International Journal of Core Engineering & Management
Volume-7, Issue-12, 2024 ISSN No: 2348-9510

# MIGRATION OF LEGACY IDENTITY SYSTEMS TO OKTA USING A STANDARDS-BASED FRAMEWORK FOR MULTI-TENANT APPLICATIONS

*Tathagata Roy*
*Independent Researcher, Colorado, USA*
*tatha.roy@gmail.com*

*Abstract*

***Legacy in-house Identity and Access Management platforms that rely on proprietary authentication mechanisms create significant limitations for scalability, interoperability, and security in multi-tenant enterprise applications. As distributed architectures evolve, the absence of OAuth2 and OpenID Connect support in these systems increases operational risk and restricts integration across diverse client types, including backend services, web applications, mobile applications, and single-page applications. This paper presents a structured migration framework for retiring non-standard identity systems and adopting Okta as an externalized standards-based Identity Provider. The methodology focuses on secure tenant-aware data migration, identity mapping, and the preservation of isolation boundaries throughout the transition. The analysis includes architectural refactoring patterns required across heterogeneous clients to support token-based authentication, centralized policy enforcement, and delegated authorization flows. By examining the migration lifecycle, coexistence strategies, and application-layer integration models, this work demonstrates how adopting an open-standards provider improves security posture, reduces maintenance overhead, and enables scalable identity management across varied tenant populations.***

***Index Terms— Identity and Access Management (IAM); OAuth2; OpenID Connect (OIDC); Okta; Multi-Tenant Architecture; Legacy System Decommissioning; Token-Based Authentication; Data Migration; Cloud Security; Enterprise Architecture; Client Integration Models.***

## I. INTRODUCTION

Enterprise security architectures have undergone a fundamental shift from perimeter-based controls toward identity-centric and zero-trust models [3], [13]. In traditional environments, network boundaries and Virtual Private Networks (VPN) served as the primary enforcement mechanisms, with implicit trust assumptions applied to internal entities. As applications evolved toward distributed deployments and cloud-native infrastructure, identity emerged as the primary security control plane through which authentication and authorization are applied across users and services [2], [3]. Zero-trust architectures formalize this transition by treating every request as untrusted and requiring continuous identity verification independent of network location [13]. However, many organizations continue to rely on legacy Identity and Access Management (IAM) platforms designed for monolithic systems [2], [7]. These proprietary platforms often depend on stateful, session-based security mechanisms and custom credential stores, which introduce architectural and operational friction when applied to modern, multi-tenant enterprise applications.

The central problem addressed in this work is the technical debt and operational risk associated with legacy identity systems that lack support for OAuth 2.0 and OpenID Connect. The absence of these standards restricts integration with modern backend services, single-page applications, and mobile clients that rely on standardized authorization flows and token semantics [5], [12], [16]. Additionally, manual provisioning processes, rather than automated identity lifecycle management using the System for Cross-domain Identity Management (SCIM), increase operational overhead and reduce consistency across tenants [6]. This paper presents a structured, standards-based migration framework for transitioning proprietary legacy identity systems to Okta as an externalized Identity Provider (IdP). The methodology emphasizes secure, tenant-aware data migration and architectural refactoring patterns that enable token-based authentication while preserving service continuity and tenant isolation.

## II. LIMITATIONS OF NON-STANDARD IDENTITY SYSTEMS

Legacy identity platforms that predate cloud-native and distributed architectures exhibit structural limitations that conflict with modern security and scalability requirements [2], [7]. These systems were often designed around centralized control planes, stateful session management, and proprietary authentication mechanisms [2]. When applied to distributed, multi-tenant enterprise environments, such assumptions introduce constraints that affect protocol compatibility, security posture, operational sustainability, and external interoperability [4].

### A. Protocol and Standards Gaps

Non-standard identity systems frequently lack native support for OAuth 2.0 and OpenID Connect, which are foundational to authentication and authorization in modern application ecosystems [5], [16]. Core flows such as Authorization Code, Proof Key for Code Exchange (PKCE), and token introspection are often absent or implemented through proprietary extensions [16], [17]. This limitation is particularly acute for Single-Page Applications (SPAs) and mobile clients, which depend on standardized redirect-based flows and short-lived tokens rather than server-managed sessions [12], [16]. In the absence of consistent token issuance and validation semantics, applications are forced to embed custom authentication logic, increasing coupling between clients and identity infrastructure and complicating integration with API gateways and downstream services [5].

### B. Security Weaknesses

Security limitations in proprietary identity systems commonly stem from bespoke credential handling and rigid authentication pipelines [7]. Custom or deprecated hashing algorithms complicate cryptographic governance and increase exposure during credential migration [7]. Session-based authentication models further introduce risk in distributed environments, particularly when sticky sessions or long-lived, non-rotated cookies are required to preserve user context [5]. These approaches undermine stateless service design and increase susceptibility to session fixation and replay attacks [7]. In addition, limited support for multi-factor authentication (MFA) and adaptive authentication restricts the ability to enforce context-aware access controls or respond dynamically to anomalous authentication behavior [3], [13].

International Journal of Core Engineering & Management
Volume-7, Issue-12, 2024 ISSN No: 2348-9510

### C. Operational and Maintenance Burden

From an operational standpoint, non-standard identity systems impose significant maintenance overhead [7]. Custom credential schemas and role stores require specialized expertise and are difficult to scale under increasing authentication workloads [5]. The absence of centralized policy enforcement leads to authorization logic being embedded within individual applications, resulting in inconsistent access controls and reduced auditability [5], [7]. These challenges are compounded by the lack of automated identity lifecycle management, particularly when user provisioning and deprovisioning are handled manually rather than through standardized mechanisms such as SCIM [6]. Over time, this operational complexity increases the total cost of ownership and elevates the risk of configuration drift and orphaned identities [7].

### D. Interoperability Challenges

Interoperability limitations further restrict the viability of proprietary identity platforms in modern enterprise ecosystems [2]. Incomplete support for federation standards such as Security Assertion Markup Language (SAML) and OpenID Connect constrains integration with third-party Software as a Service (SaaS) providers and Business-to-Business (B2B) partners [12], [18]. As organizations adopt hybrid and multi-cloud architectures, the inability of legacy systems to participate in federated identity workflows becomes a critical bottleneck [4]. To compensate, enterprises often introduce custom federation adapters or synchronization scripts, increasing system complexity and weakening security guarantees [2]. In multi-tenant environments, such fragmented integration patterns undermine consistent policy enforcement and complicate compliance and audit processes across tenant boundaries [4].

Table 1 provides a comprehensive architectural comparison between legacy proprietary IAM systems and standards-based identity platforms, illustrating the fundamental limitations that drive migration requirements across protocol support, authorization models, token security, and operational characteristics.

| Dimension | Legacy Proprietary IAM | Standards-Based IAM (Okta) |
|---|---|---|
| Authentication Protocol | Proprietary session-based | OAuth 2.0, OIDC |
| Authorization Model | Embedded RBAC in applications | Claims-based, token-driven |
| Token Format | Opaque, long-lived tokens | Short-lived JWTs with cryptographic signatures |
| Multi-Factor Authentication | Limited or absent | Native adaptive MFA support |
| Identity Lifecycle | Manual provisioning | Automated via SCIM |
| Session Management | Stateful, sticky sessions | Stateless token validation |
| Federation Support | Custom adapters required | Native SAML, OIDC support |
| Client Support | Backend services, limited web | Backend, Web, SPA, Mobile, M2M |
| Operational Overhead | High (custom maintenance) | Low (platform-managed) |
| Multi-Tenant Isolation | Application-level enforcement | Platform-level with tenant-aware claims |

Table I: Legacy IAM vs. Standards-Based IAM Comparison

International Journal of Core Engineering & Management
Volume-7, Issue-12, 2024 ISSN No: 2348-9510

## III. ARCHITECTURAL MODERNIZATION WITH OKTA

Modernizing legacy identity architectures requires the externalization of authentication and authorization concerns into a centralized, standards-based identity layer [2], [21]. This architectural shift decouples identity management from application logic and enables consistent enforcement of security policies across distributed systems [21]. Adopting Okta as the identity platform addresses the structural limitations of proprietary identity systems by standardizing protocol usage, enabling token-based authorization, and providing flexible isolation models suitable for multi-tenant enterprise applications [2], [24].

### A. Rationale for Selecting Okta

Okta is selected as the identity platform due to its native support for open identity standards and its ability to operate as a centralized control plane across heterogeneous application environments [24]. Support for OAuth 2.0, OpenID Connect, and SCIM enables consistent authentication, authorization, and identity lifecycle management across backend services, web applications, single-page applications, and mobile clients [6], [12], [16]. This standards-based approach reduces the need for custom authentication logic and minimizes coupling between applications and identity infrastructure [2]. By externalizing identity concerns, organizations can simplify application architecture while relying on a platform designed to provide high availability, auditability, and policy-driven access control at enterprise scale [24].

### B. Standardization on OAuth 2.0 and OpenID Connect

Standardizing authentication and authorization on OAuth 2.0 and OpenID Connect enables a shift from stateful, session-based security models to stateless, token-based authorization [14], [16]. Scopes and claims provide a structured mechanism for expressing authorization intent and contextual identity attributes, allowing services to make access decisions without maintaining server-side session state [11], [14]. This approach improves horizontal scalability and resilience in distributed environments by eliminating dependencies on shared session stores [22]. Structured grant types further enable consistent handling of interactive user authentication and delegated access, reducing variability across client implementations and improving the overall security posture of the system [16].

### C. Multi-Tenant Logical Isolation Models

Multi-tenant enterprise applications require strict isolation between tenants while maintaining operational efficiency [4]. Okta supports multiple isolation models that can be selected based on regulatory, security, and scalability requirements [24]. A multi-organization strategy provides strong physical isolation by segregating tenants into separate identity domains, at the cost of increased administrative complexity [4]. Alternatively, a single organization model with custom authorization servers enables logical isolation by embedding tenant-aware attributes into issued tokens and enforcing them consistently across applications and services [24]. Group- and schema-based isolation mechanisms further support tenant-specific access control and attribute management without duplicating identity infrastructure [4]. These patterns allow organizations to balance isolation guarantees with operational simplicity while preserving a unified identity architecture [21].

International Journal of Core Engineering & Management
Volume-7, Issue-12, 2024 ISSN No: 2348-9510

## IV. SECURE IDENTITY MIGRATION FRAMEWORK

Migrating from proprietary IAM platforms to a standards-based identity architecture introduces risks that extend beyond data movement [1], [2]. Identity data is security-sensitive, tightly coupled to authorization semantics, and often shared across heterogeneous applications and tenants [7]. This section presents a structured migration framework that classifies migration strategies, defines schema and credential transformation patterns, and addresses authorization and scale considerations [1]. The framework emphasizes tenant isolation, service continuity, and operational safety, enabling incremental adoption rather than disruptive replacement [19], [21].

### A. Migration Strategy Classification

Identity migration is not a uniform process and must be tailored to the constraints of the legacy system, regulatory requirements, and availability expectations [1]. Migration strategies can be broadly classified based on the degree of coexistence required between legacy and target identity systems [1]. Bulk migration with coordinated cutover is appropriate when credential formats are compatible, and downtime can be tightly controlled [1]. Incremental migration strategies, such as tenant-by-tenant or application-scoped cutovers, reduce blast radius and enable progressive validation in multi-tenant environments [4]. Just-in-time migration during authentication defers credential movement until first use, minimizing upfront disruption but extending the coexistence window [1], [6]. Strategy selection requires balancing risk, user experience, and operational complexity, and often results in different approaches being applied across tenants or application domains [1].

### B. User and Tenant Schema Mapping

Schema mapping defines how legacy identity attributes are translated into the target identity model while preserving tenant boundaries [2]. Legacy systems frequently store user attributes in relational schemas or flat files with inconsistent naming, optional fields, and embedded tenant context [7]. Migration requires normalizing these attributes into a structured profile schema and explicitly modeling tenant identifiers as first-class attributes [2], [4]. Tenant-specific metadata, such as configuration flags or branding information, must be separated from user identity to avoid schema overloading [4]. Careful schema design is critical to ensure that tenant isolation is enforced consistently by downstream authorization policies and that future tenant onboarding does not require schema refactoring [4].

### C. Credential Migration Strategies

Credential migration represents the highest-risk phase of identity modernization due to the sensitivity of password material and the diversity of legacy hashing mechanisms [7]. When legacy systems use supported hashing formats, credentials can be imported directly without user interaction [7]. For proprietary or unsupported hashes, transitional strategies are required, including just-in-time credential capture during authentication or enforced password reset boundaries [3]. Just-in-time approaches reduce user disruption but require extended coexistence and careful monitoring to ensure complete migration [1]. Forced resets provide stronger security guarantees but increase short-term friction [3]. Regardless of strategy, credential handling must minimize exposure, enforce strict access controls, and ensure that partial migrations do not weaken the overall security posture [13].

### D. Authorization and Group Migration

Authorization migration is not a direct data transfer exercise but an architectural realignment [20]. Legacy systems commonly embed authorization logic within application code or maintain tightly coupled role stores that do not translate cleanly to token-based models [5]. Migration requires decomposing coarse-grained roles into scopes and claims that express authorization intent explicitly [20], [11]. Group structures may be mapped directly where semantics align, but in many cases require consolidation or restructuring to avoid excessive claim proliferation [20]. Authorization logic should be decoupled from identity storage and enforced consistently at resource servers and gateways [11], [14]. In multi-tenant environments, all authorization attributes must be tenant-scoped to prevent cross-tenant access through shared roles or groups [4].

### E. High-Volume Migration and API Management

Enterprise identity migrations often involve large user populations and must be executed through API-driven processes to ensure consistency and auditability [19]. High-volume migrations require careful management of rate limits, batch sizing, and retry behavior to avoid service degradation [19]. Idempotent operations and backoff strategies are necessary to handle transient failures without duplicating identities or corrupting state [21]. Continuous monitoring and reconciliation are required to validate migration completeness and detect discrepancies between legacy and target systems [1]. These controls ensure that migration activities remain observable, repeatable, and safe to execute at scale [21].

## V. APPLICATION INTEGRATION WITH OKTA ACROSS CLIENT TYPES

Modern identity modernization succeeds only when applications are refactored to delegate authentication to an external identity provider while enforcing authorization consistently [2], [21]. In the steady-state architecture, applications rely exclusively on standardized token-based authentication and centralized policy enforcement rather than proprietary session management or embedded credential validation [5], [14]. This section examines application integration patterns across backend services, web and single-page applications, mobile clients, and enterprise Java applications within a standards-based identity architecture [16].

### A. Backend Services and APIs

Backend services operate as protected resource servers that validate access tokens issued by the identity platform rather than maintaining authentication state [11], [16]. Services validate JSON Web Tokens (JWT) using public signing keys obtained from the JSON Web Key Set endpoint, enforcing authorization through validation of token audience and required scopes [14], [15]. This model enables stateless request handling, horizontal scalability, and uniform enforcement of access policies across distributed services [14], [22]. By externalizing authentication and standardizing token validation, backend services eliminate dependencies on shared session stores and proprietary identity libraries [5]. This approach enables consistent security enforcement across heterogeneous deployment environments and simplifies integration with API gateways and service meshes that assume OAuth 2.0 compliant tokens for downstream authorization decisions [11], [16].

### B. Web Applications and Single-Page Applications

Web applications and single-page applications integrate with Okta using redirect-based authorization flows that externalize user authentication and session management [12], [16]. The Authorization Code flow with PKCE is used to protect public clients, mitigating authorization code interception risks [16], [17]. Access and refresh tokens enable silent re-authentication while maintaining short-lived access semantics that reduce exposure in the event of token compromise [14]. Logout flows require coordinated termination at both the application and identity provider layers to ensure consistent invalidation of authentication state [12]. By eliminating server-managed sessions and relying on standardized token flows, web applications reduce the attack surface associated with persistent cookies and improve resilience in distributed and cloud-native deployment models [5], [14].

### C. Mobile Applications

Mobile applications require authentication flows that account for untrusted execution environments and secure token handling [17]. Native authentication is performed through redirect-based flows using system browsers or embedded user agents, preventing direct exposure of credentials to the application runtime [12], [17]. Access and refresh tokens are stored using platform-provided secure storage mechanisms to minimize the risk of credential compromise [12]. Redirect handling and deep linking are configured to ensure that tokens are delivered only to authorized application instances [17]. These integration patterns align mobile authentication with the same token-based security model used by backend and web applications, enabling consistent authorization enforcement across client types [16].

### D. Spring Security 6.x Integration

In enterprise Java environments, Spring Security provides the enforcement layer for authentication and authorization concerns [8], [23], [9]. Modernized applications decommission custom AuthenticationProvider and UserDetailsService implementations backed by proprietary credential stores in favor of standardized configurations using the oauth2ResourceServer and oauth2Login constructs [23]. Authentication is delegated to the external identity provider, and the authorization context is derived directly from JWT claims [14], [23]. Claim mapping is implemented using JwtAuthenticationConverter to translate scopes and custom claims into application-specific authorities [23]. Session-based security filters are removed to enforce stateless request handling and align application behavior with token-based authorization semantics [14]. This configuration enables Spring-based services to integrate cleanly into a distributed identity architecture while preserving consistent security behavior across deployments [8].

## VI. AUTHORIZATION AND POLICY ENFORCEMENT

Modernizing the identity layer necessitates a corresponding evolution in authorization design [2], [21]. Legacy IAM systems commonly rely on Role-Based Access Control (RBAC), where permissions are statically bound to user roles and enforced directly within applications [20]. While effective in centralized systems, this model becomes brittle in distributed and multi-tenant architectures due to role proliferation and inconsistent enforcement [5], [7]. Transitioning to a standards-based identity platform enables a shift toward claims-driven authorization, where access decisions are derived from structured, cryptographically verifiable attributes and evaluated consistently across system boundaries [11], [14].

A. Transition from Role-Based Access Control to Claims-Based Authorization

Legacy authorization models frequently encode coarse-grained roles that aggregate multiple permissions without sufficient contextual separation [20]. In a modernized architecture, these roles are decomposed into scopes and claims that express authorization intent explicitly [11], [14]. Scopes define the operations a principal is permitted to perform, while claims convey contextual attributes such as tenant affiliation or privilege level [14]. Mapping legacy RBAC constructs into these artifacts enables finer-grained control and reduces coupling between authorization logic and identity storage [20]. Claims-based authorization allows resource servers to evaluate access decisions locally using information embedded within tokens, eliminating the need for synchronous calls to centralized role stores [11], [14]. Custom claims may be introduced to represent domain-specific attributes without requiring changes to application authorization logic [14]. In multi-tenant environments, this model ensures that all authorization decisions are evaluated within an explicit tenant context, reducing the risk of cross-tenant access caused by shared or misconfigured roles [4].

**B. Centralized vs. Decentralized Policy Models**

Authorization enforcement can be distributed across multiple architectural layers, each serving a distinct purpose [2]. Centralized policy definition at the identity platform enables consistent application of access rules during token issuance, ensuring that scopes and claims reflect approved access conditions [11]. This approach simplifies downstream services by allowing them to rely on standardized token contents rather than reimplementing identity-aware policy logic [2]. Decentralized enforcement complements this model by allowing API gateways and services to apply additional controls [11]. Gateways commonly enforce coarse-grained checks such as token validity, scope presence, and tenant routing, providing an initial layer of protection [11], [16]. Service-level authorization enables context-aware decisions that depend on resource state or business logic unavailable at token issuance time [21]. Effective architectures combine these approaches to achieve defense in depth, balancing centralized consistency with localized flexibility [13].

**C. Delegated Authorization for Machine-to-Machine Communication**

Service-to-service communication in distributed systems requires authorization mechanisms that operate without user involvement [16]. Delegated authorization for these scenarios is implemented using the client credentials flow, where services authenticate as confidential clients and obtain access tokens representing their own identity [16]. Tokens issued through this flow are scoped explicitly for machine access and include audience restrictions that bind them to specific resource servers [11]. Strict audience validation is essential to maintaining service isolation and preventing lateral movement within a microservices architecture [11]. Each service verifies that incoming tokens were issued for its intended audience and contain only the scopes required for the requested operation [11]. By defining service-specific scopes and enforcing least-privilege access, organizations can limit the impact of compromised credentials while enabling scalable, autonomous service interaction [13].

## VII. MULTI-TENANT ARCHITECTURE PATTERNS WITH OKTA

Multi-tenant identity architectures must enforce strict tenant isolation while preserving the operational efficiencies of shared infrastructure [4], [7]. When identity is externalized, tenant lifecycle management, access control, and observability become platform-level concerns rather

than application-specific responsibilities [2], [21]. Organizations must therefore adopt architectural patterns that define how tenants are provisioned, isolated, and monitored across a centralized identity layer [4]. This section examines these patterns with emphasis on scalability, isolation guarantees, and auditability [7].

**A. Tenant Provisioning and Lifecycle Management**

Automated tenant provisioning is essential for maintaining consistency and scalability in multi-tenant identity systems [6], [4]. Tenant onboarding workflows establish identity constructs such as groups, policies, and application assignments as part of a repeatable lifecycle process, reducing reliance on manual configuration [6]. Identity lifecycle management using SCIM enables standardized provisioning and deprovisioning of user accounts as tenants are created, modified, or retired, ensuring synchronization between identity state and application access [6]. Tenant metadata plays a critical role in lifecycle management and must be modeled explicitly within the identity system [24]. Attributes such as tenant identifiers, configuration flags, and entitlement indicators are maintained independently of user credentials and propagated into issued tokens as claims [11], [24]. This approach enables applications and services to derive tenant context dynamically at runtime, supporting tenant-aware routing and authorization without embedding tenant configuration into application code [4]. Clear separation between tenant metadata and user identity attributes further reduces coupling and simplifies long-term evolution of tenant models [21].

**B. Isolation and Access Control**

Tenant isolation within shared identity infrastructure can be achieved through both physical and logical segmentation models [4]. Strong isolation requirements may be addressed through multi-organization strategies that segregate tenants into distinct identity domains, while logical isolation within a single organization can be enforced through policy segmentation and tenant-aware claims [4], [24]. In the latter model, tokens issued to users and services contain explicit tenant identifiers that are evaluated consistently by gateways and resource servers to prevent cross-tenant access [11], [14]. Policy segmentation complements claim-based isolation by scoping authentication and authorization policies to tenant-specific contexts [24]. This ensures that access requirements, such as authentication strength or session constraints, are enforced independently per tenant even when infrastructure is shared [13]. Audit and logging separation further reinforces isolation by tagging identity events with tenant context, enabling tenant-specific analysis and compliance reporting without exposing cross-tenant activity [7]. Together, these mechanisms provide isolation guarantees comparable to physical separation while maintaining operational efficiency [4].

**C. Observability and Monitoring**

Observability is a foundational requirement for securing multi-tenant identity platforms [7], [13]. Centralized identity event logging provides visibility into authentication attempts, token issuance, policy evaluations, and administrative changes across tenants [7]. These events enable detection of misconfiguration, abuse, and anomalous access patterns that may indicate compromised credentials or policy violations [13]. Integration with enterprise monitoring and analytics systems allows identity events to be correlated with application and infrastructure telemetry [21]. Tenant-aware monitoring enables targeted alerting and investigation while preserving isolation between tenants [7]. Risk-based analysis of authentication behavior, including deviations in access patterns or unexpected token usage, further strengthens detection capabilities [13]. By combining

centralized identity logging with downstream analytics, organizations can maintain continuous security visibility across a distributed tenant population without sacrificing tenant isolation [7].

## VIII. PHASED MIGRATION AND PRODUCTION DEPLOYMENT

Migrating enterprise identity infrastructure requires a phased deployment strategy that minimizes operational risk while preserving service availability and tenant isolation [1], [21]. Identity systems are deeply embedded across applications and services, making abrupt transitions impractical [2]. A staged approach enables progressive validation of authentication, authorization, and provisioning behavior while maintaining clear rollback boundaries [1], [19]. This section outlines a production-oriented migration lifecycle spanning assessment, coexistence, controlled cutover, and post-migration hardening [1].

### A. Pre-Migration Assessment

Migration begins with a comprehensive assessment of the legacy IAM landscape [1]. This phase inventories identity components, including credential stores, authentication endpoints, authorization logic, and integration dependencies across applications and external systems [2]. Dependency mapping identifies hidden couplings, such as application-specific authentication assumptions or protocol limitations, that influence migration complexity [21]. Risk analysis categorizes applications and tenants based on criticality, availability requirements, and regulatory constraints [1]. The outcome of this assessment informs migration sequencing and enables early identification of workloads suitable for pilot adoption versus those requiring extended coexistence [1].

### B. Coexistence and Shadow Mode

To maintain service continuity, Okta is introduced in parallel with the legacy IAM system during an initial coexistence phase [1]. Applications are incrementally adapted to accept standardized tokens while legacy authentication paths remain operational [16], [10]. Token bridging mechanisms normalize identity context across authentication sources, allowing services to enforce authorization consistently regardless of token origin [11]. Shadow mode further enhances safety by validating Okta-issued tokens alongside legacy flows without enforcing them, enabling observation of behavior, claim mapping accuracy, and policy alignment [1]. This phase supports tenant-specific migration timelines and reduces risk by allowing functional validation without user-facing impact [1].

### C. Traffic Shifting and Cutover

Once coexistence stability is established, traffic shifting enables a controlled transition to Okta-based authentication [1]. Routing decisions are commonly performed at the gateway or ingress layer, directing authentication and authorization flows based on tenant, application, or request attributes [4]. This granularity supports incremental cutover and limits blast radius by isolating migration impact [1]. Zero-downtime transition is achieved by relying on stateless token validation and configuration-based routing rather than synchronized service outages [14], [16]. Clear rollback paths remain available until all tenants and applications are fully transitioned [1].

### D. Post-Migration Hardening

After migration is complete, post-migration hardening ensures long-term security and operational stability [19]. Legacy IAM components are formally decommissioned to eliminate redundant authentication paths and reduce attack surface [19]. Transitional policies and compatibility mechanisms introduced during coexistence are removed, and tenant isolation models are finalized through consistent claim-based enforcement [24]. Security posture validation confirms that authentication flows, token lifetimes, revocation strategies, and authorization enforcement align with defined security requirements [13]. Ongoing monitoring and periodic reviews ensure that the identity platform continues to meet evolving application and tenant needs [7].

## IX. RESULTS AND DISCUSSION

The migration from a proprietary IAM platform to a standards-based identity architecture results in meaningful improvements across security posture, operational efficiency, and system scalability [3], [7]. Rather than presenting vendor-specific metrics, this section discusses outcomes in terms of architectural properties and operational behaviors observed after externalizing identity in a multi-tenant enterprise environment [2], [21].

### A. Improvements in Security Posture

The most significant outcome of the migration is an elevated security baseline [3], [13]. Legacy identity systems commonly relied on password-centric authentication combined with server-managed sessions or proprietary, opaque tokens that were long-lived and tightly coupled to the issuing platform [7]. These mechanisms limited independent verification, complicated revocation, and increased exposure to replay, fixation, and token misuse attacks [7]. Transitioning to a standards-based model replaces these constructs with short-lived, cryptographically signed JWTs, enabling consistent validation of identity context at each request boundary [14]. Centralized policy enforcement and claims-based authorization further improve correctness and auditability by deriving access decisions from standardized, verifiable attributes rather than application-embedded logic [11], [14]. The use of MFA and adaptive authentication policies strengthens resilience against credential compromise through contextual enforcement based on access risk and tenant requirements [3], [13]. Collectively, these changes align access control more closely with zero-trust principles by treating every request as untrusted and continuously evaluated [13].

### B. Reduction in Operational Overhead and Maintenance

Externalizing identity services reduces operational burden by eliminating custom authentication logic, proprietary credential stores, and application-specific authorization mechanisms [5], [7]. Standardized protocols and shared integration patterns remove duplicated security implementations, enabling teams to rely on uniform token validation and policy evaluation across services [14], [16]. Automated provisioning and lifecycle management replace manual workflows, reducing configuration drift and the risk of orphaned or over-privileged accounts [6], [7]. Decommissioning legacy identity infrastructure further simplifies operations by removing the need to maintain high-availability authentication components and bespoke synchronization processes [19]. Consolidating identity and policy management into a centralized platform enables security and compliance teams to reason about access behavior through a single control plane, improving maintainability and reducing long-term operational cost [2], [21].

### C. Scalability and Performance Gains in Multi-Tenant Environments

The transition to stateless token validation removes authentication bottlenecks associated with centralized session stores and synchronous database lookups [14], [22]. Backend services and APIs scale horizontally without coordination overhead, as authorization decisions are derived directly from token contents [14]. This supports predictable performance under increasing authentication and authorization load [22]. In multi-tenant environments, tenant-aware claims and policy segmentation enforce isolation without duplicating infrastructure per tenant [4], [24]. Authorization context is evaluated consistently at gateways and resource servers, reducing contention and simplifying routing [11]. These properties allow the platform to scale in both tenant count and request volume while preserving isolation guarantees and operational stability [4].

### D. Architectural Trade-offs and Migration Considerations

Despite these benefits, the architecture introduces trade-offs that require deliberate management [21]. Claims-based authorization shifts complexity toward token design and policy modeling, necessitating governance to avoid over-privileged scopes or excessive claim proliferation [11], [14]. Authorization policies evaluated at token issuance time require careful selection of token lifetime, refresh, and revocation strategies to ensure timely enforcement of access changes [16]. Embedding rich authorization context within tokens increases token size, which can introduce network overhead, approach intermediary header limits, and impact performance in high-throughput environments [14]. These constraints require disciplined claim minimization and avoidance of transient or non-essential data [21]. Additionally, coexistence with legacy identity systems during migration introduces temporary operational complexity that must be actively governed to prevent persistent dual-stack security models [1]. Multi-tenant isolation strategies further involve trade-offs between administrative overhead and isolation strength, requiring careful balancing of operational efficiency and security requirements [4], [21].

## X. CONCLUSION

This paper presents a structured framework for migrating legacy identity systems to a standards-based identity architecture using Okta in multi-tenant enterprise environments [1], [2], [24]. The framework addresses the full lifecycle of identity modernization, including architectural refactoring, secure data and credential migration, application integration across heterogeneous clients, authorization redesign, and phased production deployment [1], [21]. By separating steady-state architecture from transitional coexistence concerns, the work demonstrates how identity modernization can be executed incrementally while preserving service continuity and tenant isolation [1], [4].

Adopting a standards-based IAM model enables organizations to decouple identity concerns from application logic and infrastructure constraints [2], [21]. Standardized protocols, token-based authentication, and centralized policy definition provide a consistent foundation for enforcing access across distributed systems and diverse client types [11], [14], [16]. This approach simplifies integration, supports tenant-aware authorization, and establishes a scalable identity control plane capable of evolving independently of individual applications [4], [24].

For organizations planning similar modernization efforts, several considerations emerge [21]. Identity migration should be treated as a platform transformation rather than a single-system replacement, with explicit attention to schema design, authorization semantics, and tenant isolation models [2], [4]. Phased deployment strategies reduce risk and enable progressive

validation, while clear governance over token design and policy management is essential to sustain long-term correctness [1], [21]. By approaching identity modernization as an architectural initiative grounded in open standards, organizations can establish a resilient foundation for secure, scalable, and multi-tenant application ecosystems [3], [13].

**REFERENCES**

1. J. Anderson and K. Keahey, "Migrating towards Single Sign-On and Federated Identity," in Practice and Experience in Advanced Research Computing (PEARC '22), Boston, MA, USA, July 2022, pp. 1-12, doi: 10.1145/3491418.3530770.
2. D. Pöhn and W. Hommel, "Reference Service Model Framework for Identity Management," IEEE Access, vol. 10, pp. 120984-121009, 2022, doi: 10.1109/ACCESS.2022.3219044.
3. P. Phiayura and S. Teerakanok, "A Comprehensive Framework for Migrating to Zero Trust Architecture," IEEE Access, vol. 11, pp. 19487-19511, Feb. 2023, doi: 10.1109/ACCESS.2023.3248622.
4. C. Zheng, Q. Zhuang, and F. Guo, "A Multi-Tenant Framework for Cloud Container Services," in 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, July 2021, pp. 359-369, doi: 10.1109/ICDCS51616.2021.00042.
5. M. G. de Almeida and E. D. Canedo, "Authentication and Authorization in Microservices Architecture: A Systematic Literature Review," Applied Sciences, vol. 12, no. 6, p. 3023, Mar. 2022, doi: 10.3390/app12063023.
6. T. Baumer, M. Müller, and G. Pernul, "System for Cross-Domain Identity Management (SCIM): Survey and Enhancement With RBAC," IEEE Access, vol. 11, pp. 86872-86894, Aug. 2023, doi: 10.1109/ACCESS.2023.3304270.
7. A. Badirova et al., "A Survey on Identity and Access Management for Cross-Domain Dynamic Users: Issues, Solutions, and Challenges," IEEE Access, vol. 11, pp. 61660-61679, 2023, doi: 10.1109/ACCESS.2023.3279492.
8. A. Chatterjee and A. Prinz, "Applying Spring Security Framework with Keycloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study," Sensors, vol. 22, no. 5, p. 1703, Feb. 2022, doi: 10.3390/s22051703.
9. Spring Security Team, "Spring Security 6.3.0 Released," Spring.io Blog, May 21, 2024. [Online]. Available: https://spring.io/blog/2024/05/21/spring-security-6-3-0-released.
10. [Okta, "Okta Identity Engine Release Notes," Okta Developer Documentation, 2023. [Online]. Available: https://developer.okta.com/docs/release-notes/.
11. E. Rescorla and B. Korver, "RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage," IETF, Oct. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6750.txt.
12. N. Sakimura et al., "OpenID Connect Core 1.0 incorporating errata set 2," OpenID Foundation, Nov. 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html.
13. J. Kindervag et al., "Zero Trust Architecture," NIST Special Publication 800-207, Aug. 2020, doi: 10.6028/NIST.SP.800-207.

14. E. Rescorla, "RFC 7519: JSON Web Token (JWT)," IETF, May 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7519.txt.
15. M. Jones, J. Bradley, and N. Sakimura, "RFC 7517: JSON Web Key (JWK)," IETF, May 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7517.txt.
16. D. Hardt, "RFC 6749: The OAuth 2.0 Authorization Framework," IETF, Oct. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6749.txt.
17. J. Bradley and N. Sakimura, "RFC 7636: Proof Key for Code Exchange by OAuth Public Clients," IETF, Sept. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7636.txt.
18. A. Nadalin et al., "Security Assertion Markup Language (SAML) V2.0 Technical Overview," OASIS Standard, Mar. 2008.
19. L. Bass, I. Weber, and L. Zhu, "DevOps: A Software Architect's Perspective," Addison-Wesley, Boston, MA, USA, 2015.
20. R. Sandhu et al., "Role-Based Access Control Models," IEEE Computer, vol. 29, no. 2, pp. 38-47, Feb. 1996, doi: 10.1109/2.485845.
21. M. Richards and N. Ford, Fundamentals of Software Architecture: An Engineering Approach, 1st ed., Sebastopol, CA, USA: O'Reilly Media, 2020.
22. M. Kleppmann, Designing Data-Intensive Applications, 1st ed., Sebastopol, CA, USA: O'Reilly Media, 2017.
23. Spring Security Team, Spring Security Reference Documentation 6.x, VMware, 2023. [Online]. Available: https://docs.spring.io/spring-security/reference/.
24. Okta, Okta Identity Engine Concepts and Architecture, Okta Developer Documentation, 2023. [Online]. Available: https://developer.okta.com/docs/concepts/identity-engine/.