HOW TO EXTEND IDENTITY SECURITY TO YOUR API’S
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>EXECUTIVE OVERVIEW</td>
</tr>
<tr>
<td>04</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>05</td>
<td>OAUTH 2.0 – THE PRESENT AND FUTURE CHOICE FOR API PROTECTION</td>
</tr>
<tr>
<td></td>
<td>How OAuth 2.0 Works</td>
</tr>
<tr>
<td>08</td>
<td>SAML – THE GOLD STANDARD FOR IDENTITY FEDERATION</td>
</tr>
<tr>
<td></td>
<td>How SAML and OAuth 2.0 Work Together</td>
</tr>
<tr>
<td>10</td>
<td>OPENID CONNECT – CONVERGENCE IN IDENTITY STANDARDS</td>
</tr>
<tr>
<td></td>
<td>How Connect Works</td>
</tr>
<tr>
<td>11</td>
<td>CONCLUSION</td>
</tr>
<tr>
<td>12</td>
<td>SECURING API’S WITH PING IDENTITY</td>
</tr>
</tbody>
</table>
The number of users and devices requesting access to applications is growing exponentially, and enterprises are scrambling to adapt their authentication and authorization infrastructure to deal with this fast growth.

REST APIs are a critical new application access channel. In a sense, REST APIs are an evolution of the SOAP-based web services of a few years ago. But because they’re much more practical for mobile clients (like native iOS and Android apps), the challenge of scaling more than SOAP web services is compounded. A REST API may be called by millions of clients, which is far more than a SOAP web service.

But early deployments of REST APIs were often authenticated by passwords being attached to the client calls. For instance, if the calls were being made on behalf of a particular user (to obtain calendar data as an example) then the user would be asked to share with the API client the password for their account at the calendar provider. This model of authentication has significant privacy and security issues, including:

- It's very hard for the user to subsequently turn off the access for a given client if they want. Doing so requires changing their password. But doing that will cancel the access of all other clients—even those the user wants to maintain.
- For mobile native application clients, the password-based model required that these passwords be stored on the mobile device so that they could be used on API calls.
- It's difficult for the user to have granular control over such API access (i.e. allow a client to be able to read the calendar data but not be allowed to create entries).

Consequently, it was recognized that new authentication and authorization mechanisms were required for REST APIs. Unfortunately the standards that had been developed to secure SOAP web services (like WS-Security, WS_Trust, WS-SecureConversation, etc.), are incompatible with the lightweight principles of REST.

New security specifications, such as OAuth and OpenID Connect, have emerged to provide the necessary authentication and authorization standards for REST APIs. This paper discusses what these standards and specifications are, how they can be composed with existing standards like SAML, and how they work to provide identity and access management and API security now and into the future.
INTRODUCTION

“Whether it’s creation of an authentication ceremony, definition and enforcement of policy, enforcement of those policies at APIs or code samples for mobile app writers, the requirements necessary to perform true IAM encompass numerous software and service entities across numerous domains.”

- Forrester®

Everything on the Internet, from people to devices, has an identity. The number of Internet users and devices is rapidly increasing—causing ‘identity’ to explode accordingly. IT professionals are scrambling to support the new masses of identities forming without jeopardizing security.

The high volume of corporate and personal devices coming into the enterprise (often referred to as BYOD) accounts for a significant portion of the identity explosion. Enterprises are learning that the systems managing identities are not meeting the needs of the business. Simply put, current identity and application security practices can’t grow to address the demands placed on them.

The demand for scaling access to all these people, devices and things has driven the emergence of APIs. APIs make some parts of an application’s internal logic, functions and data available to outside clients in a constrained fashion. APIs thereby allow that application’s functionality and value to be shared among all the multiple clients of that API. APIs enable “mash ups,” in which a variety of different applications, made available through APIs, can be mixed and matched to create completely new services and applications.

The scale in numbers of clients that REST APIs enable demands security mechanisms that can grow accordingly. Passwords don’t scale in the necessary manner; token-based systems (like those described in this paper) do.

Mobile and BYOD adoption have in large part driven the API revolution, and “headless” access to server-side functionality is fast becoming the norm. Recent web app frameworks we’ve seen are really just simplified API clients. In this context, web forms that force a classic username/password login method and accept only local account logins are becoming impediments—and traditional web access management (WAM) systems are suffering under a similar burden. -Forrester²

An important category of API clients is public or private mobile native apps (i.e., those downloaded from an app store and installed onto a mobile device). To secure both identities and APIs, security practices need to quickly evolve to incorporate modern identity standards such as OAuth 2.0 and OpenID Connect 1.0.
"Using IP addresses, or device IDs and their reputation, no longer sufficiently protects against threats because these parameters mainly affect only the first step in user interactions: front-door authentication. Once the user is logged in, they offer little protection."

- Forrester

OAuth 2.0 is an open standard protocol for authorization that replaces usernames and passwords with access tokens used on API calls. This eliminates the need for user credentials being presented to APIs, which increases identity and API security.

OAuth 2.0 is the industry-leading standard for controlling authentication and authorization of APIs. OAuth 2.0 defines a standardized mechanism that allows an application to access application data or user information from another web service without revealing the identity (i.e., username and password or passing credentials) of the user through the transfer.

There are multiple distinct benefits of OAuth 2.0:

- It provides an explicit mechanism for obtaining a user’s consent to subsequent API requests—particularly relevant for consumer applications, but also possibly in enterprise.
- It defines multiple methods for acquiring an access token, differentiated by what are called grant types. Different grant types can be selected based on the level of trust between the API, its client and the user.
- It allows for a granular consent model (i.e., where a user can stipulate that a given client can read their calendar events, but not post to the calendar).
- It leverages HTTP headers—the access token discussed in Figure 1 is typically set as the value of the authorization HTTP header. This means that the message format can be anything, such as SOAP, XML, JSON or form post data.
HOW OAUTH 2.0 WORKS

Within OAuth, there are four different actors: the user, the API client, the resource server (RS) API and the authorization server (AS). For many scenarios, the user owns the information and grants the API client access to that information. The RS provides the API and the AS issues access tokens so that the client can access the API. The client interacts with the API by presenting an access token on its API calls.

Access tokens have a ‘scope’ associated with them. A scope is a capability or permission related to API usage. For example, an activity such as ‘read invoice’ or a role such as ‘manager’ can be specified as a scope. Administrators can then write policies or roles to look at the scopes associated with specified access tokens to determine whether or not a user can do what they’re requesting through an API.

Figure 1. A typical OAuth 2.0 sequence with a mobile native application

Figure 1 shows that the OAuth 2.0 sequence starts with a mobile application (API Client) requesting an access token from an AS. In this example, the mobile application launches the device browser and loads a page at the AS, in which the user can log in. The OAuth AS then asks the user whether or not they will grant the mobile application access to their information protected by the API in question. The user grants access to the mobile application and the AS and then issues an access token to the mobile application, which it stores. From this point on, the mobile application will use the access token to request data from the API.
The API (RS) validates the access token (either directly, via crypto signature verification, or by interacting with the AS) and returns information back to the mobile application—like user identity attributes. This validation process is repeated until the access token expires and needs to be renewed.

OAuth 2.0 is today’s default choice for securing APIs. By leveraging the different grant types within OAuth, enterprises can establish the necessary trust between users, API clients and the APIs.

“Two of the top three types of data breaches identified in our 2013 Forrsights Security Survey involve username/password credentials and the personally identifiable information (PII) often used for out-of-wallet (OOW) security challenge questions because this data gives attackers everything they need to do maximum damage.”

- Forrester

Figure 1 shows that the OAuth 2.0 sequence starts with a mobile application (API client) requesting an access token from an AS. In this example, the mobile application launches the device browser and loads a page at the AS, in which the user can log in. The OAuth AS then asks the user whether or not they will grant the mobile application access to their information protected by the API in question. The user grants access to the mobile application and the AS and then issues an access token to the mobile application, which it stores. From this point on, the mobile application will use the access token to request data from the API.

The API (RS) validates the access token (either directly, via crypto signature verification, or by interacting with the AS) and returns information back to the mobile application—like user identity attributes. This validation process is repeated until the access token expires and needs to be renewed.

OAuth 2.0 is today’s default choice for securing APIs. By leveraging the different grant types within OAuth, enterprises can establish the necessary trust between users, API clients and the APIs.
Security Assertion Markup Language (SAML) is an open XML standard used for the authentication and authorization of data between an identity provider (IdP) and service provider (SP). SAML enables businesses to share identity information across domains. This process is often called federation.

The goal of SAML is to push authentication to where identities are stored. When two organizations want to share information, one takes the role of the IdP and the other takes the role of the SP. The user is authenticated with the IdP and is transferred to the SP to access an application and information. The transfer of identity information is enabled by the SAML protocol.

HOW SAML AND OAUTH 2.0 WORK TOGETHER

When SAML is combined with OAuth 2.0, organizations can bring together both the authentication (federation) and authorization for APIs. This allows an organization’s users to authenticate where the identities are stored, while gaining access to an SP’s APIs.

For example, an organization has established SAML-based authentication (federation) connections with customers or partners for access into a web application. The organization expands the application to add APIs to provide access to data or services. They can then leverage the authenticated (federated) connections to give access to their customers and partners through OAuth 2.0.
As Figure 2 illustrates, the flow of a typical OAuth 2.0 and SAML sequence for a mobile application starts with the request of an access token from the AS. The same AS is also functioning as a federation SP. When the server works as both an authorization and authentication federation server, it takes the request and redirects the mobile browser to an appropriate IdP.

The user is authenticated directly by the IdP. After authentication, the user is redirected back to the authorization and federation server with a SAML assertion. Note: SAML assertions securely carry identity information between the IdP and SP.

The federation server will validate the SAML assertion, extract identity information and work with the AS to issue an access token back to the mobile application. The mobile application then takes the access token and starts to use it to request information from the API.

By leveraging existing SAML federation between partners and tying them into the authorization provided by OAuth 2.0, the existing investment in federation and standards is easily extended into new platforms without requiring separate identity stores, authentication mechanisms or credentials.
Identity standards are converging in a new specification called OpenID Connect (Connect). As defined by the OpenID Foundation:

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows clients to verify the identity of the end user based on the authentication performed by an authorization server, as well as to obtain basic profile information about the end user in an interoperable and REST-like manner. OpenID Connect allows clients of all types, including web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID providers, and session management, when it makes sense for them. - OpenID Foundation

As stated, the goal of Connect is to add an identity layer to OAuth 2.0. In addition, Connect simplifies existing federation specifications. Because of its versatility, Connect promises to unify web and API authentication and access use cases into a single standard.

Logically, Connect combines the API authorization and access provided by OAuth with the identity-driven web single sign-on (SSO) of the SAML framework.

HOW CONNECT WORKS

Connect was developed by extending the OAuth 2.0 base protocol with additional identity features and mechanisms. Connect is a profile of OAuth 2.0 that adds an identity layer and other features that enhance dynamic interoperability. With these additional identity constructs, Connect becomes comparable to SAML in its support for web SSO and attribute lookup.

In Connect, the IdP is called the Connect provider. Functionally, the Connect provider manages authentication in the same manner as a SAML IdP.

The SP is called the relying party in Connect. The relying party trusts the Connect provider to authenticate a user and communicate identity information through an OIDC token. If the relying party doesn’t get enough information about a user to determine whether or not they can access certain parts of an application, the relying party can then go back to the Connect provider and request more information about a user.
How to Extend Identity Security to Your APIs

A benefit of the Connect specification is the concept of dynamic client registration, which greatly simplifies the process of connecting Connect providers with relying parties. Dynamic client registration helps to solve several limitations of the SAML specification, and it automates establishing a connection as well as reduces the maintenance of certificates used to establish trust.

Another innovation of Connect is the identity token (id_token), a JSON Web Token (JWT) that's additional to the access token defined by OAuth. The JWT is a signed and secure container for identity information that is meant to be consumed by the Connect client. The Connect id_token is comparable to a SAML assertion, but using JSON and not XML for its structure. As in normal OAuth, the Connect provider delivers an access token to the client, which is then used to access APIs.

Because Connect delivers both the id_token and the access token, it can be used for web authentication and access as well as API authorization. This functionality offers a simpler, tighter integration than SAML and OAuth while providing the same level of security as well as additional features like dynamic client validation.

CONCLUSION

As devices and APIs proliferate the enterprise, selecting the right security method is critical for protecting the enterprise. Tying identities to API security and access provides robust controls for authentication, access and auditing. Recognizing that, as important as REST APIs are, they deserve a secure, scalable and simple authentication and authorization framework—namely by using OAuth 2.0, SAML and OpenID Connect.

<table>
<thead>
<tr>
<th>WEB SSO</th>
<th>API AUTHORIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAML</td>
<td>Yes</td>
</tr>
<tr>
<td>OAuth 2.0</td>
<td>No</td>
</tr>
<tr>
<td>OpenID Connect</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 3. Standards and the use cases they support

OAuth 2.0 is the industry-leading standard for controlling access to REST APIs and reflects enterprise use cases more so than earlier versions. By leveraging the different grant types within OAuth, enterprises can establish the necessary trust between users, API clients and the APIs.

For enterprises with established, federated connections with customers and partners, the combination of SAML and OAuth 2.0 is beneficial because of the ability to leverage the investment to establish federated connections and provide access to both web applications and APIs.

OpenID Connect is a new specification used to extend OAuth 2.0 with an identity layer. The versatility of the specification promises to unify web and API authentication and access use cases into a single standard.
SECURING API’S WITH PING IDENTITY

Ping Identity offers two products to secure APIs: PingFederate® and PingAccess®.

PingFederate is a standards-based identity bridge that can act as an authorization and federation server. PingFederate understands OAuth 2.0 and SAML independently or layered together, and it had an implementation of OpenID Connect. Therefore, PingFederate provides an opportunity for organizations to experiment with different approaches to secure APIs, and its broad support for all of the protocols gives enterprises maximum flexibility for dealing with different partner capabilities, while protecting the enterprises against market evolution to and from protocols over time.

PingAccess provides the authorization and access management for both web and APIs and uses PingFederate for its authentication and federation capabilities. PingAccess understands SAML, OAuth 2.0 and OpenID Connect. It offers a single place to write policies and share them, and to apply those policies both to web and API resources.

To learn more about securing identities and APIs with PingFederate and PingAccess, dial U.S. toll-free 877.898.2905 or +1.303.468.2882, email sales@pingidentity.com or visit www.pingidentity.com.

1 Forrester, Top Technology Trends To Watch: 2014 To 2016, November 4, 2013, by Brian Hopkins with Leslie Owens, John C. McCarthy, Abigail Komlenic
2,3 Forrester, Predictions 2014: Identity And Access Management, Employee And Customer IAM Head For The Cloud, January 7, 2014, By Andras Cser, Eve Maler with Stephanie Balouaras, Jennie Duong
4 Forrester, Market Overview: Employee And Customer Authentication Solutions In 2013, Part 1 Of 2, Seven Major Trends Every Security Pro Needs To Understand, Including Attempts To “Kill The Password”, December 30, 2013, By Eve Maler, Andras Cser with Stephanie Balouaras, Jennie Duong
5 B2C, Bussines 2 Community, Omni-channel Retail Case Study Reveals Opportunities for CMOs, by Christopher S. Rollyson, March 11, 2014
6 OpenID, Welcome to OpenID Connect, 2014