Authentication and Authorization Based on OSS for Secure System Interoperation

OAuth 2.0 is a common standard for authenticating APIs used for system-to-system access. However, since OAuth 2.0 only defines a framework, it requires support for the latest standards and access to existing authentication information is also required. So, choosing the right software is an important requirement, generating interest in OSS for authentication applications. Hitachi has released OSS-based API management solutions and made builds easier by defining standardized sequences for access to existing authentication information. The company has also played a role in the development work being done by the OSS community, providing pioneering support for the latest standards.

Kazufumi Enomoto
Yuichi Nakamura, Ph.D.

1. Introduction

In the past, information systems usually had monolithic configurations that gathered all the required functions together in one location. Meanwhile, the recent rise of digital transformation (DX) is creating interest in system-of-systems configurations that provide new services by accessing multiple independent systems over a network. This system-to-system access is usually not enclosed within a single company or organization. Instead, access across multiple companies or organizations has become commonplace. Representational state transfer application programming interfaces (REST APIs; 'APIs') have become a widely used interface format for system-to-system access due to their ease of access. These APIs are publicly released on the Internet to promote system-to-system access. However, security problems have also arisen as APIs have ended up becoming entry points for attacks on systems. Improper API use has been responsible for a series of security incidents in Japan and other countries, so API security mechanisms that release APIs only to the systems that need them and protect the systems from attacks are a crucial requirement.

This article presents open source software (OSS) authentication and authorization technologies used for API security, and Hitachi's work in this area.

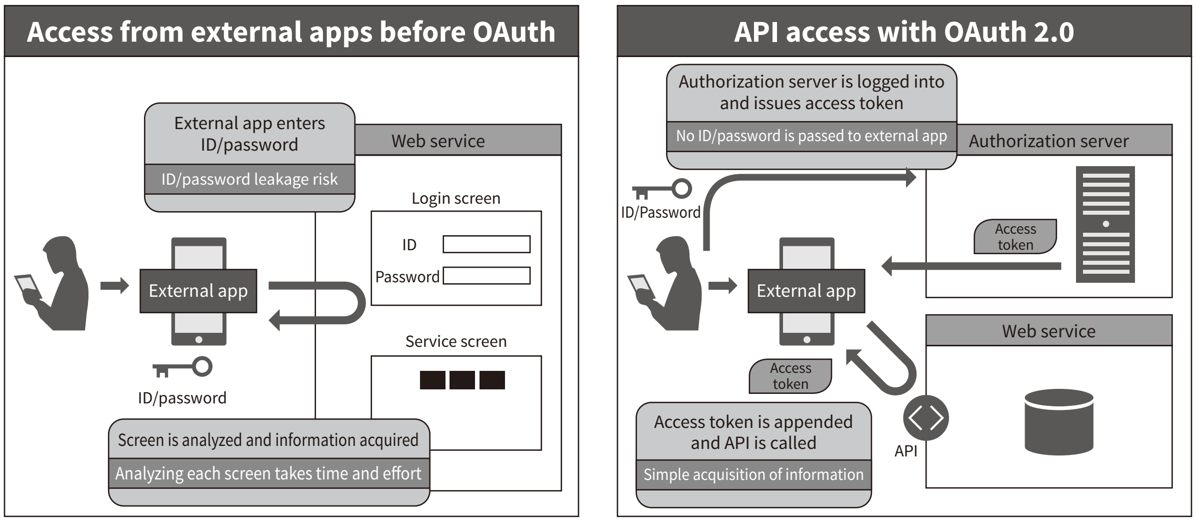
2. API Security Issues

2. 1 OAuth 2.0 Standard

Authentication and authorization done to restrict applications with API access and the users who can use them are the most basic security operations used with API releases. The OAuth 2.0 ('OAuth') standard defined by Internet Engineering Task Force (IETF) RFC 6749 is a common standard used as a framework for API authentication and authorization. The overview in **Figure 1** illustrates how external app access was handled before OAuth existed and how API access is now handled with OAuth. To enable access between an external app and service provider in the

Figure 1 — Overview Illustrating External App Access Before OAuth, and API Access with OAuth 2.0

Security risks were high before OAuth existed since authentication information was held in the external app seeking access. OAuth 2.0 enables more secure access by using access tokens to eliminate the need for passing authentication information to the external app.


Access from external apps before OAuth
External app enters ID/password
ID/password leakage risk
Web service
Login screen
ID
Password
External app
Service screen
ID/password
Screen is analyzed and information acquired
Analyzing each screen takes time and effort
API access with OAuth 2.0
Authorization server is logged into and issues access token
No ID/password is passed to external app
Authorization server
ID/Password
Access token
External app
Web service
Access token
Access token is appended and API is called
Simple acquisition of information
API

API: application programming interface

days before OAuth, a service ID and password were usually passed to the external app, and the app was made to log into the service. Instead of an ID and password, OAuth provides security by passing a package of authorization information known as an access token to the external app. The flow is as follows: First, the external app sends a request to an authorization server to issue an access token. The authorization server authenticates the user and issues the access token to the external app if authentication succeeds. The external app then accesses the API with the access token appended. These processes do not involve passing authentication information such as an ID or password to the external app, enabling more secure access. OAuth has therefore become the de facto standard for granting external apps access to APIs.

2. 2 Issues when Using OAuth in Systems

OAuth is just the framework used for API authentication and authorization. The methods used to implement processes conforming to this framework are largely left up to the developer. As a result, processes satisfying the OAuth standard can still be implemented improperly, creating the risk of security incidents.

Figure 2 shows a typical API-based service configuration example and the security issues to consider.

Issue 1: Improper OAuth implementation

Incorrect implementation of parameter usage in the OAuth flow is known to result in the creation of vulnerabilities that pass access tokens to attackers. New standards are routinely created to eliminate vulnerability gaps and need to be tracked. For example, a new technical specification that is backward-compatible with OAuth, known as Financial-grade API (FAPI) has recently been created and is attracting interest. FAPI is designed for use in systems requiring higher security such as financial institution business systems.

Issue 2: Improper user authentication

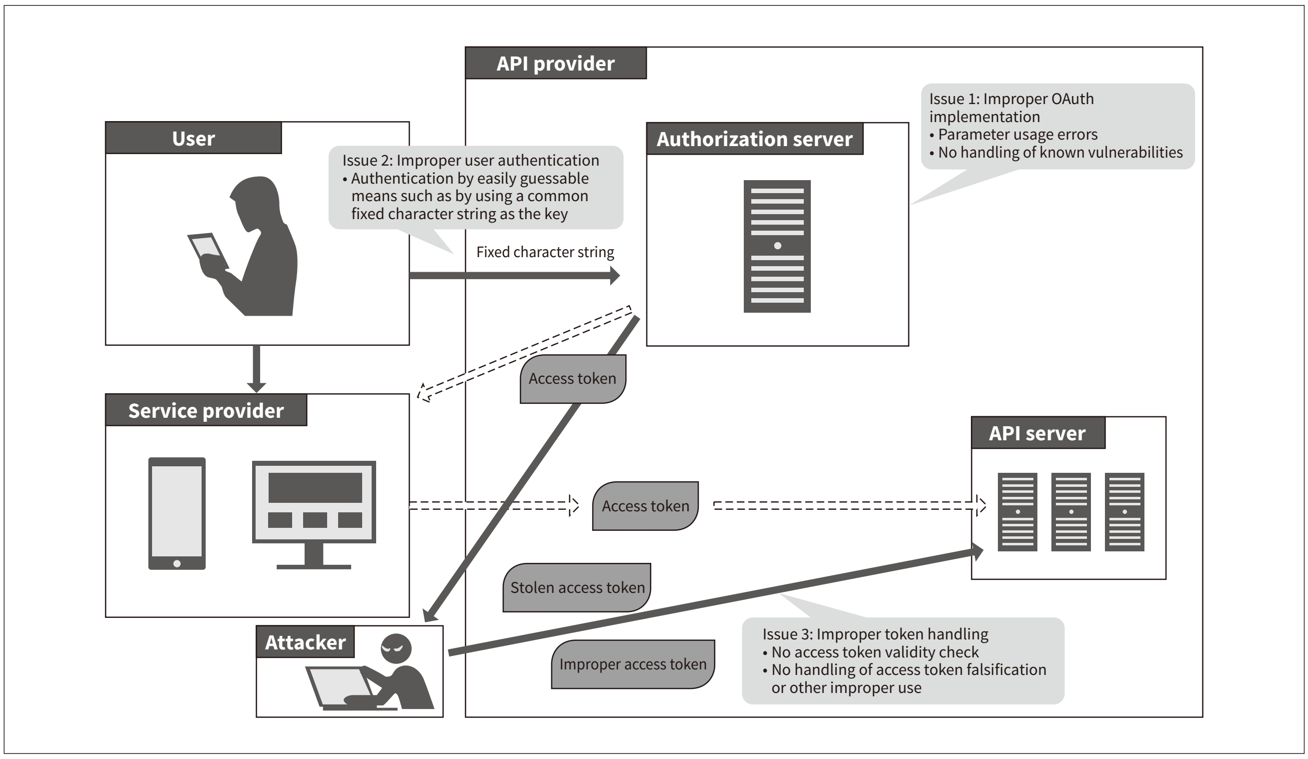
Since OAuth leaves the authentication method up to the developer, there are sometimes extreme cases in which authentication is handled using just an easily guessable character string. Attackers can easily obtain access tokens in this case, enabling improper API calls.

Issue 3: Improper token handling

The method used to handle access tokens issued by the authorization server is not defined, so problems can sometimes result. For example, failure to manage access token expiration can lead to the termination of API access being permanently disabled. Access tokens received by API servers can also be counterfeited or belong to other users, so a separate framework is needed to check for access token falsification and check which user each token was issued to. Failure to carry out these checks can lead to unauthorized APIs being called by attackers.

API-based services implemented without taking the points above into consideration are a continual source of security incidents, so developers are paying closer attention to security.

Figure 2 — Typical OAuth 2.0 Service Configuration Example and Security Issues to Be Considered
OAuth 2.0 only defines the API authorization framework. The methods used to implement authorization are left largely up to the developer, resulting in various security issues and the need for proper implementation when OAuth is used with systems.

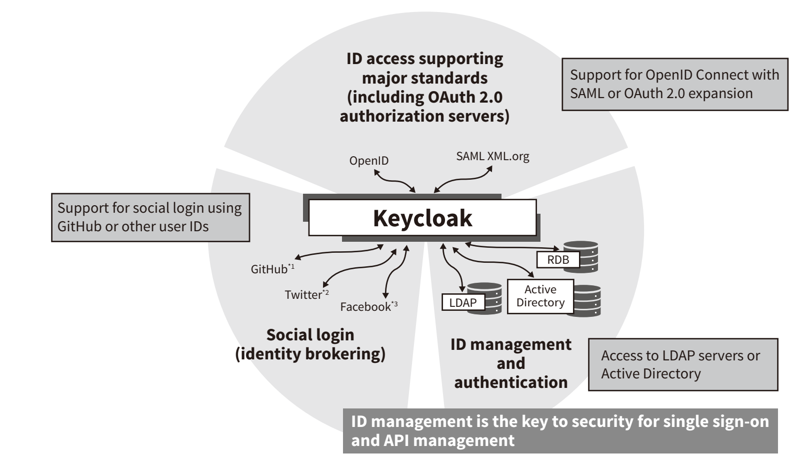

API provider
User
Issue 2: Improper user authentication
• Authentication by easily guessable means such as by using a common fixed character string as the key
Authorization server
Issue 1: Improper OAuth implementation
• Parameter usage errors
• No handling of known vulnerabilities
Fixed character string
Access token
Service provider
Access token
API server
Stolen access token
Attacker
Improper access token
Issue 3: Improper token handling
• No access token validity check
• No handling of access token falsification or other improper use

3. Hitachi's Work on API Authentication and Authorization

3. 1 Keycloak Authorization Servers

The authorization server in charge of issuing and managing access tokens is a very important element when implementing OAuth-based authentication and authorization. Keycloak is an OSS application that has rapidly been gaining a following in this area recently (see **Figure 3**). Keycloak was originally created as an authentication server implementation designed to enable single sign-on conforming to Security Assertion Markup Language (SAML) and other standards. It also has OAuth-compatible API authorization server functions. Keycloak not only supports a wide range of authentication and authorization server functions,

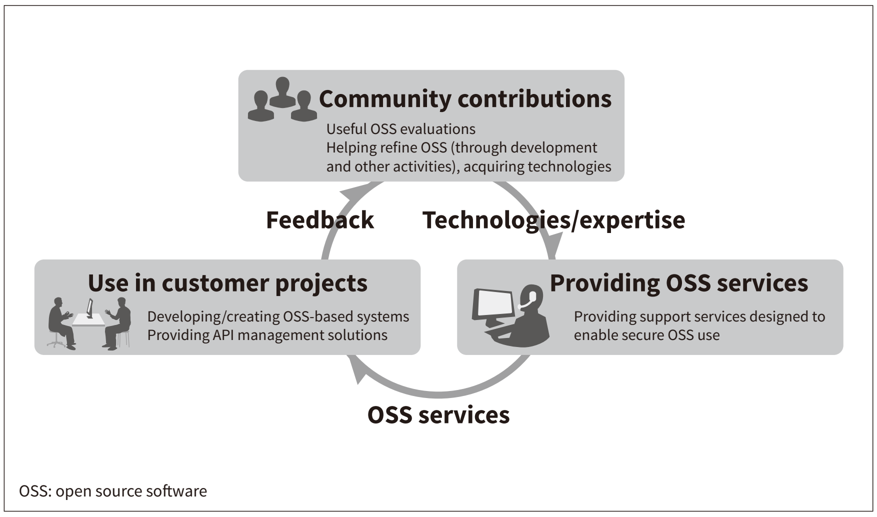
Figure 3 — Keycloak Features
Keycloak has functions enabling use as a single sign-on authentication server and authorization server for API access. Other features include support for OAuth 2.0 and other standards, social login, and access to LDAP servers or other services.


ID access supporting major standards (including OAuth 2.0 authorization servers)
Support for OpenID Connect with SAML or OAuth 2.0 expansion
OpenID
SAML XML.org
Support for social login using GitHub or other user IDs
Keycloak
RDB
GitHub*1
Twitter*2
Facebook*3
LDAP
Active Directory
Social login (identity brokering)
ID management and authentication
Access to LDAP servers or Active Directory
ID management is the key to security for single sign-on and API management

SAML: Security Assertion Markup Language XML: Extensible Markup Language
LDAP: Lightweight Directory Access Protocol RDB: relational database
*1 GitHub is a trademark or registered trademark of GitHub, Inc.
*2 Twitter is a trademark or registered trademark of Twitter, Inc.
*3 Facebook is a trademark or registered trademark of Facebook, Inc.

Figure 4 — Hitachi's Contributions to the Keycloak Community and Their Use in Client Projects

Hitachi's contributions to the Keycloak community have generated knowledge used to provide support services and assist client projects. A cycle of feedback to the community has also enabled it to refine Keycloak and provide services of higher value.


Community contributions
Useful OSS evaluations
Helping refine OSS (through development and other activities), acquiring technologies
Feedback
Technologies/expertise
Use in customer projects
Developing/creating OSS-based systems
Providing API management solutions
Providing OSS services
Providing support services designed to enable secure OSS use
OSS services
OSS: open source software

but it also has a large number of expansion points, enabling easy customization for individual system requirements.

Since it is OSS, Keycloak offers the major benefit of having a development community that is open to anyone. It has an active development community that routinely develops functions conforming to new standards. Hitachi plays a role in this community, and has assisted extensively in developing major functions and making revisions to fix bugs and improve convenience and other features. The in-depth technical knowledge gained from these community contributions has been used to provide a support service and a design/build assistance service for Red Hat* Single Sign On, the commercial version of Keycloak. Hitachi has also used Keycloak for customer projects, creating patches to solve issues and meet demands for additional functions that have arisen as a result. These patches have been posted to the development community and included in Keycloak. Hitachi's activities have helped make Keycloak better suited to the demands of the company's clients (see **Figure 4**).

3. 2 API Management Solutions

Hitachi provides API management solutions that enable secure API access using a core of Keycloak-centered OSS authentication and authorization technologies. **Figure 5** shows the system configuration for these solutions.

(1) Support for OAuth and latest standards

Hitachi's API management solutions use Keycloak as the authorization server and support OAuth. To support the latest FAPI standard, Hitachi plays a leading role in Keycloak community development activities to ensure that basic FAPI function requirements can be satisfied(1).

(2) Ensuring proper user authentication

Increasing user authentication strength requires ID/password authentication and multi-factor authentication. Hitachi's solutions can provide multi-factor authentication using standard Keycloak one-time password authentication. The company also plays a leading role in community development activities to support WebAuthn, the latest multi-factor authentication standard(2).

Hitachi's customer projects have a high demand for authentication using authentication information managed by existing systems. Since the use of existing systems involves dealing with individual system requirements, Hitachi's API management solutions cannot be implemented with only the standard functions, they need to incorporate Keycloak expansions. Hitachi has created templates for typical expansion sequences, enabling rapid use of existing authentication information by using the applicable template.

(3) Proper token handling

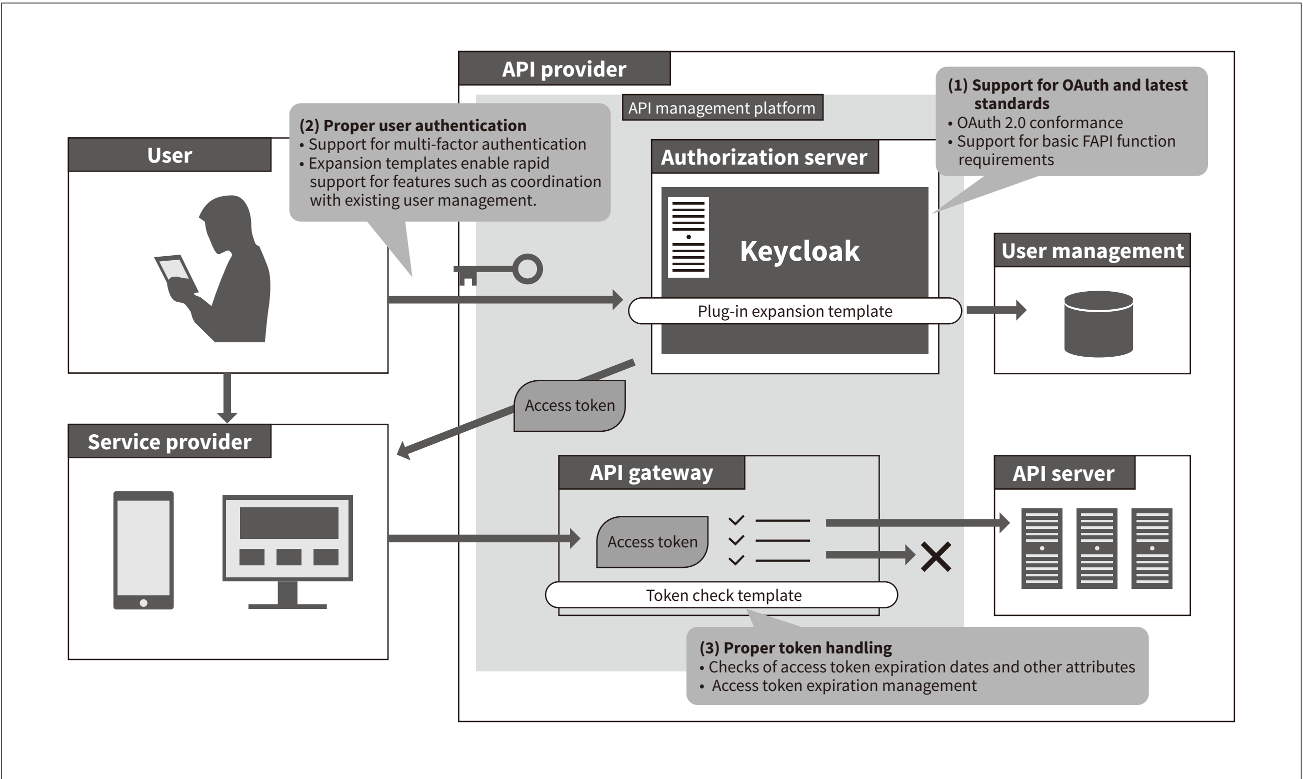
As described for Issue 3 in Section 2.2, token expiration needs to be managed and tokens need to be checked when APIs are called. Hitachi's API management solutions can handle token expiration management using standard Keycloak functions. They check tokens when APIs are called using the API gateway placed in front of the API server. API gateways are elements that control API flow rates and access. They are implemented using various types of software. Hitachi's solutions can guarantee that only valid access tokens are passed to the API server by using its API gateway to detect access token falsification and check expiration dates. An expansion in the API gateway is often needed for the access token check process it uses. Hitachi has created templates for the major API gateways to handle these expansions.

To ensure secure API access, Hitachi uses the configuration described here as the foundation for providing solutions that cover everything from upstream consulting to system construction, operation, and maintenance. Hitachi has a track record of creating API management platforms that ensure API security, and the company's API management solutions have been used in various projects for digital

* Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and other countries.

Figure 5 — Example Configuration of Keycloak-centered Secure API Management System

Shown here is an example configuration for a system that manages APIs securely using Keycloak as the authorization server. The system solves the common security issues using Keycloak and an API gateway.


API provider
API management platform
(2) Proper user authentication
• Support for multi-factor authentication
• Expansion templates enable rapid support for features such as coordination with existing user management.
User
Authorization server
(1) Support for OAuth and latest standards
• OAuth 2.0 conformance
• Support for basic FAPI function requirements
Keycloak
User management
Plug-in expansion template
Access token
Service provider
API gateway
API server
Access token
Token check template
(3) Proper token handling
• Checks of access token expiration dates and other attributes
• Access token expiration management

FAPI: financial-grade API

services. Examples include projects for cashless services provided by financial institutions, and for mobility as a service (MaaS) in the rail transport sector.

4. Conclusions

This article has presented OSS authentication and authorization technologies used for API security, and Hitachi's work in this area. The rise of DX is seeing API-based system-to-system access growing at a faster pace and adding to the importance of API security. Hitachi wants to provide and expand on solutions designed to make it easier to quickly create systems that ensure API security. These solutions draw on the technology and expertise assets that Hitachi has gained by developing features to conform with the latest standards and making other contributions to the OSS community.

References

1) T. Norimatsu, "What are Protected and Secured by Security Requirements for APIs Providing Financial Services," APIdays Paris 2019 (Dec. 2019), https://www.slideshare.net/ssuserbeb7c0/apidays-paris-2019-financialgrade-api-securityprofile
2) T. Norimatsu, "WebAuthn Support for Keycloak," DevConf.cz 2020 (Jan. 2020), https://static.sched.com/hosted_files/devconfcz2020a/78/webauthn-support-keycloak.pdf

Authors

Kazufumi Enomoto
OSS Solution Center, Software CoE, Services & Platforms Business Unit, Hitachi, Ltd. *Current work and research:* Development of API management solutions using OSS.

Yuichi Nakamura, Ph.D.
OSS Solution Center, Software CoE, Services & Platforms Business Unit, Hitachi, Ltd. *Current work and research:* Development of API management solutions using OSS. *Society memberships*: The Information Processing Society of Japan (IPSJ).