such as Software as a Service (SaaS) has transformed the landscape for Broadband Service Providers (BSPs), enabling the delivery of subscriber services with high scalability and efficiency. Initially, the adoption of such platforms was driven with a cause for simplifying workflows and increase usability for BSP operational teams. Over time, however, both BSPs and their associated Operations Support Systems (OSS) and Business Support Systems (BSS) have become increasingly reliant on these cloud-based solutions. This has resulted in the accumulation and management of large quantities of sensitive data, including personally identifiable information (PII), as well as critical internet and voice service records for individual subscribers. Basic authentication method typically involved the exchange of simple username and password which were stored in databases. Basic auth. proved inadequate in securing sensitive information against evolving security threats. As regulatory requirements and industry standards have grown more stringent, BSPs have been compelled to transition to more robust authentication frameworks. OAuth 2.0 has emerged as the industry standard for secure data transport, offering enhanced protection through token-based authentication and granular access controls.*

Keywords *- OAuth 2.0, REST API Security, SaaS Security, Broadband Service Providers (BSPs), Token-Based Authentication, Access Control, OSS/BSS Integration, Cloud-Native Architecture, PII Protection, Secure Data Transport.*

1. Introduction

Calix is a platform provider for Internet Service Providers (ISPs) and BSPs across the United States and offers a variety of SaaS products. SaaS products are used as a platform by the BSP for integrating the data related to their network and service offerings.

1.1. At a high level it includes

- Network Layer 2 switches
- Network alarms reporting
- Customer Premises Equipment (CPE) such as Wi-Fi routers.
- Optical Network Terminal (ONT)
- Subscriber billing data form the Business Support Systems (BSS)

To enable data transport over the internet between two or more applications requires the use of an Application Programming Interface (API). APIs are the pragmatic access to the data and systems. It is an interface defined by the user to data and system that is consumed by the applications [1]. In the context of API integration, communication occurs between two endpoints: the server, which provides the API, and the client, which submits data for integration through the API. APIs commonly utilize the Hypertext Transfer Protocol (HTTP) for communication and employ a Uniform Resource Locator (URL) to designate specific HTTP endpoints.

Calix as the provider of API in the integration with BSP and their corresponding BSS has adopted JavaScript Object Notation (JSON) as a format for data transfer. JSON comprises of “keys” and “values”. For a client to be able to be allowed communication to a server using an API endpoint it has to get authenticated. Use of Basic Auth only requires a username and password. The client takes these two credentials, smooshes them together to form a single value, and passes that along in the request in an HTTP header called Authorization [2].

2. Basic Auth

Calix being the API provider was the authentication server in permitting communication between Calix software and the BSP or BSS database.

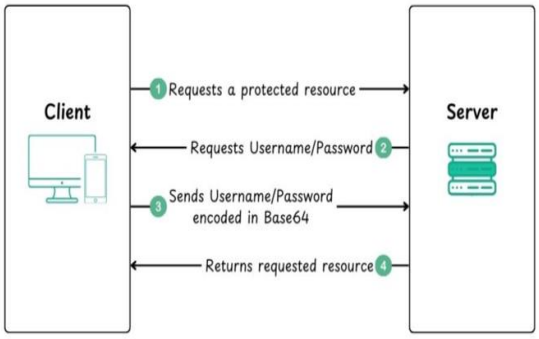

Fig 1: Client-Server Intrface Steps Using Basic Auth[3]

2.1. How Calix SaaS products implemented Basic Authentication

- The Calix web server was set to prompt clients for authentication when they requested access to protected resources as part of the Saas.
- The server used the default credential prompting mechanism that are offered by web browser or login forms.
- Using a pre-defined server logic the credentials provided by the client were verified against a secure user database. This meant that the username and password for all such BSPs, while securely, were stored permanently into the server end database.

2.2. Pros and Cons of using Basic Auth:

- Advantages: Basic Authentication offers straightforward implementation and minimal configuration requirements for the client, making it a suitable option for smaller BSPs with constrained resources.
- Limitations: The most glaring issue with using basic auth is the credentials are vulnerable to interception. There was no mechanism to expire or revoke the authenticated request hence each request would require the credentials to be included in the header.

BSPs, particularly those operating with constrained internal resources, found significant value in the straightforward approach offered for integrating their network infrastructure and BSS data with the Calix SaaS platform. The simplicity of this integration process was especially advantageous, as it enabled BSPs to leverage advanced, state-of-the-art solutions without having the need for substantial investments in technical expertise or financial outlay on the client side. The integration of BSS was intentionally designed to minimize complexity; BSPs could seamlessly authenticate their web applications as clients with the Calix servers, utilizing standard HTTP protocols to facilitate secure and efficient data exchange. This ease of implementation not only reduced the operational burden on BSPs but also accelerated their ability to adopt and benefit from innovative cloud-based services, thereby enhancing their overall service delivery capabilities.

3. Api Keys

While Basic Auth remains a valid authentication method, its reliance on the same set of credentials namely, a single username and password for both API access and account management was suboptimal from a BSP's security and operational standpoint. As more web applications were drawn towards the Calix Saas platform it became imperative to be compliant with the industry standards within the modem cloud environment.

Entrusting credentials to all internal resources regardless of their role, or even a third-party contractor, introduced significant risks for the account owner. In the event that a user on the client end acted in bad faith, it could prevent the legitimate owner from accessing their own account. A considerable next step beyond the use of Basic Auth method with the Calix platform was the use of API keys. This concept introduced benefits where the client using the API had different permissions than that of the account owner.

To implement key-based API, the Calix server would be required to generate one or more secret keys or key pairs. These keys would act as credentials that client side web applications use for authenticating API requests. The client user would copy the keys from the Calix cloud's web server and store locally. This posted security risks as the keys are in clear text and the user's computer may not be secured against web threats. Usability is an issue due to the cumbersome manual handling of the keys [4]. If the key was encrypted before storing, it would need to be decrypted before use, while staying in clear text inside the local storage. In case of a security breach on the client application, the key could be easily retrieved making it vulnerable.

Calix recognized that API keys, while solving the drawbacks of Basic Auth, would bring its own security limitation for handling sensitive user data due to its very static nature and potential for exposure if stored locally in client code and logs. API keys could have led to errors and complicated the setup process. BSPs have millions of subscribers and their Persinally Identifiable Information (PII) data is the most critical compmenet when it comes to data integration over cloud.

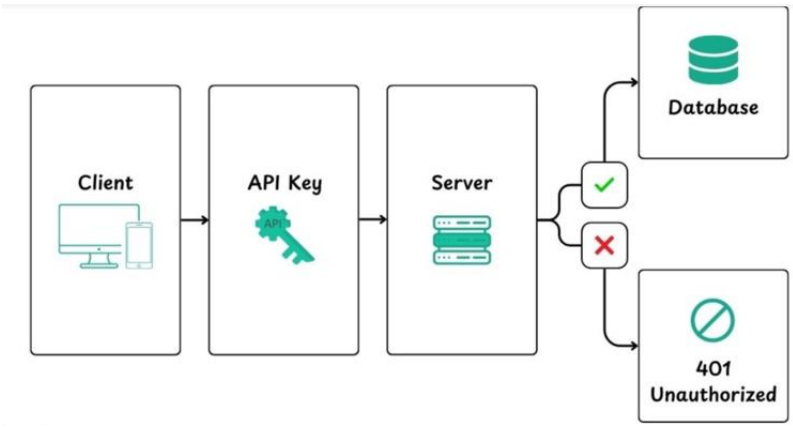

Fig 2: Client-Server Intrface Steps Using API Key Authentication[3]

Using API keys as a long term authentication solution was not the most secure and trusted way of data integration. For enabling automated workflows and seamless communication with the BSP network and the BSS databases, a more streamlined solution such as OAuth was preferred.

4. Oauth2.0

4.1. Use Case

Calix SaaS platform saw a surge in requests from BSPs all over US to integrate network equipment and subscriber data together for a cloud native monitoring and support product. A new product was launched that was built on the foundation of new API endpoints compliant with a highly secure OAuth2.0, the most widely used API standard across all API servers. OAuth is an open security standard that enables users to provide specific and time-bound rights to an application to access protected user resources. It is stored on some external resource servers without needing them to share their credentials with the application [6].

4.2. Design

In the SaaS architecture the OAuth2.0 actors such as resource owner, resource server, client application and authorization server were strategically designed to accommodate the 3 party intgegration involving Calix, the BSP and their BSS provider which hosted the web application. A BSP would be granted access to the SaaS platform on a domain-level, wherein all users that signed up under the domain would be eligible to reach the SaaS product using the domain's own authentication. This did not involve the authentication server used for API integration. Typically, this domain would be hosted and controlled by the BSP and acted as the resource owner.

4.3. Solution

Since OAuth2.0 is compatible with the use of Role Based Access Control (RBAC), the Calix server, while reachable to any user in the domain could be further classified for access using roles under RBAC. Roles, with certain set of permissions would be assigned to the users in the BSP's domain, and only those with appropriate roles would be granted access to the SaaS product.

5. Security

API is an interface of applications where client web application on one end of the interface can call on API endpoints only if there is an application on the API producer's end. To fulfill this standard requirement, and maintaining unique interfacing from the web application to the BSP, the BSP would be required to create their own application on the Calix SaaS platform which acted as the resource server. This application generated unique credentials required by the client for authenticating with the application server.

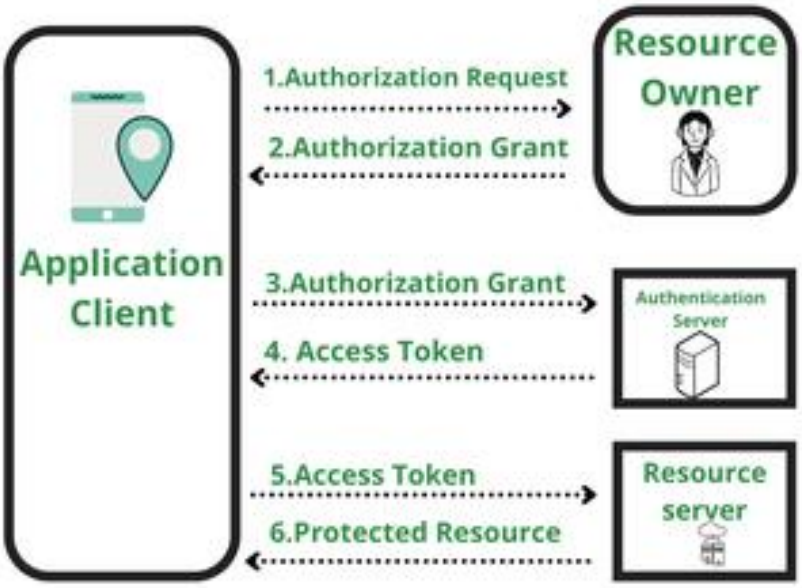

Fig 3: Client-Server Intrface Steps Using API Key Authentication[7]

Once the BSP had an approved application on the server side, the client would use credentials from this application to get authenticated with the server.

- This would first require an authorization for with the resource owner, meaning the BSP would have to authorize the use of credentials by the BSS provider, the client.
- Using this authorization, the client could proceed to get authenticated by the authentication server, the Calix platform, which would administer an access token for the client.

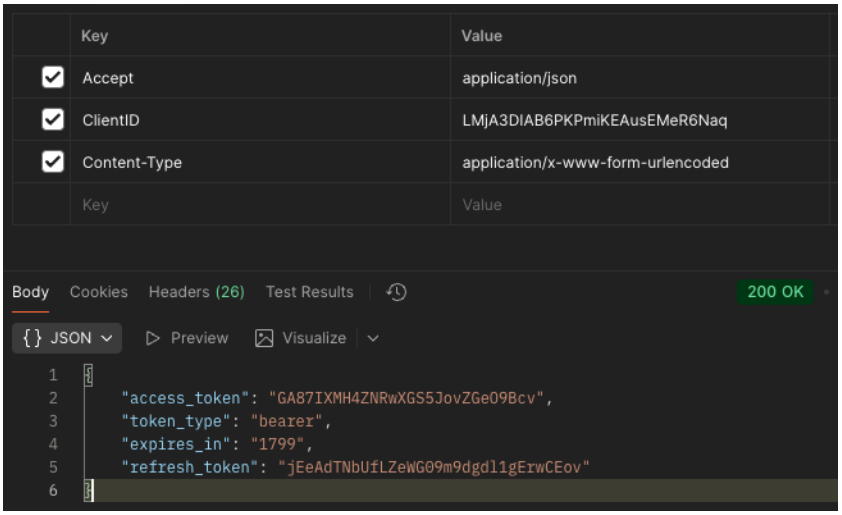

Fig 4: Json Response For With Access Token For Client Authentication

Access tokens are short lived and expire very quickly, offering robust security unlike the use of API keys which are static in nature and can be stored and re-used over longer periods of time.

The client could then request protected resources from the resource server by presenting the access token. The server returns requested data, the API endpoints and associated BSP data upon query.

6. Broadband Data on SaaS

BSPs manage extensive databases that host thousands or millions of subscriber accounts, encompassing sensitive information and subscription data. Within current cybersecurity environments, a single security vulnerability or operational oversight could cause significant service outages and critical data breaches. Therefore, the scope of required security protocols had expanded beyond the protection of just PII.

In the world of Broadband subscribers, the comprehensive security model integrated a variety of infrastructure components:

- Network Equipment & Infrastructure: Core networking hardware required robust security safeguards.
- Customer Premises Equipment (CPE) & IoT Devices: These edge devices represented potential attack vectors and needed to be secured at the endpoint.
- IP Address Management & Associated Services: The management of network identifiers and integral managed services, such as sophisticated home security systems and advanced firewalls, were pivotal elements of the overall security architecture.

The enforcement of rigorous authorization and access control mechanisms for all parties accessing resources on a cloud-native SaaS platform, had been a foundational principle in product development and market adoption.

Historically, an Internet Service Provider's operational framework was bifurcated into two distinct, parallel domains:

- Network Operations & Hardware Management: The physical layer and its associated maintenance and operational procedures.
- Billing & Subscriber Data Management: The business and financial information systems concerning end-users.

The journey of early API usage with Basic Auth in Broadband SaaS products that has ramped to the use of Oauth2.0 can be captured with a comparison based on key aspects:

1. Primary Function:
   a. BasicAuth: Authentication to the extent of identifying a user/system
   b. Oauth2.0: Authorization with delegated access to resources on behalf of a user.

2. Mechanism:
   a. BasicAuth: Sends username and password (base64 encoded) with every request.
   b. OAuth2.0: Uses short-lived access tokens obtained through specific authorization flows.

3. Security:
   a. BasicAuth: Low; credentials are not encrypted and are prone to interception without HTTPS
   b. OAuth2.0: High; tokens have limited scope and expiration times, reducing the risk of a breach.
4. Token:
   a. BasicAuth: No tokens; credentials are valid until the password is changed.
   b. OAuth2.0: Tokens are short-lived and can be refreshed, enhancing security.

5. Use case:
   a. BasicAuth: Legacy systems or low-security internal systems where simplicity is priority.
   b. OAuth2.0: Web integrations, mobile applications, or any applications requiring secure, delegated access to user data.

7. Conclusion

API security patterns are best practices and solutions to protect APIs from various security threats. They ensure that the data transmitted to and from the API is safeguarded and that only authorized users can access it. Key security patterns include using tokens for authentication, such as JWT (JSON Web Tokens), implementing OAuth for secure delegated access, employing HTTPS to encrypt data in transit, and applying throttling and rate limiting to prevent abuse. Input validation, consistent logging, and regular security audits are also integral to maintaining the integrity and confidentiality of API endpoints [8]. The advent of modern SaaS platforms, architected specifically to facilitate the seamless integration of these operational paradigms, allows BSPs to pursue enhanced objectives in data governance, customer experience optimization, and market expansion. The telecommunications industry has undergone significant evolution in network technology and the subsequent integration of subscriber data, establishing a highly specialized niche domain. Consequently, securityparticularly API security, which was foundational to all data integration processeswas escalated to become a top priority for platform providers seeking to mitigate advanced persistent threats (APTs) and ensure data integrity.

References

[1] James Gough, Daniel Bryant and Matthew Auburn, “Mastering API Architecture” in O'Reilly Media, Inc., 2022

[2] An Introduction to APIs By Brian Cooksey - 2014 - cdn.zapier.com

[3] https://testfully.io/blog/api-authentication/

[4] Accessing Cloud through API in a More Secure and Usable Way HongQian Karen Lu Gemalto, Inc. Austin, Texas, U.S.A

[5] Securing the Digital Backbone: An In-depth Insights into API Security Patterns and Practices - Mayank Hindka - Computer Information Systems, Texas A&M University-Central Texas, United States

[6] OAuth 2.0: A Framework to Secure the OAuth-Based Service for Packaged Web Application, January 2020

[7] https://www.geeksforgeeks.org/software-engineering/workflow-of-oauth-2-0/#

[8] Securing the Digital Backbone: An In-depth Insights into API Security Patterns and Practices Mayank Hindka Computer Information Systems, Texas A&M University-Central Texas, United States