Secure Access to Legacy Web Applications with Okta

okta

Introduction

Traditional web applications pre-date modern standards like SAML and OpenID Connect, so they often use legacy authentication methods to grant end users access. Historically, providing Single Sign-On and centralized access management for end users across these legacy applications required a Web Access Management, or WAM, product. These on-premise software tools could be expensive to maintain and complex to deploy. With Okta, organizations can protect both on-premise and cloud apps from a single Identity Provider. This white paper describes how web applications that lack modern standard support can be integrated into a cloud-based Identity-as-a-service architecture.

Supporting Legacy Authentication Methods with Okta

At Okta, our customers are the most innovative, forward-leaning, and bold enterprises in their respective businesses. They look to Okta to securely connect their employees, partners, and customers to any technology and Okta is built to manage access to thousands of applications and resources, right out of the box. Regardless of how innovative an organization can be, mature enterprises inevitably have legacy resources that rely on the technologies of the previous generation. Authentication is one area where older models persist; critical business applications use older approaches to authentication which are closed and inefficient. Enterprises need to enable seamless, secure access to every application or resource, so it must be able to support legacy and modern technologies. This whitepaper describes how organizations can address this problem while simplifying the integration architecture.

How We Got Here

In the 1990s, many companies faced a problem: with the proliferation of web applications in the enterprise, end-user access was difficult to manage and a poor user experience inhibited adoption. There was no ubiquitous standard for authentication that worked well for web applications, so many organizations turned to Web Access Management (WAM) solutions like CA SiteMinder, Oracle Access Manager, and IBM Tivoli Access Manager to control authentication and authorization to corporate resources. WAM tools provide single sign-on, centralized policy management and reporting and auditing capabilities for web applications.

In the late 2000s, two things happened: federated authentication standards like Security Assertion Markup Language (SAML) and later OpenID Connect (OIDC) gained popularity, and SaaS, PaaS, and IaaS started gaining traction in the enterprise. This is when Identity-as-a-Service (IDaaS) emerged as an alternate approach, with a cloud-based bridge to the cloud and lightweight directory integration. By leveraging the power of federation standards and the benefits of the cloud service model, IdaaS could provide a great user experience across web applications without the need for expensive infrastructure deployment or maintenance.

This shift continued, and now many organizations are beginning to centralize their Identity and Access Management (IAM) programs around IDaaS, moving the center of gravity of access control to the cloud. At the same time, businesses still depend on legacy applications, so a modern IAM architecture cannot neglect them. Enterprises need to modernize on-premises applications, or implement solutions that enable more direct integration to IDaaS.

At Okta, we offer approaches that enterprises can employ to centralize access control and visibility across legacy and cloud applications and provide a great end user experience, while minimizing on-premises infrastructure.

Web Access Management, or WAM

There are generally considered to be two traditional WAM models: proxy-based, and agent- or plugin-based. The proxy-based approach routes all web traffic through network traffic manager, where HTTP requests can be denied or granted based on policies. This model introduced an additional network component, but it offers protocol-level granular access control without installing any software. With the agent-based approach, agents are installed on each app or web server. These plugins intercept HTTP requests, call out to the centralized policy server, and enforce access rules before responding. This approach removes the need to route all traffic through a proxy, but carries the burden of having to install, update, and manage proprietary agents on every app server in your environment.

By contrast, modern standards like SAML and OIDC use a token-based approach. In this model, an identity provider supplies a token to the application (service provider), such as a JSON Web Token (JWT) or SOAP payload, with information about the user. With SAML, for example, the token is a SAML assertion, a SOAP-based Web Service message, signed by an identity provider, which contains claims about the user that application code can use to make access decisions. The token model uses the end user's encrypted browser context to exchange information between the identity provider (IdP) and the service provider (SP)—that is, the app. The nature of this model eliminates the need for the IdP and the SP to communicate directly, so networking changes are not required, no agents are required, and traffic need not be routed through a proxy. These benefits have contributed to SAML and OIDC's emergence and rise in popularity and traditional WAM models are now falling out of vogue.

However, applications need to be modified to support SAML or OIDC natively. Because modernizing these legacy applications competes with the enterprise's other priorities, that's not going to happen immediately or completely. The result: a disjoint architecture. New applications support modern standards, and older applications do not. IT Administrators must reconcile these two worlds with a single identity architecture to realize the potential of IAM.

Authentication Patterns

If your high-level goal is to manage identity and access across all of your apps, a good starting point is to understand the authentication patterns in use at your organization. At Okta, we've seen that web applications use one of the following methods to authenticate the end user:

Modern methods

Modern methods include:

- **Forms-based Authentication**—This pattern uses a custom page to capture the end user's username and password to authenticate the user. Okta supports Forms-based Authentication natively using our Secure Web Authentication plugin.
- **SAML or WS-Fed-based Federation**—This pattern allows end users to authenticate to an Identity Provider, which issues secure tokens that the end user can use to access other service and applications. Okta supports SAML and WS-Fed natively. You can read more about Okta and SAML on the Okta developer site.
- **OIDC-based Federation**—This pattern is a modern version of SAML. It allows end users to authenticate to service and provides a means to exchange identity information securely across services. Okta supports OIDC natively. You can read more about Okta and OIDC on the Okta developer site.

Legacy Patterns

- **No Authentication**—This is also known as Anonymous Access. In this pattern, anyone can access a site without authenticating first. For web applications intended to be public, this is fine, but sometimes these pages require more security. In these cases, Okta recommends that customers improve security by forcing authentication, and allowing only authenticated users to access the app.
- **Header-Based Authentication**—A web access management system prompts the end user for authentication, then injects identity data into the HTTP Headers in the user's browser for consumption by the protected application. Common WAM systems include CA Siteminder, Oracle Access Manager and Tivoli Access Manager. Okta recommends migrating to a modern proxy-based architecture to accommodate this pattern.
- **Client Certificate-based Authentication**—This pattern utilizes a PKI certificate to authenticate the end user to an application. This is facilitated by most web servers natively, but can also be implemented using a WAM system. If the app cannot be modernized, Okta recommends leveraging a modern proxy-based architecture to accommodate this pattern.

- **Windows Authentication**—This pattern is also called Kerberos authentication (depending on the protocol used). This pattern silently logs the user using the active Windows domain session. This requires domain permissions and only works for internal users by default. Okta recommends integrating with a proxy-based architecture to provide remote access to these applications.
- **URL-based Authorization**—This pattern is complementary to the authentication patterns listed above. In URL authorization, the web access management system evaluates the URL (also known as realm or URI) requested by the end user against an authorization policy before granting access to the resource. The authorization policy contains rules – usually based on groups – to validate if a user should have access to the resource.

Bringing Legacy Apps into the Fold

Now that we understand the different patterns, let's talk about how to approach integrating all of your existing web applications into your IAM platform. We've used the following decision tree with customers and it's worked well. Here's the whole thing, and after the jump we'll step through it.

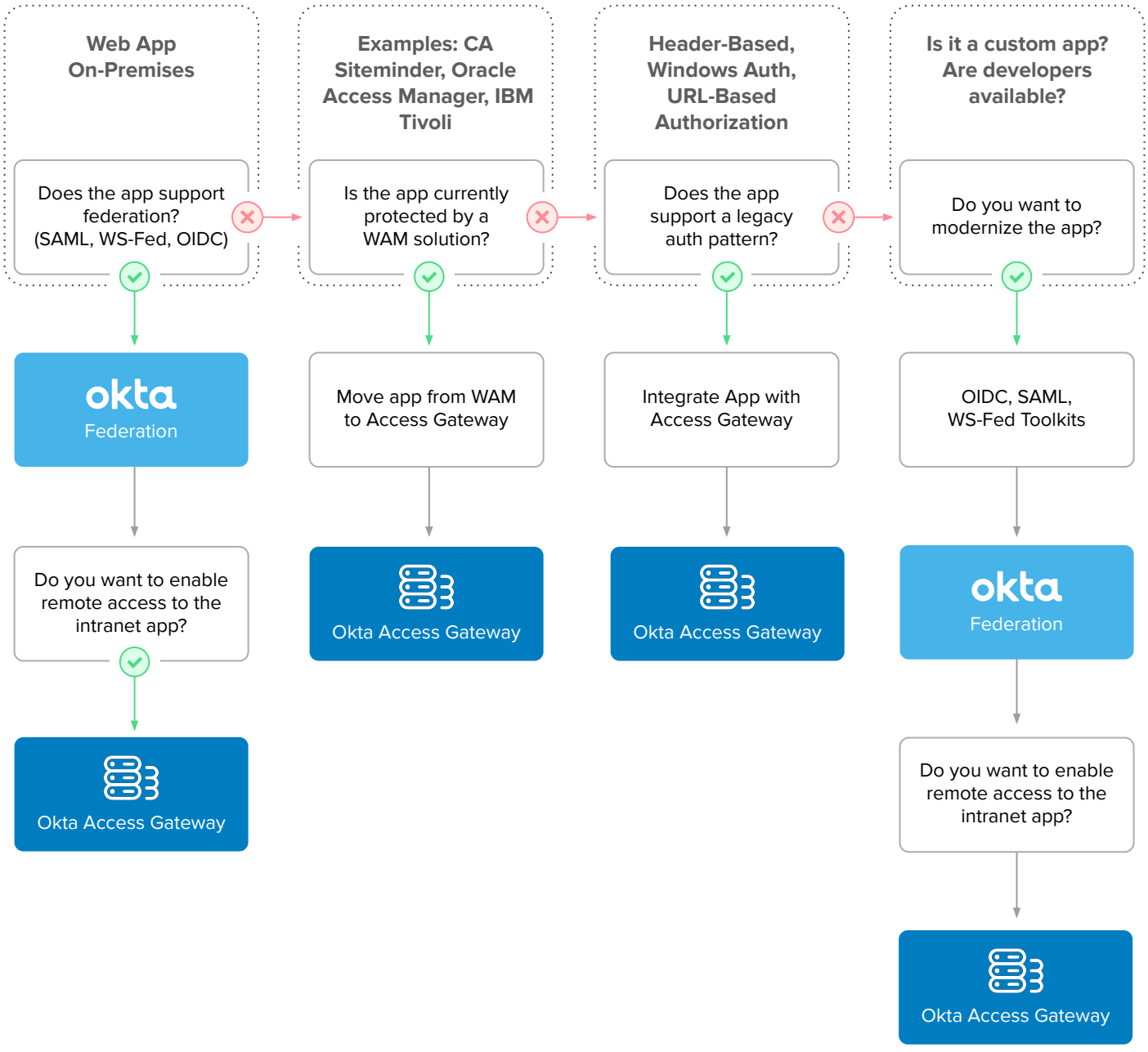

Figure 1. Decision Making Process for Integration Legacy Applications into a Modern Identity Platform

Decision Process—Step-by-Step

1

Does the app already support a modern pattern?

Determine whether the application supports a modern standard such as SAML or OpenID Connect. Most enterprise-focused web applications have a built-in SAML capability, but the capability may need to be enabled, and sometimes an add-on needs to be purchased. Once the capability is enabled on the application, use an Okta Integration Network (OIN) pre-built SAML, WS-Fed, or OpenID Connect integrations to connect the application rapidly. Okta have over 1000 pre-built SAML application integrations, but if for some reason an integration is not available, create your own using the WS-Fed template, or the SAML and OpenID Connect Application Integration Wizard in the OIN. (And make sure you let us know, so that we can add your app to the catalog.)

2

Is the app currently protected by a WAM solution?

You may already be using a WAM solution like CA Siteminder or Oracle Access Manager to protect applications that don't support modern standards. If that's your case, you can migrate your on-premise applications from WAM to Okta, using Okta Access Gateway. The gateway acts as a broker between Okta and on-prem resources. On the on-premises side, it connects to apps using the legacy patterns they natively support. On the cloud side, the Gateway connects each application to Okta, using the secure standards broadly adopted by SaaS platforms. With the gateway, we've seen customers adopting a migration path on three incremental phases:

- **Step 1**: Integrate Otka with WAM using SAML. This step provides global SSO for users when accessing apps both on-prem and in the cloud, and also contains the WAM growth, since new apps will be directly integrated to Okta.
- **Step 2**: Scope and migrate on-prem apps from WAM to Okta. In this step, customers typically start migrating header-based and IWA apps and then move to agent-based apps. The agents are replaced with the gateway or other methods, such as SAML for WebLogic Server or our native integration with systems like E-Business Suite and PeopleSoft. This process continues until app apps are migrated to Okta.
- **Step 3**: Backup and Uninstall legacy WAM.

3

Does the app support a legacy authentication pattern?

Some organizations have on-premise apps that support legacy authentication patterns, but are not integrated to a WAM solution. These apps can be integrated with Okta using Access Gateway in the same way you would migrate from a WAM solution. The only difference on this approach is that you start you configure the integration by leveraging the app and access gateway documentation instead of migrating out of a legacy SSO platform.

4

Do you want to modernize the app?

This applies mostly to custom web applications. If you want to modernize the application, it's straightforward to add SAML or OIDC support to an existing web application. The implementation varies based on platform and development language so the Okta Developer site offers plenty of guidance across the most popular platforms. Modernizing applications take some time and effort, but it might be worth the investment in case you're already modernizing it due to a business requirement (i.e. porting application to a mobile platform).

Authentication (AuthN) versus Authorization (AuthZ)

Authentication refers to the binding of a user to an account using some secure credential, like a password. Authorization refers to the enforcement of access control within the app, and there are three levels, app-level – when users are authorized to launch the app, url-level – when users are authorized to access URLs within the app, and fine-grained – when users are allowed to see and interact with specific components within an app page, such as clicking a button or filling a form.

Legacy WAM tools can provide very fine-grained authorization. They usually do so using proprietary SDKs and protocols, which makes it costly to replace a WAM solution in scenarios where the fine-grained authorization is a requirement. That is, without modernizing the app. So Okta's recommendation here is to figure that additional complexity into your cost analysis when determining the full WAM replacement to your organization.

Putting It All Together

So, what does this Okta simplified integration architecture looks like? Figure 2 below shows how everything fits together using Okta Access Gateway.

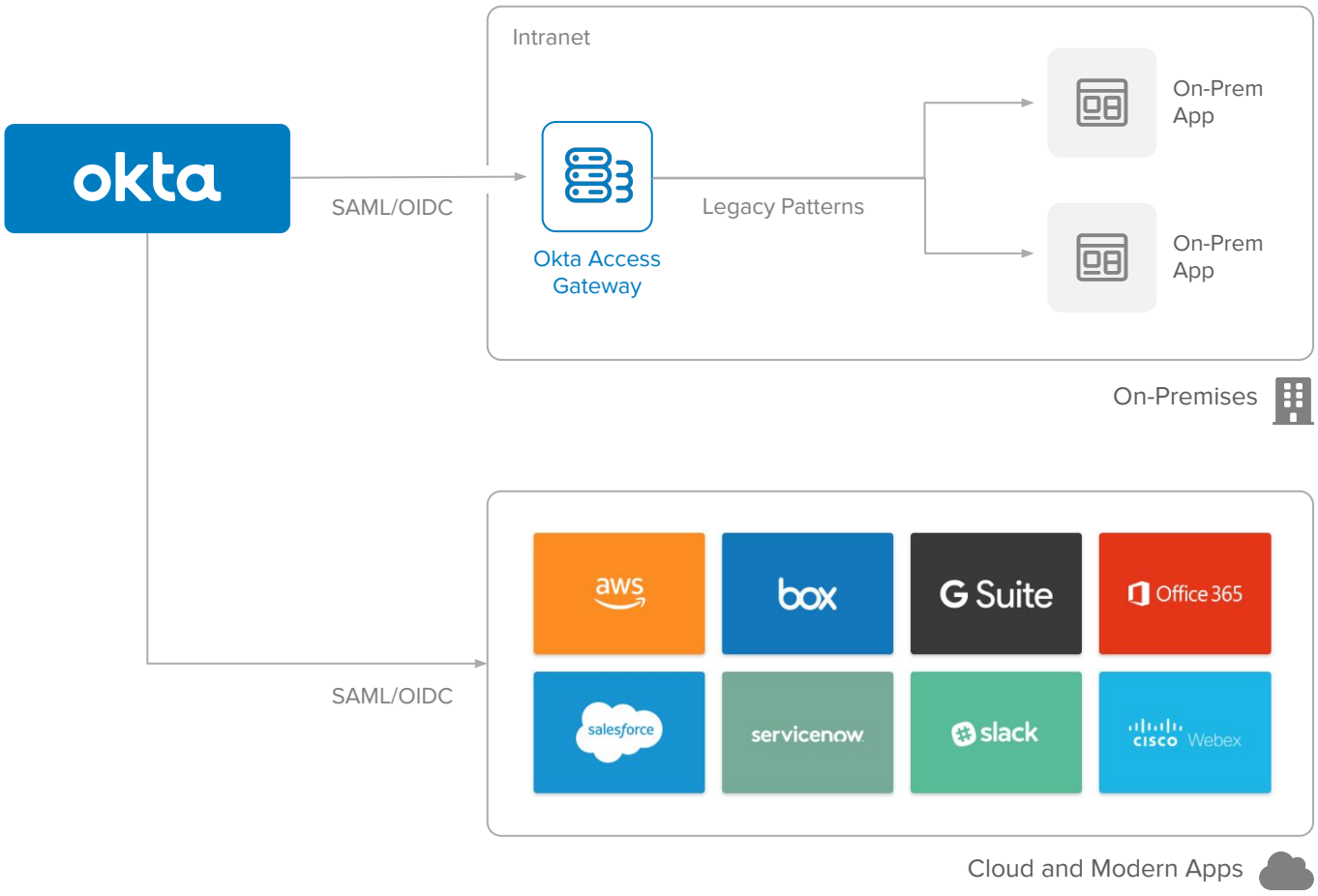

Figure 2. Reference Architecture for Okta with access to Cloud and Legacy Applications

This is a simple architecture, and that's really the point. The primary benefit is that it consolidates access with a single Identity Provider.

With a single identity provider, you can deliver a consistent single sign-on experience for end users whether they're accessing on-premises applications or SaaS solutions. You also cut costs by collapsing the infrastructure and removing legacy Single Sign-On solutions, and improve security by having policies in a single place and by leveraging modern cloud Multi-Factor Authentication.

Other Common Use Cases

Secure Access to On-Prem Apps from Outside the Firewall

Enterprises typically use Okta for the 6,000+ integrations pre-built into the Okta Integration Network. Okta also has full support for federation protocols for additional applications that support federation standards. Applications in the cloud with any kind of login form can, additionally, be easily added to Okta. When applications are behind the firewall, authentication is not enough. Users must gain network access to the application. This can be cumbersome with the standard VPN approach, requiring multiple steps for the end user.

With Okta Access Gateway, end users can authenticate once into Okta and seamlessly access on-prem applications. This architecture extends Okta's authentication capability to applications that do not have native authentication mechanisms or support header-based authentication.

Contractor and Partner Access to On-Prem SharePoint Portals

It can be a challenge to expose SharePoint Server (on-prem) to external users such as contractors or partners. Okta can integrate to SharePoint for SSO via federation. However, in order to use certain SharePoint modules, such as SharePoint business intelligence features, users must have a Kerberos token. Okta Access Gateway supports exchanging SAML assertions from Okta to Kerberos tokens, enabling use of the full set of functionality in SharePoint. Okta, paired with Access Gateway can manage contractor or partner identities and enforce multi-factor authentication.

Multi-Factor Authentication for Legacy Applications on IaaS

Enterprises that are moving on-prem servers to IaaS need to have a strategy for protecting access to those resources. One of the benefits of moving to IaaS may be that the service can be more easily reached from any network. Access Gateway can expose these on-prem servers to the internet. Given the greater exposure, a good practice is to require multi-factor authentication to access these services. Okta can easily add multi-factor authentication with a soft token (iOS, Android or Windows Phone), SMS or voice as factors.

One End User Portal for All Applications, On-Prem and Cloud

The Okta end user portal is built to make it easy for end users to access all their applications from one place. The portal is customizable by end users, which drives a high level of user adoption. Typically, organizations using the Okta portal want all the end users' applications exposed and accessible through the portal. Okta Access Gateway enables the user to log in once to Okta, and access all applications, cloud and on-prem, in one place.

Benefits of this Approach

This solution provides the best integration between on-prem apps with Okta, without having to change the way your on-prem app works today, offering the following benefits:

- Using a single identity provider for apps regardless of where they are hosted
- Reduce costs by collapsing the identity infrastructure and retiring legacy SSO servers
- Improve security with unified security policies and Adaptive Multi-Factor Authentication
- Reduce vendor risk by migrating our of deprecated or defunded solutions

Conclusion

By embracing the cloud, you will help your business to accelerate and gain critical advantages over your less agile competitors. Of course, the transition does not happen in the blink of an eye, so it's important to support your legacy systems for the foreseeable future. A modern identity management platform and a smart access management strategy will accelerate your IT evolution while bridging the gap between your trusty on-premises applications and the new technologies you're adopting.

To learn more

To learn more about Okta Access Gateway and how to secure access to on-premise web applications without changing how they work, visit our website www.okta.com/products/access-gateway.