How Microservices Can Improve Your IAM Strategy

Abstract

Identity Management is key to digital transformation and the next generation of enterprise IT. These foundational identity systems are rapidly evolving to support new business models, applications and ecosystems. A key element of this change is a pervasive movement to microservice and DevOps—providing a more flexible, dynamic and timely means of providing identity services.

Microservices, as applied to identity management, breaks up monolithic Identity and Access Management (IAM) applications into smaller services and provides them to a variety of applications and other consumers as needed. Microservices enable an "abstraction layer" that can dramatically simplify application development, integration and operational support. In this model, IAM services and functions that are enabled in a secure, easy-to-consume manner. As new protocols, techniques and infrastructure approaches emerge (e.g., blockchain, distributed ledgers, verified claims), and old techniques fade away the impact on the IAM infrastructure will remain minimal. Doing so is the surest way to 'future proof' your IAM strategy.

This highly actionable report provides a context for approaching microservices as an increasingly important part of an enterprise identity management strategy. We also describe a few significant vendor offerings in this emerging, but rapidly developing category.

Authors:

Doug Simmons
Managing Director, Consulting &
Principal Consulting Analyst
dsimmons@techvisionresearch.com

Nick Nikols
Managing Director, Research &
Principal Consulting Analyst
nick@techvisionresearch.com

Gary Rowe
CEO & Principal Consulting Analyst
gary@techvisionresearch.com

Table of Contents

Executive Summary

As enterprises embrace digital transformation, a renewed focus has been levied on the Development and Operations groups responsible for developing, integrating and maintaining IT services - including IAM, for all constituents, whether internal or external to the organization. Traditionally, these two groups operated in a somewhat isolated fashion, with Development working feverishly in its own silo, releasing new "IT products" (whether developed in-house or integrating commercial-off-the-shelf solutions) to the IT Operations staff - who are then challenged with maintaining these products. Such a waterfall model for application development has often led to disconnects between Development and Operations, because the process is innately non-iterative and the model makes timely feedback and regular cross-functional collaboration difficult.

There are two emerging, synergistic approaches to this dilemma that we'll evaluate as to its applicability in support of IAM. They are:

1. Logically combining the Development and Operations groups into a single entity referred to as DevOps, with a primary focus on agile development (the antithesis of the waterfall model) coupled with,
2. Microservices architecture and deployment strategies that break down business functions into atomic-level IT functions, or "services", that can be reused, retrofitted and replaced with minimal disruption to the overall IT infrastructure.

One of the most significant results of such a DevOps model is improved accessibility of IAM services functions, such as user/thing authentication, authorization, lifecycle management and audit readily available to application developers and integrators. Without an effort to move toward a better DevOps paradigm, enterprises will continue to suffer from siloed applications and services that will perpetuate the issues of disconnect that largely exist today.

Delivering IAM functionality via microservices involves the identification of discrete IAM functions such as user authentication within a compact, single-purpose code set. This single-purpose IAM microservice can then be easily and securely integrated into an in-house developed or COTS application. This approach provides an environment for further integration of a related set of microservices as needed by a set of related, callable services. These related services can include specific IAM services such as 'authenticate identity' or 'create identity', and so forth. Typically, individual, related microservices are 'packaged' in such a way using a container structure or an API gateway.

Many of the major IAM vendors have not been asleep at the wheel with respect to understanding, facilitating and furthering the development and usefulness of IAM microservices. That said, we see this movement as rapidly accelerating and recommend vendors and enterprises examine the opportunities associated with applying microservices and DevOps in an IAM context.

As TechVision Research has been emphasizing for the past three years, a combination of loose-coupling, virtualization and federation are critical elements in 'the future of identity management'. Microservices enables an 'abstraction layer' that can dramatically simplify application development, integration and operational support. In this model, IAM services and functions that are enabled in a secure, easy-to-consume manner. As new protocols, techniques and infrastructure approaches emerge (e.g., blockchain, distributed ledgers, verified claims), and old techniques fade away the impact on the IAM infrastructure will remain minimal by following the core principles we describe in this report. Doing so is the surest way to 'future proof' your IAM strategy.

Introduction

TechVision Research believes a flexible, loosely-coupled and federated Identity and Access Management (IAM) strategy is appropriate for most enterprises. This type of approach generally improves efficiency and reduces systems integration and operational overhead. These recommendations also focus on a bigger picture - that of a global trend toward digital transformation. Digital transformation is an area of focus for many large enterprises as they seek to digitally engage with clients, prospects, trading partners and employees.

Digital transformation can be broadly described as a significant overhaul of existing IT infrastructure to adopt a much more flexible, adaptable and scalable set of capabilities that can allow enterprises to quickly deploy or decommission online services with minimal impact to development and operations - which is precisely what we've been advocating for IAM transformation.

Over the past decade, an intense transformation of IT has taken place with the widespread adoption of cloud computing, SaaS and PaaS integration, system and infrastructure virtualization, mobility, federation, IoT, IAM and more. Add to this the growth in managed IT services supported by 3rd parties and the typical enterprise IT environment now looks nothing like it did 10 years ago.

As part of this transformation, a renewed focus has been levied on the Development and Operations groups responsible for developing, integrating and maintaining IT services - including IAM, for all constituents, whether internal or external to the organization. Traditionally, these two groups operated in a somewhat isolated fashion, with Development working feverishly in its own silo, releasing new "IT products" (whether developed in-house or integrating commercial-off-the-shelf solutions) to the IT Operations staff - who are then challenged with maintaining these products. Such a waterfall model for application development has often lead to severe disconnects between Development and Operations, because the process is non-iterative and comprises no viable near-real-time feedback loop.

We call these symptoms "DevOps anti-patterns", which have become prevalent across projects and platforms, typically resulting in infrequent releases with traditional change controls leading to long lead times, and manual, time-consuming, painful, and tedious delivery processes, which act as great inhibitors to digital transformation.

There are two emerging, synergistic approaches to this dilemma:

1. Logically combining the Development and Operations groups into a single entity referred to as DevOps, with a primary focus on agile development (the antithesis of the waterfall model) coupled with,

2. Microservices architecture and deployment strategies that break down business functions into atomic-level IT functions, or “services”, that can be reused, retrofitted and replaced with minimal disruption to the overall IT infrastructure.

One of the most significant results of such a DevOps model is improved accessibility of IAM services functions, such as user/thing authentication, authorization, lifecycle management and audit readily available to application developers and integrators. Without an effort to move toward a better DevOps paradigm, enterprises will continue to suffer from siloed applications and services that will perpetuate the issues of disconnect that largely exist today.

In this report, TechVision Research dives into the architecture and best practices for IAM to successfully participate in and support the emerging DevOps paradigm encompassing a microservices architecture strategy. The benefits of adopting this model will enable enterprises to provide more efficient, consistent, seamless and secure IAM capabilities.

What Are Microservices?

TechVision Research describes microservices and a microservices architecture as simply a way of designing applications as independently deployable services. Microservices start with a *focus on business capabilities*, not technology. It is an architecture style of developing a single application based on small, self-contained set of services running their own processes and communicating via a lightweight mechanism. There is limited centralized management or governance of these services and they can be written in different languages and use different storage approaches. Some key takeaways of this approach are:

1. Microservices define clear boundaries, but keep these boundaries small and not corrupted
2. Microservices are independently deployable by fully automated deployment machinery, which is particularly useful in deploying services on cloud-based PaaS environments, such as Amazon Web Services, Google and Microsoft Azure.
3. Microservices may be written in different programming languages and use different data storage technologies
4. Microservices support security, monitoring, and associated hooks to deploy code across multiple services

TechVision is currently drafting a report titled “*Microservices Enterprise Level-Set*” to be published in early 2018 which will add to the explanation of microservices and strategies to broadly leverage this approach. For the purposes of this report, microservices are components that can be thought of as independently replaceable and upgradeable building blocks or services.

Now that we’ve described microservices at a general level, we’ll next focus on describing how microservices and IAM fit together and the benefits to the enterprise.

What Is an IAM Microservice?

Identity and access management typically involves a number of functions regarding the establishment, management and use of identities built into a monolithic application to provide access to information as supported by policies. The enterprise goal is to provide end users (and applications) with appropriate access to enterprise systems and applications. The word "access" has two primary components: authentication and authorization.

1. Within authentication, systems and applications identify who someone or something is by looking at a host of attributes: login IDs, passwords, digital certificates, federation, one-time password tokens, etc.
2. Within authorization, attributes such as roles, group membership or other attributes or affiliations are used to grant or deny access.

The attributes necessary for determining whether to authenticate and authorize a person, service, application or thing must be provisioned and properly managed throughout the lifecycle of that person, application or thing. That leaves us with a host of typical IAM functions that can be isolated into atomic units such as:

- User (or application or thing) identity create
- User authenticate
- User authorize
- Session validation
- User identity modify
- User identity audit
- User identity disable or delete

A microservices approach looks to translate these specific business needs such as establishing a user's identity, authenticating a user, authorizing their access to services, making changes, auditing and removing a user to self-contained programs responsible for providing the appropriate functions. These are typical IAM functions that get aggregated in massive IAM solutions; this model decomposes these functions and reassembles them as needed. Each function can be independently developed, maintained and updated.

Such overarching functions may also have many permutations. For example, a 'user identity create' function may involve self-registration through a web page or app, or it may be triggered from an event in the enterprise HR system (e.g., new hire). The process may require workflows and approvals, and it may be necessary to log and audit these processes. Similarly, user authentication may involve a password, a digital certificate, biometric data and so forth; so there is an additional level of granularity associated with the overall authentication process. A set of authentication microservices could be developed to handle each of these permutations.

How the services communicate with each other depends on the application's requirements, but in many cases DevOps focuses on REST (Representational State Transfer) as a useful integration method because of its comparatively lower complexity over other protocols. REST is an architectural style that sets certain constraints for designing and building large-scale distributed systems. As an architectural style, REST has very broad application. The designs of both HTTP 1.1 and also URIs follow RESTful principles. Many other web services also follow the REST architectural style. Examples include OAuth 2.0 and OpenID Connect 1.0. In REST, IAM functions are referred to as 'resources' that are typically exposed at an API endpoint. An endpoint by itself is just a reference to a URI that accepts web requests that are RESTful.

IAM solutions can apply RESTful principles to define common endpoints for HTTP-based APIs that access web resources and collections of web resources for the following representative identity and access management operations, for example:

- Authentication Login
- Authentication Logout
- Session (cookie) Information
- Token attribute retrieval
- Token validation
- Logging
- Authorization
- OAuth 2.0 Authorization
- OpenID Connect 1.0
- User Self-Registration
- Resetting Forgotten Passwords
- Dashboard Service
- Identity Management (creating, reading, updating, deleting identities)
- Realm Management (creating, reading, updating, deleting realms)
- Secure Token Service (STS)

For instance, a simple User Authentication microservice would invoke user name/password authentication and return a token ID that applications can present as a cookie value for other operations that require authentication, while Authentication Logout can be performed by using the token ID to terminate the user (or application or thing) session. In this way, single sign-on (SSO) and single logout (SLO) can be supported.

Authorization, on the other hand, may require different microservices depending on the state of the user and application. For example, a microservice's REST API might simply perform cookie validation in order to ensure the user/application/thing have maintained a valid session from a previous authentication call. Another microservice REST API may be called to lookup attributes about the user/application/thing in order to allow the application to perform more fine-grained authorization based on the attribute values returned on the lookup. In fact, the logic to determine the authorization level would likely be another microservice that simply evaluates returned attribute values via rules and passes a 'go/no-go' message to the application. There is also an opportunity for additional machine learning/AI/predictive analytics to be applied here and could also be invoked via a microservice.

The key takeaway here is to understand that by making these standard IAM functions available as microservices, DevOps can make incremental, regular changes (as simple as a single change or patch to one line of code) without disturbing the entire IT infrastructure. This sets the stage for the continuous development and continuous deployment models that are a core part of the DevOps approach.

Granular Solutioning: Containers and API Gateways

Delivering IAM functionality via microservices involves the identification of discrete IAM functions such as user authentication within a compact, single-purpose code set. This single-purpose IAM microservice can then be easily and securely integrated into an in-house developed or COTS application. This approach provides an environment for further integration of a related set of microservices as needed by a group of related, callable services. These related services can include specific IAM services such as 'authenticate identity' or 'create identity', and so forth. Typically, individual, related microservices are 'packaged' in such a way using a container structure or an API gateway.

Containers

In many cases, applications may be 'containerized', so that once a business function is successfully enabled through a series of interconnected microservices, that business function can be easily and quickly leveraged across the enterprise as a cloud service - whether an internal, external or hybrid cloud. A container is the bundling of an application's code, configuration and dependencies into a single object. It can be thought of as operating system-level virtualization in that they are designed to run anywhere and therefore are independent of the external environment.

Containers are intended to be very lightweight and function as a unit of software delivery in support of microservices. Containers, therefore, are typically comprised of one or more microservices that taken together perform discrete business IT functions in the form of easy to use building blocks that deliver environmental consistency, operational efficiency, developer productivity, and version control.

In the world of IAM, a container called "Authenticate Identity" may be deployed with the contents of this particular container being three or four independent microservices. Each of these microservices will perform one specific function within a typical authentication event, such as password authentication, certificate authentication or biometric authentication. Depending on what data is passed to the container by the application, the appropriate microservice within that container will be invoked.

API Gateways

API gateways are used to integrate multiple related microservices into a logically discrete and addressable function. An API gateway functions as an "API wrapper" such that individual microservices behave as a single application to the client. An API gateway is often referred to as a "façade" that provides simple API interfaces to a complex subsystem (e.g., the IAM infrastructure). It essentially decouples the interface that clients see (in this case API consumers which could be mobile apps, thin clients, application service accounts, smart meters, home coffee makers, etc.) from the underlying implementation. In addition, the API gateway can perform several additional tasks such as API limit throttling, logging, security, etc.

API gateways provide consistent RESTful application programming interfaces (APIs) for mobile and web applications to access backend services, whether on-premise or in the cloud.

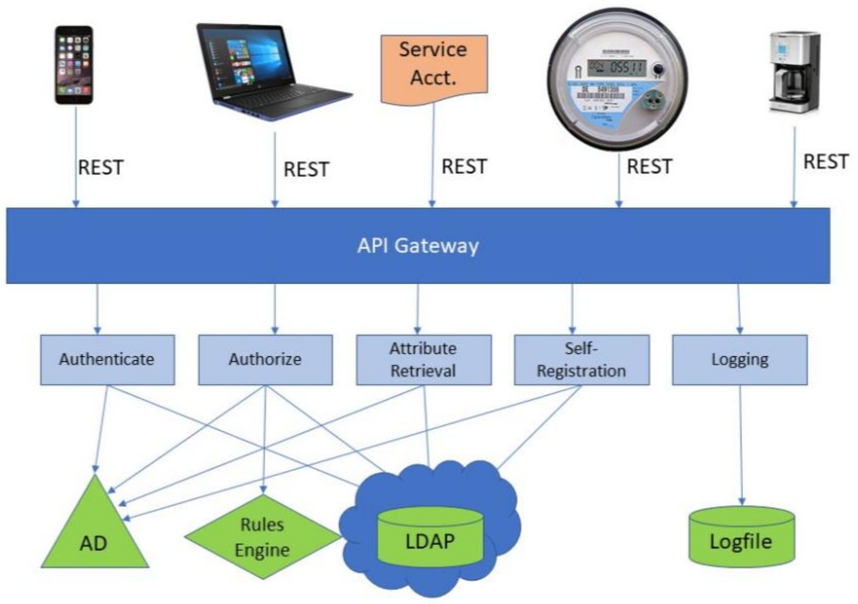

Figure 1: API Gateway Servicing IAM Functionality

As shown in Figure 1, API gateways provide consistent access to identity services for mobile, web, service account and IoT device. All client requests go through the API gateway via a RESTful interface. The API gateway then routes requests to the appropriate microservice. These requests are often handled by invoking multiple microservices and aggregating the results, as illustrated by multiple connections (e.g., lines) emanating from a single microservice to multiple underlying infrastructure. Since this gateway provides client specific APIs it reduces the number of round-trips between the client and application which reduces network latency and it also simplifies the client code.

There are a number of vendors that offer API gateways; IBM, Akana, Apigee, and CA all have strong solutions. Additionally, Amazon provides an API Gateway within AWS that handles the tasks involved in accepting and processing large pools of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. Google offers Google Cloud Endpoints with similar functionality, while Microsoft offers Azure API Management as a solution for publishing APIs to external and internal customers. The combination of API gateway vendors and the big three (Amazon, Google and Microsoft) provide a strong set of services for enterprises moving towards microservices and DevOps.

Whether using containers or API gateways, the message here is that the DevOps model requires a viable means of integrating discrete IAM functionality into a set of unique but consistently applied callable services. This starts with breaking down IAM functionality into atomic-level functions and then re-assembling them into a series of callable services. This type of approach supports the true essence of what we characterize as loose-coupling and flexibility in providing identity services. DevOps allows necessary and regular changes to production code to regularly refresh the enterprise IAM capability. This is in stark contrast to searching for and replacing logic in multiple siloed instances of redundant IAM functionality that may take years to deploy.

Does Blockchain IAM Fit This Model?

TechVision Research has published several research reports focused on blockchain and blockchain-based IAM. Blockchain (or distributed ledgers) can provide greater discoverability of an identity and secure connections to the data needed to support a transaction. Each block in one's blockchain may contain "pointers" to encrypted and signed verifiable claims (e.g., a qualification, achievement, quality, or piece of information about an entity's background such as a name, government ID, payment provider, home address, or university degree) that can be used to prove a specific attestation in real time.

Entities (people, organizations, devices) need to make many kinds of claims as part of their everyday activities. As organizations progress towards digital transformation, entities need to be able to transmit instantly verifiable claims (e.g., about their location, accomplishments, value and so forth) providing electronic proof that the claim is valid. These claims can support the next generation of web applications as they provide the basis for authorizing entities to perform actions based on rich sets of credentials issued by trusted parties.

TechVision Research sees a synergistic relationship between IAM microservices and blockchain-based IAM because a massive proliferation of identities over the next few years-especially of "things", will require a much more scalable and secure IAM ecosystem.
.

TechVision Research sees a synergistic relationship between IAM microservices and blockchain-based IAM because a massive proliferation of identities over the next few years-especially of "things", will require a much more scalable and secure IAM ecosystem. By expanding this ecosystem to include verifiable claims as a principal means of authentication and subsequent authorization, application developers will be able to use REST endpoints to quickly and securely extend IAM to a broader range of clients, be they end-users, applications or devices (e.g., IoT "things").

A much more in-depth discussion regarding blockchain-based IAM is provided in the TechVision Research report titled "Identity as the New Perimeter", October, 2017 and "Blockchain-Based Identity" research published in October, 2016.

The First Step: Enable an Identity Data Service

Moving to an IAM microservices model takes some preparation. There is a lot of complexity involved in performing identity operations across disparate systems, especially in large enterprises. That said, identity operations can be decomposed in a way that is easier for developers to use in a way that helps the enterprise to establish a level of consistency, efficiency and security while providing regular updates.

To start the process enterprises should decompose business applications into discrete services that provide a broad range of 'common functionality' required by application consumers. A high-level example of this is shown in Figure 2, below.

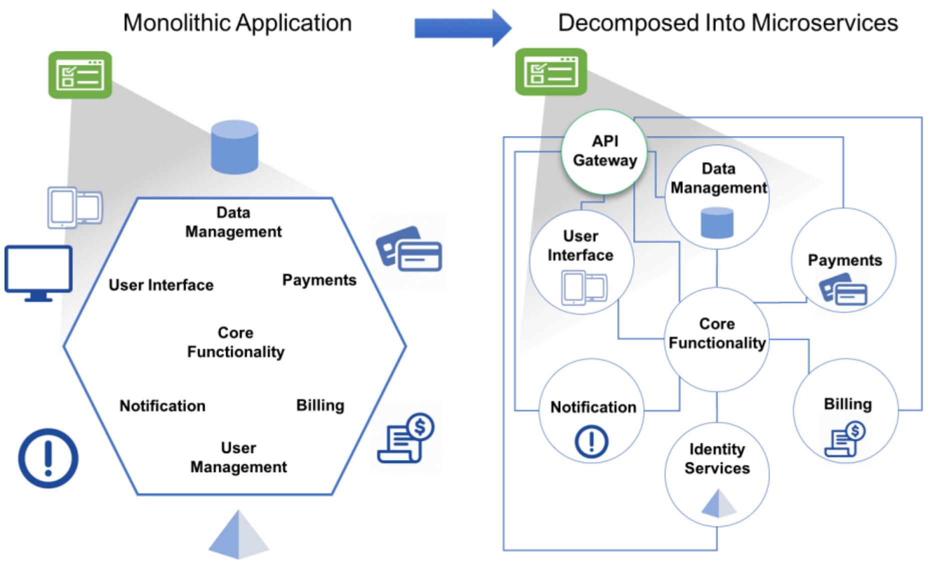

Figure 2: Decomposing Applications Into Microservices

Nearly every application contains functionality similar with the monolithic application illustrated on the left of Figure 2. These monolithic applications include functionality such as user management, data management, user interface, billing, payments and such functions that are quite common. However, almost every application is also supported by a set of core functions or services.

The challenge is to decompose the common services from the unique core functions. At the end of this decomposition exercise, the application should be comprised mainly of its unique core functions, while incorporating the common enterprise functions (e.g., authentication, identity management, etc.) through service calls to the Identity Service as illustrated on the right of Figure 2.

What is especially appealing to such a composable architecture is the ability to hide complexity, especially of complex IAM functions, from application developers and operations staff. Microservices enables an 'abstraction layer' that can dramatically simplify application development, integration and operational support. As we have been describing throughout this report, such an abstraction layer is the product of an API set that serves up the necessary IAM functions to developers and integrators. We illustrate this concept more fully in Figure 3, below.

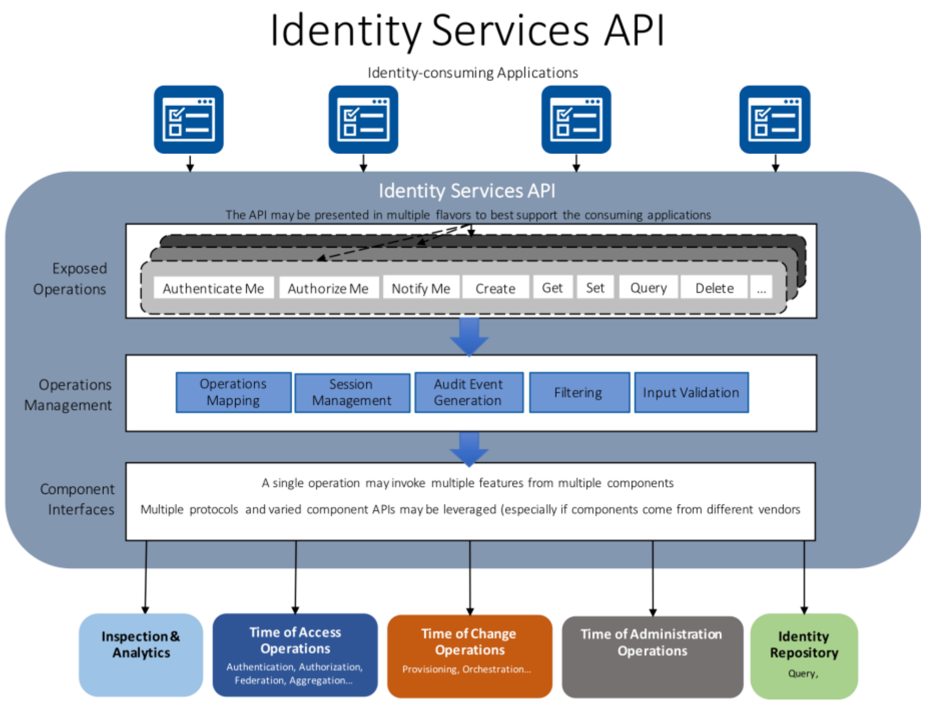
Identity Services API
Identity-consuming Applications
Identity Services API
The API may be presented in multiple flavors to best support the consuming applications
Exposed Operations
Authenticate Me
Authorize Me
Notify Me
Create
Get
Set
Query
Delete
...
Operations Management
Operations Mapping
Session Management
Audit Event Generation
Filtering
Input Validation
Component Interfaces
A single operation may invoke multiple features from multiple components
Multiple protocols and varied component APIs may be leveraged (especially if components come from different vendors
Inspection & Analytics
Time of Access Operations
Authentication, Authorization, Federation, Aggregation...
Time of Change Operations
Provisioning, Orchestration...
Time of Administration Operations
Identity Repository
Query,

Figure 3: Identity Services API

Consider that the functionality enabled through this API described above is a deeper dive into the concepts first illustrated in Figure 1 earlier in this document. The 'exposed operations' illustrated across the top of the Identity Services API example in Figure 3 are the types of services that can be containerized or exposed via API Gateways in order to enable these services in the 'real world'.

As an example, note the Identity Orchestration Service illustrated in Figure 4, below:

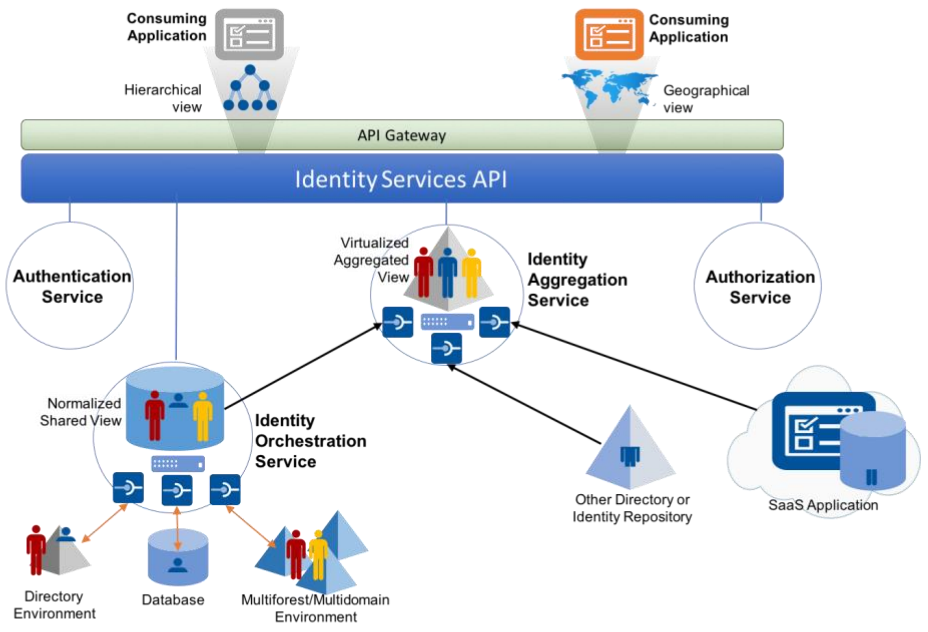

Figure 4: Tying It All Together

While microservices provides a degree of flexibility and regular updates, the challenge is to then reassemble all the components into an integrated solution. This is where identity orchestration comes into play. The identity orchestration service supports numerous very important capabilities within the realm of IAM, such as collecting identity data from numerous integrated repositories (e.g., directories, databases, AD, etc.). Within the identity orchestration service, mapping rules are invoked to assemble a complete picture of the identity and make this data available through the secure identity APIs identity aggregation microservice, shown in the center of the diagram. This pattern is often used for entitlement collection (building an entitlement catalog), which further enables coarse and fine-grain authorization microservice exposure. If attributes are changing frequently – collection can be more dynamic and in real time and the collection point doesn't need to be persistent (Virtual Directory). All this decomposition 'adds up' to a much more flexible yet consistent means for integrating IAM functions across the enterprise.

All this decomposition 'adds up' to a much more flexible yet consistent means for integrating IAM functions across the enterprise.

Let's now turn to what the vendors and service providers are doing to drive this movement towards microservices-based IAM.

Early Vendor Focus on IAM Microservices

Many of the major IAM vendors have not been asleep at the wheel with respect to understanding, facilitating and furthering the development and usefulness of IAM microservices. That said, we see this movement as rapidly accelerating and recommend vendors and enterprises take a closer look at the opportunities associated with applying microservices and DevOps approaches to IAM.

While we examine a few of the early leaders in this movement, it bears mentioning that more vendors are climbing onboard and re-architecting their solutions to better fit a microservices-centric DevOps approach to IAM. We'll describe the offerings and approach of 5 vendors that have been early movers in this space:

- Cloudentity
- ForgeRock
- Okta
- Microsoft
- IBM

These vendors represent some new players and established IAM vendors.

Cloudentity

Cloudentity, based in Seattle, WA, has been in business for several years as a security and IAM consulting company and has recently transitioned to a product/service vendor. TechVision had the opportunity to recently interview Nathaniel Coffing, their founder and CEO. Nathaniel explained he saw a unique opportunity to respond to what he perceived as a major need in Finance, Insurance, Telecommunications, Government, Retail and Healthcare industries. He explained that they were looking for secure Customer Identity for cloud-native, hybrid-cloud and multi-cloud applications, and Cloudentity responded with an IAM microservices offerings optimized for speed and flexibility. This was productized as the Cloudentity TRUST Engine, which uses behavioral machine learning to deliver continuous adaptive authentication, authorization and management across users-services-things.

The Cloudentity Customer IAM solution delivers a stateless identity solution built on a cloud-native microservices architecture with RESTful APIs. Their solution provides a broad range of identity functionality with more than 30 microservices including full lifecycle management of users+services+things, user self-service, strong authentication and session mobility, and is designed specifically for modern container architectures. The high-level Cloudentity solution is shown in Figure 5, below.

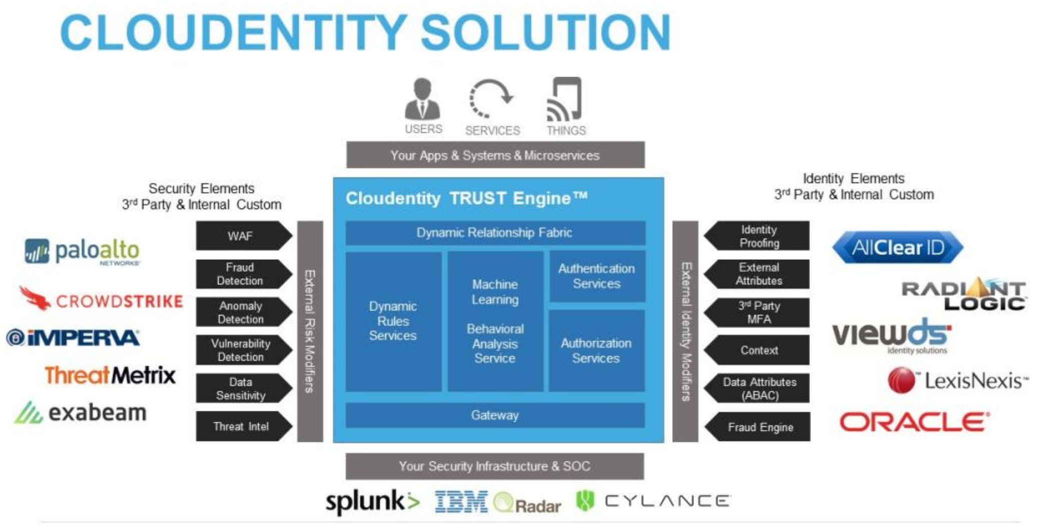

Figure 5: The Cloudentity Solution Framework

Some of the aspirations behind Cloudentity's IAM architecture and strategy include:

1. Attack surface reduction
   a. Patch the service, not the platform
   b. Deploy only required features
2. More efficient deployment
   a. Enable new features as microservices - reduces DevOps churn as new IAM capabilities are added, existing capabilities are changed and defunct capabilities removed.
   b. Cloud readiness - as a cloud native solution, Cloudentity aims to better facilitate distributed deployment on global cloud services such as AWS, Microsoft Azure, Google and so forth.
   c. DevOps friendly - facilitates integration into the Continuous Integration/Continuous Delivery (CI/CD) pipeline - developed with a microservices foundation.
   d. Adaptable API - by virtue of its ground-up microservices pedigree, the IAM APIs can be integrated in customer business-specific ways in order to provide more flexibility.
   e. Scalable - again, as a comprehensive set of discrete IAM microservices, Cloudentity's solution is intended to fit the deployment requirements for a globally scalable infrastructure deployed on the cloud.

The Cloudentity TRUST Engine uses behavioral machine learning to deliver continuous adaptive authentication, authorization and relationship management between users/services/things. It employs an adaptive risk-based transaction authorization approach to dynamically score risk across user roles, transactional context and risk profiles. The TRUST

Engine detects changing risk across behavioral, data security and threat intelligence, escalating identity requirements for authorization on higher risk transactions while detecting, mitigating and blocking suspicious users, devices or activities.

Cloudentity's TRUST Engine serves as the logically centralized 'nervous system' of the IAM platform, enabling the following features:

1. Secure containers, APIs and microservices to enable:
    a. Distributed authentication and authorization of users, services and things
    b. Distributed session store
    c. Federation - including OAuth2, OpenID Connect and SAML2
    d. Activity profiling
    e. Integrated threat intelligence
    f. Policy promotion between environments
    g. Parameter inspection
2. Inbound transactional security
    a. Verified claims
    b. SSL key offloading
    c. SSL key rotation
    d. Traffic inspection
    e. Parameter validation
    f. Behavioral baseline

ForgeRock

In June 2017, ForgeRock introduced ForgeRock Identity Microservices, a set of standards-based identity microservices, designed to provide a way to increase application security by integrating identity into service-to-service interactions.

TechVision has the opportunity to recently interview Lasse Andresen, ForgeRock's co-founder and CTO. He described ForgeRock's Identity Microservices as allowing developers to secure transactions between microservices, users, things, and other entities by validating the identity of the caller and checking for proper authorization. ForgeRock has focused on both IoT support as well as Customer IAM and microservices can drive increased scalability and availability for both environments.

ForgeRock's Identity Microservices are a subset of the ForgeRock Identity Platform and run as self-contained microservices in a stateless mode. This functionality helps microservices developers provide identity services in their applications, independent of the platform technology used. The ForgeRock Identity Microservices are supported on containers such as Docker, Kubernetes and Cloud Foundry (discussed further, below).

Four self-contained ForgeRock Identity Microservices are currently available:

1. ForgeRock Authentication Microservice: This microservice will authenticate other microservices to ensure they are legitimate services and will provide OAuth2-based authorization tokens with a limited scope that is bound to the calling service.
2. ForgeRock Token Validation Microservice: This microservice will offer a token

introspection endpoint and be able to validate authentication tokens issued by the ForgeRock Authentication Microservice as well as ForgeRock OpenAccess Manager (OpenAM).
3. ForgeRock Token Exchange Microservice: This microservice will validate the identity of a user and a service that is trying to call another service on behalf of a user. It will ensure that other microservices in the same environment can talk to each other only if they are authorized to do that and with the minimum amount of authorization necessary.
4. ForgeRock Authorization Microservice: This microservice will provide local declarative validation of policies as well as proxy over external policy decision points (PDPs). It is designed to use external policy decision points such as ForgeRock OpenAM only when needed.

Cloud Foundry is an open source cloud computing Platform as a Service (PaaS) that is available as freeware, and also as commercial offerings from Pivotal Software, IBM Bluemix, Swisscom, HP and several other vendors. All of the major iterations of Cloud Foundry offer a collection of platform elements that enable developers to create and host production versions of online services and applications. These platform elements include features for monitoring, logging, messaging, authentication, traffic routing and other tasks.

One of the core concepts of the Cloud Foundry project is its use of a service broker. A service broker is code that enables an application in the cloud to invoke or point to a needed service for that application to run.

For example, the ForgeRock service broker supports OAuth2, allowing customers to extend OpenAM capabilities to secure those microservices. OAuth is a standard for authorizing access to applications and data. OAuth promotes a least privilege model, allowing a user to grant limited access to their applications and data by issuing a token with limited capability. OAuth is beneficial because it hands the management of web delegation to the actual resource owner. One of the significant advancements OAuth brings to the table is formalizing the process of delegating identity mapping to users. OAuth originated through the OpenID project at Twitter, and became a standard with input from Google and other Internet companies.

Okta

Based in San Francisco, Okta was founded in 2009 by two ex-Salesforce.com employees, and became a publicly-traded company in 2017 with over 2,500 customers. Unlike other identity providers that are moving their on-premise offerings to the cloud, Okta started with a cloud-based identity management service built on AWS and has established a leadership position in this space.

TechVision had the opportunity to interview Eric Berg, Okta's Chief Product Officer, earlier this year as he described a strategy of enabling developers to leverage Okta services. This vision was more clearly visible with the acquisition of developer identity tool vendor Stormpath in May of 2017. In August, 2017 Okta announced a new developers edition for employee and customer IAM with an expanded portfolio of APIs and new developer tools

Okta currently provides multiple API endpoints that fit into the model we are describing as they may be encapsulated in a RESTful architecture and provide specific services including:

1. Authentication - The Okta Authentication API provides operations to authenticate users, perform multi-factor enrollment and verification, recover forgotten passwords, and unlock accounts. It can be used as a standalone API to provide the identity layer on top of your existing application, or it can be integrated with the Okta Sessions API to obtain an Okta session cookie and access apps within Okta.
2. OAuth - The OAuth 2.0 API provides API security via scoped access tokens, and OpenID Connect provides user authentication and an SSO layer which is lighter and easier to use than SAML.
3. OpenID - The OpenID Connect API endpoints enable clients to use OIDC workflows with Okta. With OpenID Connect, a client can use Okta as a broker. The user authenticates against identity providers like Google, Facebook, LinkedIn, or Microsoft, and the client obtains an Okta session.
4. Client Registration - The Dynamic Client Registration API provides operations to register and manage client applications for use with Okta's OAuth 2.0 and OpenID Connect endpoints.
5. Session Management - Okta uses a cookie-based authentication mechanism to maintain a user's authentication session across web requests. The Okta Sessions API provides operations to create and manage authentication sessions for users.
6. Applications - The Okta Application API provides operations to manage applications and/or application access rights to users or groups.
7. Events - The Okta Events API provides read access to the system log and enables exporting of event data as a batch job for reporting or analysis.
8. Factors - The Okta Factors API provides operations to enroll, manage, and verify factors for multi-factor authentication (MFA).
9. Groups - The Okta Groups API provides operations to manage Okta groups and their user members.
10. Identity Providers - The Okta Identity Providers API provides operations to manage federations with external Identity Providers (IDP), to support logging in with credentials from Facebook, Google, LinkedIn, Microsoft, or an enterprise IdP using SAML 2.0 protocol.
11. Logs - The Okta System Log API provides read access to the Okta system log, and provides more functionality than the Events API, containing more structured data.
12. Admin Roles - The Okta Admin Roles API provides operations to manage administrative role assignments for a user.
13. Schemas - Okta's Universal Directory allows administrators to define custom user profiles for Okta users and applications. Okta has adopted a subset JSON Schema Draft 4 as the schema language to describe and validate extensible user profiles. JSON Schema is a lightweight declarative format for describing the structure, constraints, and validation of JSON documents.
14. Users - The Okta User API provides operations to manage users, including create, modify, deactivate, password assignment, password update and many more.

Microsoft

Azure Active Directory (Azure AD) is Microsoft's multi-tenant, cloud based directory and identity management service. Azure AD serves as a platform that enables developers to deliver access control to their applications, based on centralized policy and rules. Azure AD supports REST APIs as service endpoints that enable sets of HTTP operations (i.e., methods), which provide create, retrieve, update, or delete access to the AD service's resources.

Azure AD serves as a platform that enables developers to deliver access control to their applications, based on centralized policy and rules.

Azure AD services require client code to authenticate with valid credentials before calling the service's API. Authentication is coordinated by Azure AD, and provides the client application with an access token as proof of the authentication. The token is then sent to the Azure service in the HTTP Authorization header of subsequent REST API requests. The token's claims also provide information to the service, allowing it to validate the client and perform any required authorization. In accordance with the OAuth2 Authorization Framework, Azure AD supports two types of clients:

- Web/confidential clients run on a web server and can access resources under their own identity (for example, a service account or daemon), or obtain delegated authorization to access resources under the identity of a signed-in user (for example, a web app). Only a web client can securely maintain and present its own credentials during Azure AD authentication to acquire an access token.
- Native/public clients are installed and run on a device. They can access resources only under delegated authorization, using the identity of the signed-in user to acquire an access token on behalf of the user.

IBM

IBM is an established IAM vendor with IBM Security Access Manager and Identity Manager deployed in thousands of organizations. Customers can develop custom applications that use the IBM Security Access Manager for Mobile REST application programming interfaces (APIs) that are implemented based on REST services. The REST APIs are available for customers to administer the management tasks, using JAX-RS, outside of the IBM Security Access Manager for Mobile user interface. This includes exporting and importing configuration from one environment to another or backing up configuration.

Additionally, customers can develop custom applications by using the REST application programming interfaces (APIs) that come with the IBM Security Identity Manager in support of identity lifecycle management. The REST APIs enable identity administration tasks outside of the IBM Security Identity Manager user interface. The REST APIs are segregated into a set of functional components of IBM Security Identity Manager that are listed below:

- Person Management
- View or edit user profiles.
- System User Management
- Search capability for the IBM Security Identity Manager system users based on unique identifiers.
- Password Management
- Change or reset the password, and recover the forgotten password.
- Access Management
- Request, view, edit, or delete the access.
- Activity Management
- View and act on your activities.
- Delegation Management
- Delegate activities, view, edit, and delete the delegation schedule.
- Generic Search APIs
- Assorted set of search capabilities that are provided by the REST APIs.

These overviews of various vendors' capabilities in this space are certainly not exhaustive. However, it should be clear that the horse has left the barn with respect to the shifting paradigm associated with DevOps, microservices and IAM.

Recommendations and Best Practices

By acknowledging that application development is shifting toward leveraging microservices based architectures, enterprises need to understand that this evolution involves IAM in a not-so-minor way. Identity microservices can simplify how applications interact with identity information while improving security and consistency across applications and can take the burden of this complexity off of the application developers' shoulders. These services can also be readily updated and consistently available to any appropriate identity consumer.

Key enterprise recommendations and best practices for microservice-enabled IAM include:

1. Organizations committed to a digital transformation strategy should spend time focusing on the DevOps paradigm discussed earlier in this report. An iterative process from design, development, production and retirement should better fit with an Agile development model than the traditional waterfall model. You should start by examining your own development and operations processes and determine whether to embark upon an Agile Development program. If you move forward with Agile/DevOps you should:
    a. Ensure the program is governed properly. Solidify stakeholder buy-in from the top-down (key executives and application 'owners') as well as the bottom-up (application developers and operations).
    b. If the organization is already embarking on an agile DevOps model, ensure that IAM microservices are considered key building blocks to this strategy.

2. Applications should be inventoried to understand the current, inherent complexity

(assuming there is) of IAM integration and to begin the process of disentangling IAM from siloed code supporting these applications. Specific steps include:

a. Identify commonality across the many ways existing applications perform IAM functions such as identity authentication, authorization, lifecycle management and so forth.
b. The commonalities identified will lead to a smaller subset of key IAM functions that would be better served as microservices that are easily and securely callable from these applications.
c. Understand that in some cases, existing applications will be very difficult - if not impossible, to refactor in order to remove the siloed IAM functionality. Applications that fall into this category should be risk-ranked in order to fully capture and assess the risk of leaving them as-is rather than assuming the cost to re-write the applications.
d. Develop a roadmap for application refactoring in order to prioritize and resource the DevOps effort, understand the residual risk while in the process of refactoring and identify possible mitigation approaches (e.g., increasing application activity monitoring for high-risk applications that remain further down in the roadmap for reasons such as complexity, budget, resourcing, etc.

3. Thoroughly assess your existing IAM infrastructure to determine its ability to enable all necessary IAM services to be called via RESTful endpoints from within applications.
   a. Request that your current IAM vendor provide information regarding their ability to function in a microservices oriented IT ecosystem.
   b. Request that your IAM vendor (or systems integrator) provide contact with other customers of theirs who have embarked on a similar journey, in order to fast-track your own deployment.

4. Determine whether containerization or an API gateway is the better predominant model for enabling and securing IAM REST endpoints. If an API gateway model suits your organization better, spend time evaluating API gateway vendors' experience with IAM integration.

5. Start with a smaller, well-contained deployment strategy in order to avoid the pitfalls of the boil-the-ocean 'strategy'.
   a. Pick a few lower-risk applications to begin with, establishing a proof-of-concept with an eye on operationalization.
   b. Once the concept has been proven, identify the higher risk applications from your deployment roadmap to transition first as a means to bring immediate value and risk reduction.

Summary

What does it mean to be 'cloud-first'? Many of the organizations that TechVision Research works with on an ongoing basis have adopted a cloud-first strategy over the past several years - with the number growing more rapidly day-by-day. Being cloud-first doesn't only mean that you run your applications and services on a cloud platform - whether SaaS, PaaS or IaaS. It means that your organization has taken a deep and focused look at digital transformation and has determined that the place to start this transformation is to leverage multi-tenant, elastic cloud infrastructure as the 'delivery platform'.

But, it also means that your organization has inherently taken on a more loosely-coupled, federated and dynamic (e.g., virtualized) application integration and delivery model. As TechVision Research has been expounding for the past three years, loose-coupling, virtualization and federation are the keys to 'the future of identity management'.

We have described a path toward evaluating current IAM services in a digital transformation context, breaking up monolithic services into smaller components and leveraging these components to provide more scalable, dynamic and available identity services. We've also provided some background on vendors that are investing in IAM microservices.

To walk the walk, an organization must consider the importance of IAM services and functions that are enabled in a secure, easy-to-consume manner. In this way, as new protocols, techniques and infrastructure approaches emerge (e.g., blockchain, distributed ledgers, verified claims), and old techniques fade away the impact on the IAM infrastructure will remain minimal. Doing so is the surest way to 'future proof' your IAM strategy.

About TechVision

World-class research requires world-class consulting analysts and our team is just that. Gaining value from research also means having access to research. All **TechVision Research** licenses are enterprise licenses; this means everyone that needs access to content can have it. We know major technology initiatives involve many different skill sets across an organization and limiting content to a few can compromise the effectiveness of the team and the success of the initiative. Our research leverages our team's in-depth knowledge as well as their real-world consulting experience. We combine great analyst skills with real world client experiences to provide a deep and balanced perspective.

TechVision Consulting builds off our research with specific projects to help organizations better understand, architect, select, build, and deploy infrastructure technologies. Our well-rounded experience and strong analytical skills help us separate the hype from the reality. This provides organizations with a deeper understanding of the full scope of vendor capabilities, product life cycles, and a basis for making more informed decisions. We also support vendors when they carry out a product and strategy review and assessment, a requirement analysis, a target market assessment, a technology trend analysis, a go-to-market plan assessment, or a gap analysis.

TechVision Updates will provide regular updates on the latest developments with respect to the issues addressed in this report.

About the Authors

Doug Simmons brings more than 25 years of experience in IT security, risk management and identity and access management (IAM). He focuses on IT security, risk management and IAM. Doug holds a double major in Computer Science and Business Administration.

While leading consulting at Burton Group for 10 years and security, and identity management consulting at Gartner for 5 years, Doug has performed hundreds of engagements for large enterprise clients in multiple vertical industries including financial services, health care, higher education, federal and state government, manufacturing, aerospace, energy, utilities and critical infrastructure.

Nick Nikols has more than 25 years of experience in the software industry, architecting solutions and developing innovative products for identity, security and compliance management, as well as directory services and directory/application integration.

Before working with TechVision Research, Nick was Senior Vice President of Product Management and CTO of Cybersecurity at CA Technologies, where he was responsible for CA's Cybersecurity Product Strategy and Roadmap. At CA, he was particularly focused on modernizing CA's Identity-centric Security portfolio and successfully promoted CA's Identity Manager and Access Governance solution into a leadership position within Gartner's Magic Quadrant for Identity Governance and Administration.

Gary Rowe is a seasoned technology analyst, consultant, advisor, executive and entrepreneur. Mr. Rowe helped architect, build and sell two companies and has been on the forefront the standardization and business application of core infrastructure technologies over the past 35 years. He was President of Burton Group from 1999 to 2010, the leading technology infrastructure research and consulting firm through the sale of Burton to Gartner.

Mr. Rowe has personally led over 100 consulting engagements, 50+ educational seminars, published over 50 research reports/articles and led three significant technology industry initiatives. His combination of business skills and his deep understanding of technology provide a balanced perspective for clients. Core areas of focus include identity and access management, directory integration, cloud computing, security/risk management, digital transformation, IT business model changes, privacy and blockchain/distributed ledger."

Related Reports

The following reports might be helpful in your continued exploration of this domain:

Identity is the New Perimeter

Authors: Doug Simmons – Managing Director, Consulting / Principal Consulting Analyst,
Nick Nikols – Managing Director, Research / Principal Consulting Analyst,
Gary Rowe – CEO/ Principal Consulting Analyst,
Gary Zimmerman – CMO / Principal Consulting Analyst

Cloud-based Identity Management

Authors: Nick Nikols – Managing Director, Research / Principal Consulting Analyst,
Gary Rowe – CEO/ Principal Consulting Analyst

Banking on Identity

Authors: David Goodman, D. Phil, Principal Consulting Analyst
Rhomaios Ram, Principal Consulting Analyst

Getting to Know Your Customers: The Emergence of CIAM

Authors: David Goodman, D. Phil, Principal Consulting Analyst

Blockchain-based Identity Management

Authors: Doug Simmons – Managing Director, Consulting / Principal Consulting Analyst

Context-based Identity Management

Authors: David Goodman, D. Phil, Principal Consulting Analyst

The Future of Identity Management

Authors: Doug Simmons – Managing Director, Consulting / Principal Consulting Analyst,
David Goodman, D. Phil, Principal Consulting Analyst,
Gary Rowe – CEO/ Principal Consulting Analyst,
Bill Bonney – Principal Consulting Analyst