Authentication Protocol Overview: OAuth2, SAML, LDAP, RADIUS, KerberosU

User authentication in applications is one of the biggest current challenges the IT department is facing. There are a lot of different systems a user needs access to, and that's why most authentication protocols are typically open standards.

When reading questions about authentication protocols on Stack Overflow, it becomes pretty clear that this can be a confusing and overwhelming topic.

In this blog post, we introduce the five most commonly used authentication protocols and explain how they work and their benefits.

LDAP (Lightweight Directory Access Protocol) is a software protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or a corporate intranet.

It is fair to say that LDAP has become a popular program. It served as the foundation on which Microsoft built Active Directory, and has been instrumental in the development of today's cloud-based directories (also known as Directories-as-a-Service).

LDAP sends messages between servers and client applications which can include everything from client requests to data formatting.

On a functional level, LDAP works by binding an LDAP user to an LDAP server. The client sends an operation request that asks for a particular set of information, such as user login credentials or other organizational data. The LDAP server then processes the query based on its internal language, communicates with directory services if needed, and responds. When the client receives the response, it unbinds from the server and processes the data accordingly.

Kerberos

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. A free implementation of this protocol is available from the Massachusetts Institute of Technology. Kerberos is available in many commercial products as well.

Here are the most basic steps taken to authenticate in a Kerberized environment.

1. Client requests an authentication ticket (TGT) from the Key Distribution Center (KDC).
2. The KDC verifies the credentials and sends back an encrypted TGT and session key.
3. Client requests to access an application on a server. A ticket request for the application server gets sent to the KDC which consists of the client's TGT and an authenticator.
4. The KDC returns a ticket and a session key to the user.
5. The ticket is sent to the application server. Once the ticket and authenticator have been received, the server can authenticate the client.
6. The server replies to the client with another authenticator. On receiving this authenticator, the client can authenticate the server.

Oauth 2

OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub, and DigitalOcean.

Here is a description of the basic steps in the authorization process:

1. Application requests authorization for access service resources from the user.
2. If that user approves then the application receives an authorization grant.
3. Application requests an access token from the authorization server (API). This is done by presenting its identity and the authorization grant.
4. If the application identity is authenticated and the authorization grant is valid, the API issues an access token to the application. Authorization is complete.
5. The application requests the resource from the API and presents the access token for authentication.
6. If the access token is valid, the API serves the resource to the application.

SAML

Security Assertion Markup Language (SAML) is an XML-based, open-standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider. SAML is a product of the OASIS Security Services Technical Committee.

JumpCloud is one of the best Single Sign-On (SSO) providers which supports SAML authentication protocols. JumpCloud's SSO provides SAML integrations with 700 popular business applications (including Kisi) and automated user lifecycle management features like Just-in-Time (JIT) provisioning and SCIM provisioning/deprovisioning.

Here is a description of the typical steps in the authentication process:

1. User accesses remote application using a link on an intranet or similar and the application loads.
2. Application identifies user's origin (by application subdomain, user IP address, or similar). It redirects the user back to the identity provider, asking for authentication.
3. User either has an existing active browser session with the identity provider or establishes one by logging into the identity provider.
4. Identity provider builds authentication response in the form of an XML-document containing user's username or email address. This is then signed using an X.509 certificate and then posted to the service provider.
5. Service provider (which already knows the identity provider and has a certificate fingerprint) retrieves authentication response and validates it using certificate fingerprint.
6. The identity of the user is established, and the user is provided with app access.

RADIUS

Remote Authentication Dial-In User Service (RADIUS) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA or Triple A) management for users who connect and use a network service.

RADIUS authentication begins when the user requests access to a network resource through the Remote Access Server (RAS). The user enters a username and a password, which are encrypted by the RADIUS server before being sent through the authentication process.

Then the RADIUS server checks the accuracy of the information by employing authentication schemes to verify the data. This is done by comparing the user-provided information against a locally stored database or referring to external sources such as Active Directory servers.

The RADIUS server will then respond by accepting, challenging or rejecting the user. Individual users may be granted restricted access without affecting other users. In the case of a challenge, the RADIUS server requests additional information from the user to verify their user ID - which may be a PIN or a secondary password. In the case of a reject, the user is unconditionally denied all access to the RADIUS protocol.

So which one to choose?

LDAP, Kerberos, OAuth2, SAML, and RADIUS are all useful for different authorization and authentication purposes and are often used with SSO.

The protocol you choose should reflect your application needs and what existing infrastructure is in place. It helps to choose a simple and standardized solution that avoids the use of workarounds for interoperability with native applications. This is why SAML is a good choice as it integrates with JumpCloud's SSO and 700 popular business applications.

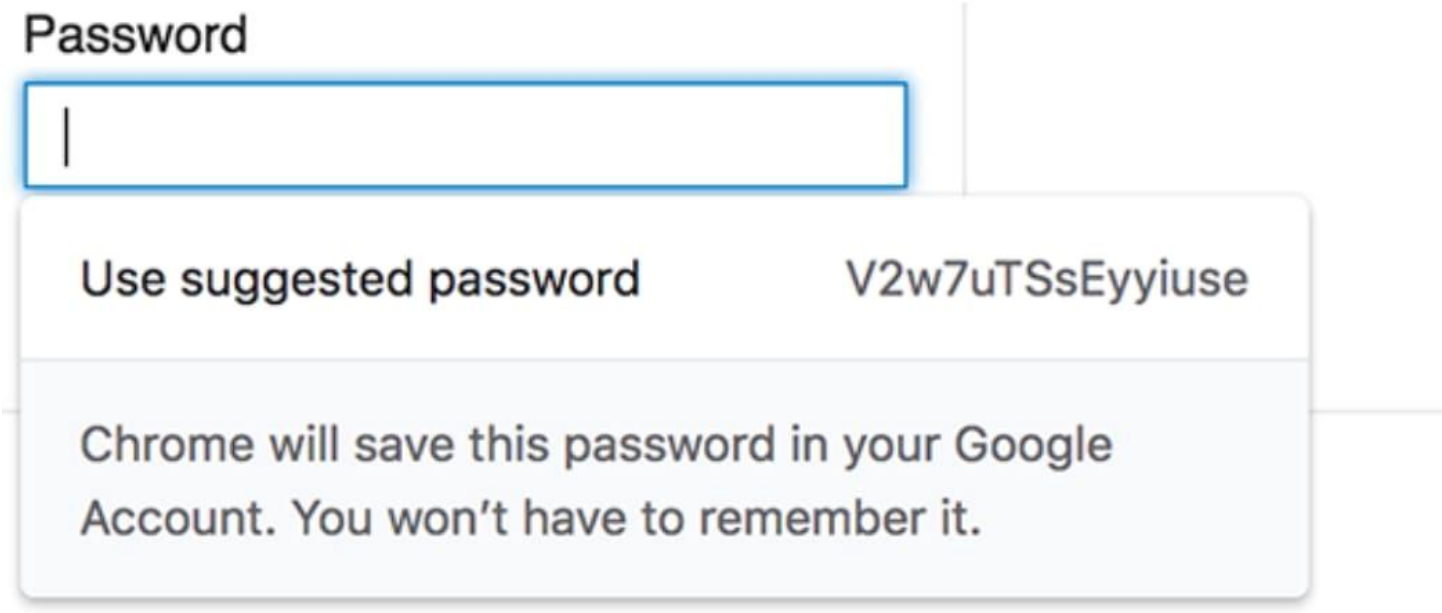
How to Decide What Type of SSO to Use

We'll first take a look at popular alternatives to SSO that sometimes get conflated with it, because they share similar functionality.

Password Managers

Password managers provide a service that is very similar to SSO, at least superficially, but it is a completely different technology on the backend, with more limited functionality.

Password managers like Lastpass or OnePassword are services that generate and store a random and seemingly secure password whenever you create a new account that it is linked to. In other words, if a web or mobile application is integrated with (linked to) your password manager, and you want to create a new account on that app, the password manager will generate a complicated sequence of characters and letters for you, and store that so that you don't have to remember it.



For those who use Google Chrome as a primary browser, you've likely already experienced something similar: When you create an account on a new server or web app, Chrome will generate a long string of numbers and letters and 'suggest' that you use it as your password. Then it will store that information for you, and autofill whenever you try to access that website again. This is exactly the functionality of a password manager — storing all your passwords for you so that you don't have to remember every single one.

How a Password Manager is Different from SSO

A password manager is more of an intermediary between you and the app, insofar as all it does is create and remember your passwords. The functionality is still limited, and is nothing more than a log where you can store strings of characters that represent your passwords. On the other hand, SSO services link your entire account on an app to the provider, and you sign in to the app via the SSO provider. You only really have one account, on the SSO provider, and that handles your identity information on all other apps.

For the casual user, the difference between these can seem minimal — both make it so that you don't have to remember your password for every app, and are themselves quite secure to make your account more resistant to hacks. However, most commercially available password managers aren't fully integrated, meaning that the password won't necessarily be stored in your laptop's cookies. This means that for many websites and apps, you'll have to log in to your password manager, and retrieve the relevant password manually, rather than having it auto-fill.

There's also the difference for enterprises: SSO providers allow you to create a company account, and generate logins for all employees on all linked apps, letting you monitor usage and set restrictions Password managers offer nothing of the sort. All they do is store passwords, plain and simple.

IAM — Identity and Access Management

Another term that you may have heard in the same context as SSO is IAM, or identity and access management. While, again, the two are related, they denote different things, and should not be conflated.

If you've heard the term SSO before, then odds are you've heard that it's a subsection of the overarching IAM umbrella. As the name suggests, IAM is a general security tech and business discipline that encourages granting the right users the right access to the right resources or apps… at the right time. There are several emerging technologies under the sphere of IAM, namely SSO, multi-step authentication in its various forms, and new methods of data governance and analysis. Password managers, too, fall under the umbrella of IAM.

How IAM is different from SSO

IAM is itself under the larger umbrella of general IT security. Narrowing in a bit from IAM, we find IDaaS, a new set of services that comprises all IAM techs that are offered through cloud-based or SaaS apps. IDaaS stands for ID as a Service, and is itself reminiscent, at least in name, of SaaS, or software as a service (note — you will sometimes see it called DaaS, directory as a service, instead — they denote the same idea).

IDaaS is different from more traditional IAM in that it doesn't require large hardware packages (being entirely cloud-based), and relies instead on new Active Directory technology with the emerging LDAP — lightweight directory access protocol (in layman's terms, these are just fancy acronyms for new and secure online protocols for managing identity data). In a recent study, Gartner predicts that while IDaas only comprises 40-50% of the access management market share now, that will rise to around 80% by 2022. It's here to stay. And as you may have guessed, SSO is one of the core functionalities of any IDaaS company.

Basically, the idea is that IAM denotes a whole swath of new and emerging technologies related to identity management and access management. An example of such tech is password managers, which also fall under IAM. SSO also falls under the umbrella of IAM, but it's not strictly the same thing. It's like saying that a golden retriever is a dog, but a dog isn't necessarily a golden retriever.

SAML 2.0

We'll begin with the most common type of SSO authentication: SAML 2.0.

SAML 2.0 stands for "Security Access Markup Language," and is essentially a way to encode text that is used specifically to exchange identification information. SAML is an open standard, meaning anyone can have access to the documentation, and ensure that their code adheres to it. As opposed to a proprietary or closed standard, which would 'belong' to a certain company, and that nobody else would have permission to use.

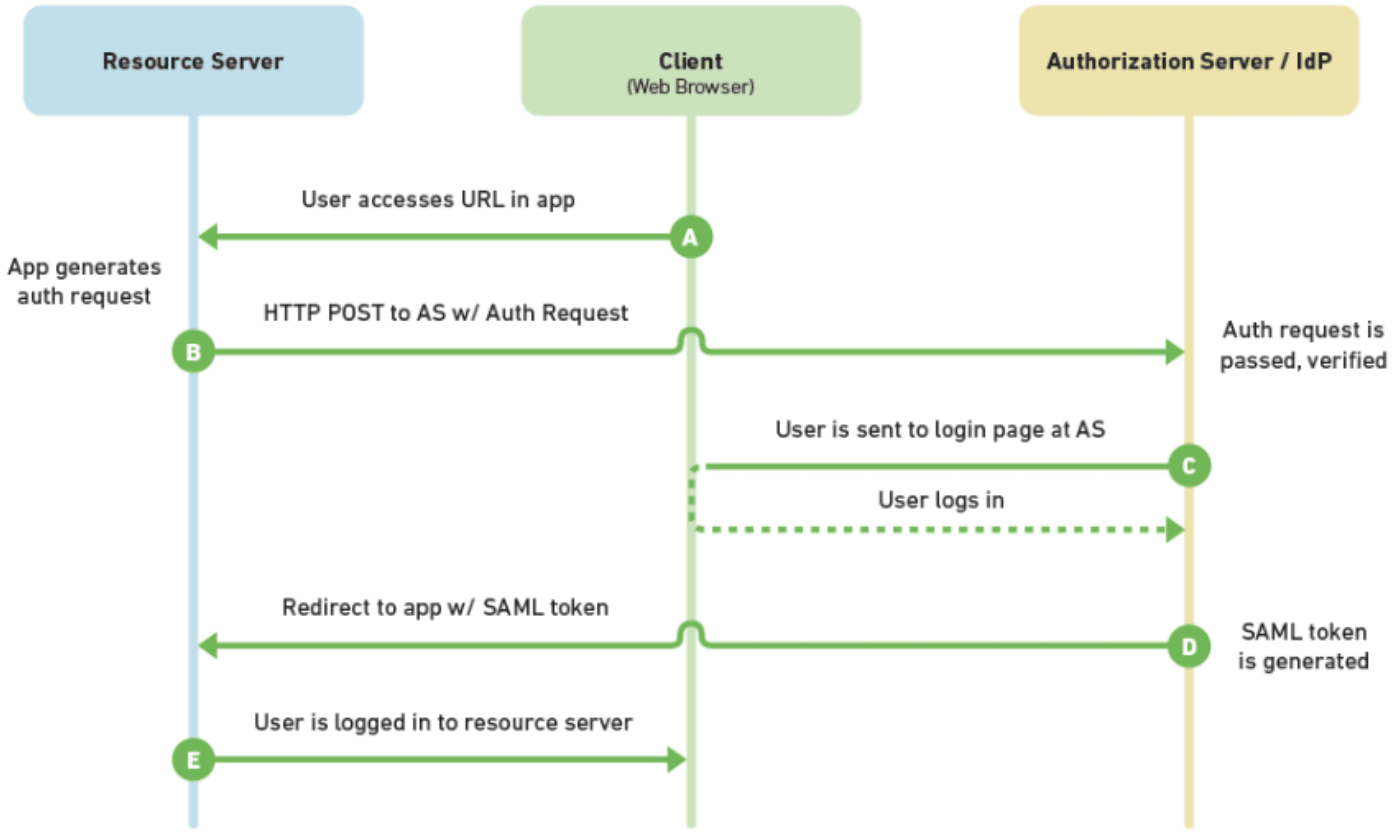
The direct upshot is that this has become one of the primary standards for SSO. The fact that it is an open standard means that any application can make sure that their authentication requests are up to par, and when that is satisfied, they can be enabled with SSO providers, like Okta and OneLogin.

Concretely, a markup language is a wrapper around plaintext that modifies it in such a way as to be readable by computers. The most famous example is HTML, which stands for hypertext markup language. It looks like items in brackets around text, like this: <body>Plaintext</body>. These commands turn the text into machine language, and for SAML 2.0, encrypt it in a universally recognizable and secure way, specifically for identification information.

The last point to note is that SAML 2.0 is optimized for web applications. This means that if you're transmitting information using a web browser, then SAML 2.0 is the right way to go. However, if you're attempting a login in a native app, like an iPhone or Android app, it will not work as well, and you're better off using OAuth2, which we'll discuss next.

How SAML 2.0 is Used for SSO

As mentioned, SAML 2.0 is a standard for encrypting identification information. The following diagram gives an overview of the process. There are three players here: The client (you, or your web browser), the resource server (the app you are trying to access), and the authorization server (the SSO provider). They exchange information in the following way, when you send a new login attempt on your web browser to the resource server.



Basically, the complicated path of arrows denotes that you send the request to login to the app, it then requests the SSO provider for login credentials in SAML 2.0, which then prompts you, the user, to agree to send the information to the app via the provider, and then the ID information is sent to the app in SAML 2.0, and voilà, you're logged in!

OAuth2 SSO

Beyond SAML 2.0, there's also the OAuth2 protocol that can govern SSO. It is largely the same for the end user, but there are some differences.

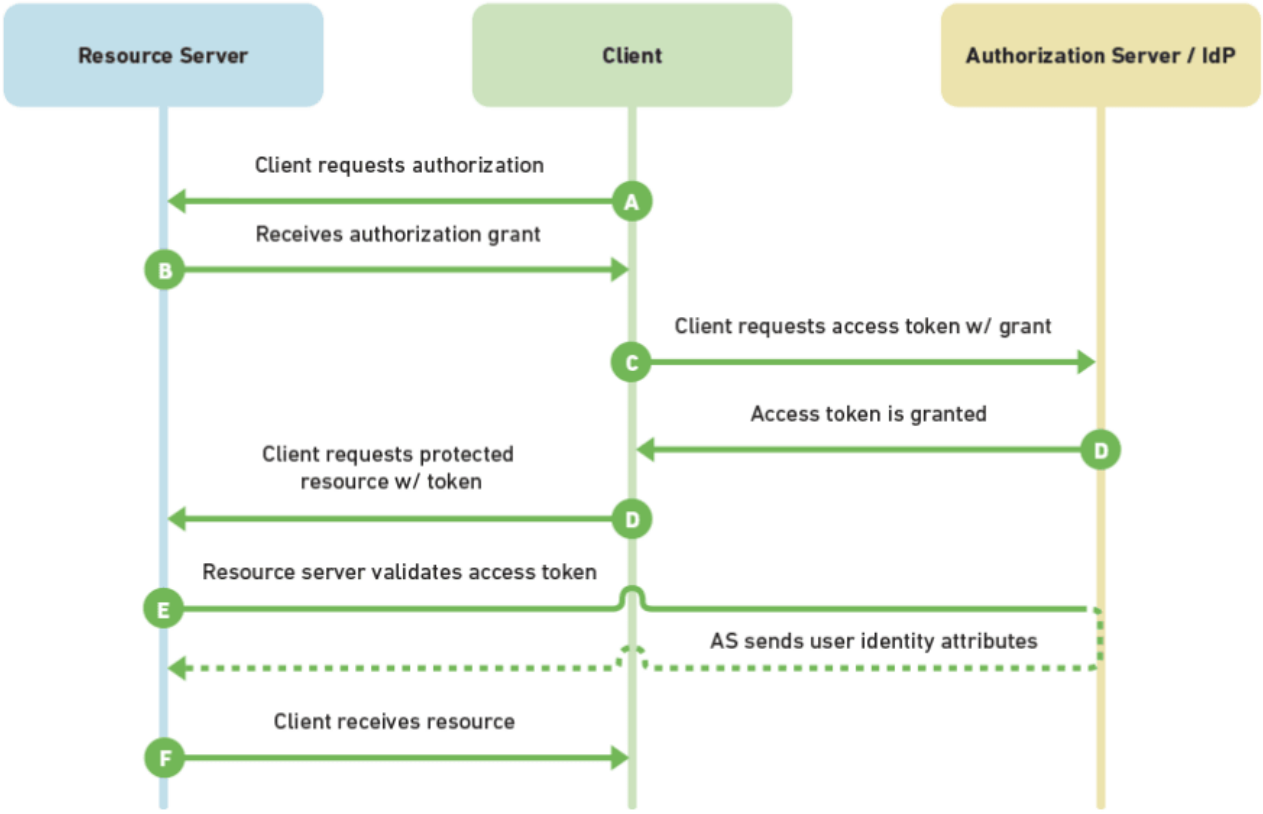
OAuth2 is another protocol governing SSO. As mentioned, its most noticeable difference with SAML is that SAML is optimized for web apps, whereas OAuth is better suited for native apps, like those on a smartphone. Beyond that difference, they work largely in similar ways, and their backend is largely similar. Both are standards for transferring identification information, and both encrypt said information by turning it into machine-only readable code.

OAuth2 is a newer standard, which is why it's more versatile, but it's also less widely used than SAML 2.0, and most of the big SSO providers (Okta, Onelogin) prefer SAML 2.0.

How OAuth2 is Used for SSO

The process of exchanging information is slightly different in OAuth2 as opposed to SAML 2.0, with the main difference being that the client (your browser or app) is called one extra time. You request a login to the relevant app, it sends you a 'grant' to request access from the SSO provider, which in turn gives you an access token, which you pass to the resource server. This is all done to ensure that it is actually you making the

request, rather than a malicious third party. Afterwards, the app communicates directly with the SSO provider in OAuth2 to login.



The diagram above depicts this interaction well. The flow is different from that prescribed by SAML 2.0, but the end result is the same.

Custom-Built Solution

Finally, there is one last option that we'll briefly touch on: Building your own custom authentication solution. We won't spend much time on this because it's not necessarily advisable, but if you have an organization and you don't want to use any of the major SSO providers, it is possible to build your own custom solution. This would entail setting up an authorization server, setting a standard for communicating ID information, making sure that it's secure, and making sure that all apps used by your organization are able to communicate in that standard. For the sake of simplicity, we'd recommend sticking with SAML 2.0 or OAuth2.

Main Takeaways

While there are some lesser alternatives out there, like password managers and other sub-categories of IAM, SSO providers remain the most secure providers of login credentials and ID services. Whether you opt for the more widely used SAML 2.0, or the newer and more versatile OAuth2 standard, will largely depend on what your primary use case is. If you want more information, check out the Gartner Magic Quadrant and Capterra Market Overview of SSO services, both excellent resources for researching further SSO systems, and specific SSO providers. **Source** getkisi.com Andrés Camperi