International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# Secure Identity Governance Architecture for Cloud-Native Enterprise Platforms in Regulated Industries

**Aravind Yedmala**

**Abstract**

Regulated enterprises are moving core business functions onto distributed, application programming interface (API) driven platforms, and in such platforms an access decision is no longer made once at a point of login. Each microservice, batch job, and cloud workload evaluates whether a caller is the identity it claims to be and whether that identity is permitted to perform the requested action. Strong authentication alone does not establish why access was granted, whether it remains appropriate, who approved it, or when it was last reviewed, and these are the questions a regulated organization is required to answer. This article develops a vendor-neutral reference architecture for cloud-native identity governance by synthesizing normative standards, primary protocol specifications, and peer-reviewed research on cloud-native access control. Unlike Zero Trust guidance, which specifies how a request is verified, and conventional identity and access management, which specifies how a credential is issued, the architecture's distinguishing claim is a governance plane that holds entitlement and lifecycle state independently of the authorization plane that issues tokens-a separation that makes an entitlement reviewable, recertifiable, and revocable without touching the credentials that carry it. The architecture organizes identity governance into six planes and three flows connecting them, situates role-based and attribute-based access control, OAuth 2.0, OpenID Connect, JSON Web Token validation, policy decision and enforcement points, API gateway controls, service mesh authentication, and workload identity federation via SPIFFE/SPIRE within that structure, and traces one entitlement end to end-from approval through enforcement to scheduled review and revocation-as a worked illustration of the model. A discrete-event simulation, built for this article and reported with its parameters and code, estimates policy-evaluation overhead and revocation-propagation latency under stated assumptions; it is a simulation of the model's behavior, not a measurement of a deployed system, and is reported on that basis. Four contributions are offered: the plane separation itself; a layered authorization model distinguishing authentication from role, organization, object, action, and field-level authorization; the treatment of workload identity as a governed lifecycle rather than an authentication mechanism; and worked applications to healthcare, financial services, and insurance settings. The model is intended to support the production of compliance evidence; it does not by itself establish regulatory compliance, which depends on organizational processes and controls beyond the technical architecture.

**Keywords:** Access Control, Auditability, Cloud-Native Security, Identity Governance, Workload Identity, Zero Trust Architecture

## I. Introduction

In regulated enterprises, strong authentication is insufficient when the organization cannot demonstrate why access was granted, whether it remains appropriate, who approved it, and when it was reviewed or revoked. This failure mode is less visible than an authentication breach and becomes harder to avoid as platforms move from a small number of monolithic applications to dozens or hundreds of independently deployed services. Every employee, customer administrator, third-party integration, service account, and cloud workload becomes a potential point of unexplained access where no consistent model governs which actors may perform which operations, and on what basis.

Access management asks a narrow question: can this actor authenticate? Identity governance asks a wider set. Should this user still hold the access granted to them eighteen months ago? Is the entitlement scoped to the correct organization or business unit? Was the access approved through a defined process, and can that approval be reconstructed later? Can the access be revoked promptly when the underlying business

*Independent Researcher, USA*

relationship ends? These questions connect identity management to risk management and to compliance reporting, and they are answered by recorded state rather than by a successful login.

Cloud-native modernization increases the difficulty of answering them. Legacy platforms often concentrated access decisions inside one or two applications sharing a database and a small number of administrative interfaces. A modernized platform distributes the same functionality across APIs, background workers, event consumers, integration jobs, and third-party connectors, each of which can become a point of exposure where access is granted without a governing model. Identity therefore becomes a shared control plane, because no single application boundary contains the risk. Treating identity governance as a foundational design concern rather than a feature added to individual applications is what allows modernization to proceed without accumulating access that cannot be explained.

### *A. What Is New Here*

Three bodies of existing guidance each cover part of this problem, and none covers the seam between them. Zero Trust standards and surveys specify how a request is verified at the moment it is made-continuous authentication, resource-centered decisions, trust evaluation-but treat entitlement administration as an organizational process outside the architecture's scope [1], [2], [15], [16]. Conventional identity and access management specifies how a credential is issued and authenticated, including federation and proofing, but stops at the point of login and does not model what happens to an entitlement afterward [3]. And cloud-native access-control research documents which authorization models are used in practice and where enforcement is commonly placed, without prescribing a structural separation between the state that says an entitlement should exist and the mechanism that evaluates whether a presented credential may act on it now [19].

The contribution of this article is that separation, stated as an architectural claim rather than an operational recommendation: a governance plane holding entitlement and lifecycle state, kept structurally distinct from the authorization plane that issues tokens and evaluates requests. This is not a restatement of Zero Trust's "never trust, always verify" principle-that principle governs the runtime request flow addressed in Section III.A. It is a claim about a second flow, developed in Section III.B, that Zero Trust guidance does not model: how an approved entitlement becomes enforceable policy, and how it is reviewed, recertified, or revoked independently of whether the credentials carrying it happen to still be valid. Four contributions follow from this separation and are developed in the sections below.

1)The six-plane structure itself-human identity, workload identity, governance, authorization, enforcement, and evidence-in which the governance plane is separable from the authorization plane in a specific, testable sense: an entitlement can be revoked without reissuing or invalidating any credential. Section III.D traces one entitlement through this structure end to end, from approval to enforcement to review to revocation, as a concrete illustration of what the separation means operationally.

2)A specification of three distinct flows across those planes: the runtime request flow from authentication to enforcement, the governance flow through which approved entitlements become enforceable policy, and the audit flow through which each decision becomes evidence and feeds the next access review.

3)A layered authorization model that separates authentication from role, organization, object, action, and field-level authorization, addressing the object-level and function-level authorization failures that recur in API deployments.

4)The treatment of workload identity as a governed lifecycle-registration, ownership, attestation, issuance, rotation, review, and decommissioning-rather than as an authentication mechanism, together with an explicit statement of what transport authentication does and does not decide.

The article also applies the model to healthcare, financial services, and insurance settings in Section VI, and reports a discrete-event simulation of policy-evaluation overhead and revocation propagation in Section VIII, alongside the limitations that follow from the architecture not having been implemented as a production system.

### *B. Source Selection and Use*

This is a conceptual architecture article, supplemented by a simulation described in Section VIII; it reports no production deployment or field measurement, and the architecture itself should be read as a design proposal. Sources were identified from the normative catalogues of the standards bodies concerned and from searches of IEEE Xplore, the ACM Digital Library, and MDPI-indexed venues, using terms combining identity governance, access control, Zero Trust, workload identity, API authorization, and cloud-native security.

Sources fall into three categories, and the current normative version was preferred in each case. Normative standards supply governing principles and vocabulary: NIST SP 800-207 for Zero Trust architecture [1], SP 800-207A for its extension to cloud-native microservice environments [2], SP 800-63-4 for human identity proofing, authentication, and federation [3], and SP 800-162 for attribute-based access control [4]. Primary protocol specifications supply normative mechanism definitions, cited directly rather than through secondary description: the OAuth 2.0 security best current practice [5], OpenID Connect Core [6], the JSON Web Token specification [7] and its best-practice companion [8], and OAuth 2.0 token exchange [9]. Practitioner-maintained security guidance is used only where it documents observed vulnerability classes, in the case of the OWASP API Security Top 10 [10]. Workload identity specifications and platform documentation are cited where they define the mechanisms the architecture depends upon: the SPIFFE identity and verifiable identity document specification [11], the SPIFFE Workload API [12], the SPIRE runtime environment [13], and Kubernetes service-account handling [14].

Peer-reviewed research supplies empirical grounding and observed failure modes. Two surveys establish the state of Zero Trust research and its open problems [15], [16]. A systematization of Kubernetes security practice supplies evidence on recurring configuration weaknesses [17]. A study extending SPIRE to confidential workloads shows how verifiable workload identity integrates with trusted execution environments [18]. A systematic mapping study of access control in cloud-native architecture establishes what models are used in practice and where enforcement is placed [19], and a systematic literature review of inter-service security threats establishes the relative maturity of transport security compared with service-to-service authorization [20].

No production measurements were collected for this article. Where a statement describes observed behavior in the field, it is attributed to a cited study; where it describes a simulated result, it is drawn from the discrete-event model reported in Section VIII with its parameters and seed; where it describes a design consequence, that consequence follows from the structure of the mechanism rather than from measurement. The comparative tables should be read on that basis.

Section II distinguishes access management from identity governance and positions the article against existing work. Section III presents the reference architecture, its six planes, its three flows, and a worked entitlement lifecycle. Section IV examines API authorization and token handling. Section V develops workload identity and federation. Section VI applies the model to three regulated settings. Section VII addresses operational scale and auditability. Section VIII reports the simulation and states the limitations. Section IX concludes.

## II. Background and Related Work

### *A. From Access Management to Identity Governance*

Identity and access management (IAM), in its narrower historical sense, covers authentication, single sign-on, and account administration: it establishes whether an actor can gain entry. Identity governance builds on that foundation and addresses what follows-how the entitlement was assigned, whether it remains appropriate, who reviewed it, and how it will be removed when no longer required. The difference is one of state: IAM evaluates a credential at a point in time, whereas identity governance maintains a relationship between a person, a role, an organization, and a set of permissions over time.

The practical consequence is a shift from static role assignment toward policy-aware, context-aware authorization. In many enterprise platforms the same person occupies different administrative scopes depending on context. An administrator may manage users for one client organization while having no visibility into a second organization served by the same platform. A customer-facing user may be authorized for one application module and blocked from another reached through the same session. A backend service may read a record but never modify or delete it. No single static role label expresses these distinctions; they require role-based access control (RBAC) for broad job function, attribute-based access control (ABAC) for contextual properties evaluated at decision time [4], token claims that carry organizational scope, and a policy engine able to evaluate that scope at the moment of the request rather than only at login [15], [19].

Access lifecycle management gives this flow its shape: onboarding, role assignment and approval, active application access, periodic access review, and suspension or de-provisioning when the relationship changes. Each lifecycle event should leave an auditable trace-who requested the access, who approved it, which policy governed the approval, and what occurred when the access was revoked. The objective is not only to grant access efficiently but to make it explainable afterward, revocable on short notice, and reviewable on a defined cadence.

Table I contrasts the two approaches across the dimensions on which they differ operationally.

**TABLE I.** LEGACY ACCESS MANAGEMENT AND IDENTITY GOVERNANCE COMPARED

| Dimension | Legacy access management | Identity governance |
|---|---|---|
| Provisioning | Manual and ticket-driven | Automated through defined lifecycle workflows |
| Role model | Static and application-specific | Policy-driven and organization-scoped |
| Audit trail | Fragmented across applications, inconsistent in format | Centralized, structured, and joined by correlation identifiers |
| Access review | Ad hoc or absent | Scheduled certification cycles with recorded outcomes |
| Revocation | Manual de-provisioning, variable in latency | Policy-triggered, with a defined propagation objective |

| Dimension | Legacy access management | Identity governance |
|---|---|---|
| Scope granularity | Coarse, typically at application level | Fine-grained, extending to object and field level |
| Observability | Limited to individual applications | Cross-service, with decisions recorded in a consistent format |

*The left column characterizes conditions that arise where governance is not deliberately specified, rather than choices made explicitly. The comparison is qualitative.*

### *B. Related Work and Positioning*

Existing work addresses parts of this problem; the contribution of the present article lies in the connections between them rather than in any single mechanism. The two Zero Trust surveys are comprehensive on principles, architectural components, trust evaluation, and adoption obstacles [15], [16], but treat identity primarily as an authentication and continuous-verification input, without developing the entitlement lifecycle of approval, certification, and revocation that a regulated organization must evidence. NIST SP 800-207 and SP 800-207A are normative on resource-centered access decisions and on establishing trust between distributed microservices [1], [2], and are deliberately silent on how entitlements are administered and certified, which they treat as organizational process rather than architecture. SP 800-63-4 is authoritative on human identity proofing, authentication, and federation [3] and does not address workload identity or entitlement governance.

On the cloud-native side, the systematic mapping study of access control in cloud-native architecture reports that selecting a model and placing enforcement correctly are recurring difficulties [19], without prescribing a governance plane distinct from authorization or an evidence model. The systematic literature review of inter-service security threats finds transport-level protection comparatively well covered in the literature while service-to-service authorization is less developed [20], corroborating the gap this article addresses at the enforcement plane. The Kubernetes security systematization [17] and the SPIRE extension study [18] operate at platform and mechanism level respectively, and neither addresses the lifecycle governance of the identities they secure.

The present article differs in three respects: it treats human and workload identity within one control plane rather than as separate programs; it separates the governance plane from the authorization plane so that entitlement state is reviewable independently of token issuance; and it makes the evidence plane a first-class element with a defined feedback path into access certification, rather than treating audit as a logging concern. What it does not offer, and what the cited surveys do offer, is empirical breadth across production deployments. Table II summarizes this positioning.

**TABLE II.** POSITIONING RELATIVE TO EXISTING WORK

| Source category | Sources | Primary focus | Aspect not addressed |
|---|---|---|---|
| Zero Trust surveys | [15], [16] | Principles, components, trust evaluation, and adoption obstacles | Entitlement lifecycle, approval, and certification |
| Zero Trust standards | [1], [2] | Resource-centered access decisions; trust between distributed microservices | Entitlement administration and evidence modelling |
| Digital identity guidance | [3] | Human identity proofing, authentication, and federation | Workload identity and entitlement governance |
| Cloud-native access-control mapping | [19] | Access-control models in use and where enforcement is placed | A governance plane distinct from authorization |

| Source category | Sources | Primary focus | Aspect not addressed |
|---|---|---|---|
| Inter-service security review | [20] | Inter-service threats and mitigations; transport security comparatively mature | Depth of service-to-service authorization |
| Platform and mechanism studies | [17], [18] | Kubernetes security practice; verifiable workload credentials | Lifecycle governance of the identities secured |
| This article | — | Six-plane governance architecture uniting human and workload identity, with an evidence feedback path into certification | Empirical evaluation, which is proposed in Section VIII rather than performed |

## III. Reference Architecture for Cloud-Native Identity Governance

The architecture separates six planes rather than concentrating identity in a gateway or a directory service. The human identity plane establishes the identity of a human actor, whether authenticating directly or through federation, consistent with the human-identity guidance in SP 800-63-4 [3]. The workload identity plane performs the equivalent function for non-human actors, using a workload identity provider and an attestation service to establish verifiable identity for containers, batch jobs, and service accounts. The governance plane holds entitlement and lifecycle state: the entitlement catalog, lifecycle management from onboarding through de-provisioning, approval workflows, and scheduled access review. The authorization plane issues tokens carrying scopes, audiences, and expiration rules, and evaluates access requests through a policy decision point (PDP) against policy derived from the governance plane. The enforcement plane applies decisions at two points: an API gateway performing coarse-grained validation at the edge, and a service-level enforcement point-a policy enforcement point (PEP) located in the service-applying fine-grained authorization close to the protected resource. The evidence plane captures audit records, authorization decision logs, and distributed traces.

The separation of the governance plane from the authorization plane is the load-bearing decision. Governance answers whether an entitlement should exist; authorization answers whether a request presenting a credential may proceed now. Combining them makes entitlement review dependent on token infrastructure and obscures the distinction between an entitlement that was never approved and one that was approved and has since lapsed. Table III states the planes, their representative components, and their functions, using generic component labels so that the model reads as a reference architecture rather than a description of one implementation. Fig. 1 shows the same six planes together with the three flows that connect them; Section III.D then traces one entitlement across all three flows as a single worked example.

**TABLE III.** SIX-PLANE REFERENCE ARCHITECTURE FOR CLOUD-NATIVE IDENTITY GOVERNANCE

| Plane | Representative components | Primary function |
|---|---|---|
| Human identity plane | Identity provider, federation service | Establishes the identity of a human actor and, where relevant, organizational context |
| Workload identity plane | Workload identity provider, attestation service | Establishes verifiable identity for services, containers, and scheduled workloads |

| Plane | Representative components | Primary function |
|---|---|---|
| Governance plane | Entitlement catalog, lifecycle management, approval workflows, access review, revocation | Holds and administers entitlement and lifecycle state, independently of token issuance |
| Authorization plane | Authorization server, policy store, policy decision point | Issues scoped tokens and evaluates requests against policy derived from the governance plane |
| Enforcement plane | API gateway, service-level enforcement points, optional service mesh | Applies coarse-grained validation at the edge and fine-grained authorization at the resource |
| Evidence plane | Audit store, authorization decision logs, distributed tracing | Records actor, action, object, policy, and outcome for every decision |

*The plane names used here are used consistently in the surrounding text and in Fig. 1.*

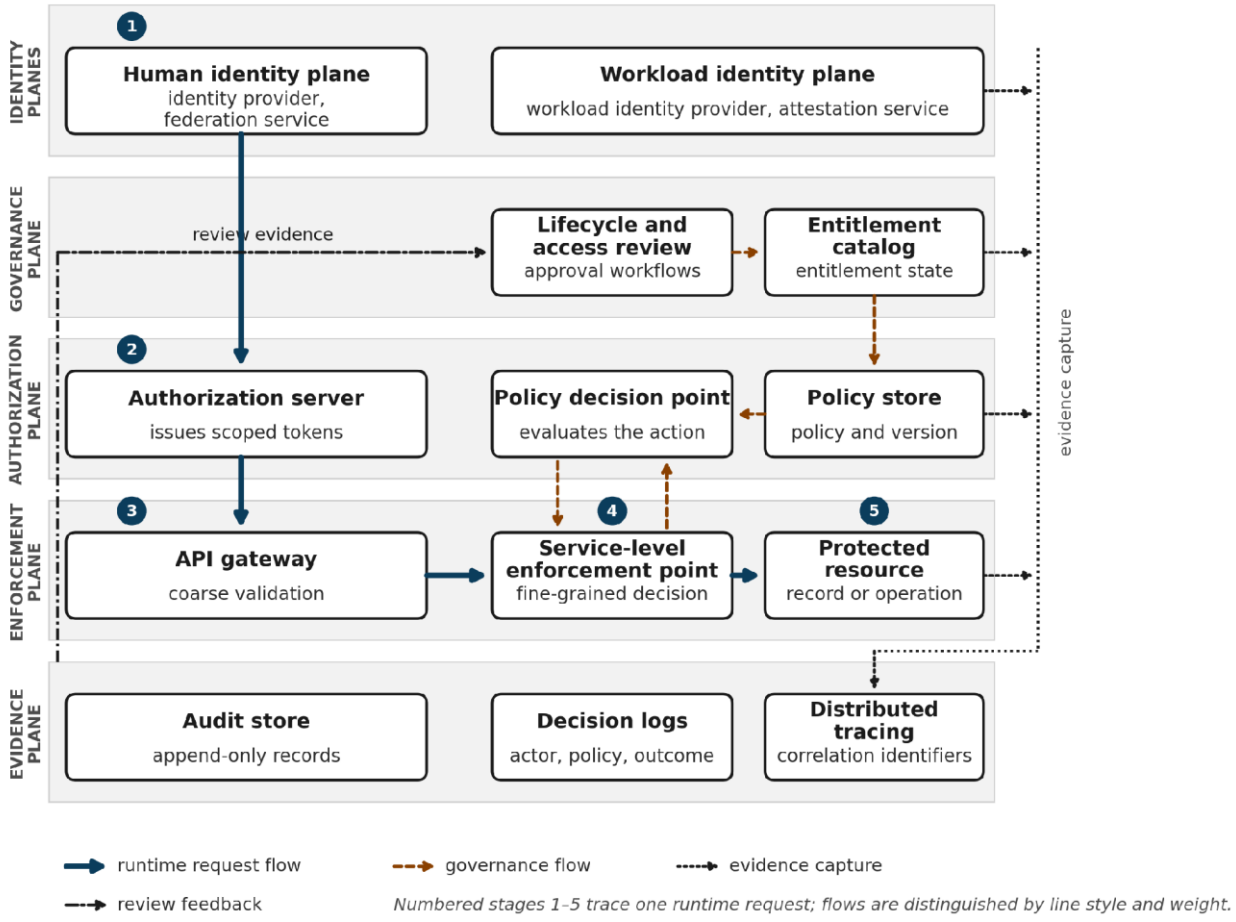


**Fig. 1.** Six-plane cloud-native identity governance architecture, showing the runtime request flow, the governance flow, and the audit flow. Numbered stages 1 to 5 trace one runtime request: authentication in the human identity plane, token issuance in the authorization plane, coarse validation at the API gateway, fine-grained evaluation at the service-level enforcement point, and access to the protected resource. A workload-initiated request follows the same enforcement path with a workload credential in place of a user token. Plane names correspond to those used in the text and in Table III. Flows are distinguished by line style and weight rather than by colour alone, so the figure remains unambiguous in greyscale.

### *A. Runtime Request Flow*

A request authenticates through the human identity plane, which establishes actor identity and organizational context. The authorization plane issues a token, and the API gateway performs a first pass of coarse-grained validation-confirming that the token is well formed, unexpired, and issued by a trusted authority-before the request is routed internally. Within the platform, the policy decision point evaluates the specific action against role, attribute, and organizational rules derived from the entitlement catalog, and the service-level enforcement point applies that decision close to the protected resource rather than relying on gateway validation alone. A request initiated by a workload rather than a person follows the same path, entering with a workload credential in place of a user token, and is evaluated by the same policy decision point.

### *B. Governance Flow*

The governance flow runs on a different cadence from the runtime flow and is easily overlooked because it is not on the request path. An access request enters an approval workflow; an approved entitlement is written to the entitlement catalog; the catalog state is translated into policy held in the policy store, which the policy decision point evaluates at runtime. Scheduled access review examines existing entitlements against current business need, and its outcome either recertifies or revokes them. Revocation must propagate to the enforcement plane within a defined objective for the sensitivity of the resource, rather than taking effect only when a credential next expires. Without that propagation objective, an entitlement removed in the governance plane can remain effective at the edge for as long as an issued token remains valid.

### *C. Audit and Evidence Flow*

Security-relevant authentication events, authorization decisions, entitlement changes, and lifecycle actions emit structured records to the evidence plane. Audit records may be captured asynchronously so that evidence collection does not delay the protected transaction, provided delivery is durable and gaps are detected rather than assumed absent. Consistent correlation identifiers connect access logs, decision logs, application logs, and distributed traces, so that the path from request to outcome can be reconstructed without ad hoc log matching across disconnected systems. The evidence plane also closes a loop that is frequently left open: the decisions it records are the input to the next access review, which makes certification a judgement informed by observed use rather than a periodic confirmation of the existing state.

### *D. Worked Example: One Entitlement, End to End*

The three flows above are easiest to see combined, so this subsection traces a single entitlement through all three-approval, enforcement, review, and revocation-using the financial-services setting developed further in Section VI.B. The steps are numbered against Fig. 1 where they touch the runtime path.

A client-facing administrator, Contractor A, joins a financial institution's support team and needs read access to that institution's claims records on the shared platform. Governance flow, onboarding: Contractor A's manager submits an access request scoped to the one client organization; the request enters the approval workflow, and a designated approver-someone other than the requester, per segregation-of-duties policy held in the governance plane-approves it with a stated business justification and a review date six months out. The approved entitlement is written to the entitlement catalog as a triple: identity, organization scope, permitted action (read). Governance flow, policy translation: the catalog change is compiled into policy held in the policy store, associating Contractor A's identity with a read-scoped, organization-bounded rule; no token has yet been issued, and no runtime request has yet occurred.

Runtime flow (Fig. 1, stages 1–5): Contractor A authenticates through the human identity plane (stage 1); the authorization plane issues a token carrying the organization claim (stage 2); a request to read a claim record reaches the API gateway, which validates the token's signature, expiry, and issuer (stage 3); the service-level enforcement point evaluates the request against the policy compiled from the governance plane-confirming organization scope and the read-only action-and permits it (stage 4); Contractor A receives the record (stage 5). Audit flow: the decision-actor, organization, resource, policy identifier, outcome, timestamp, and a correlation identifier-is written to the evidence plane in the same request cycle described in Section III.C.

Governance flow, review: at the six-month mark, a scheduled access review surfaces Contractor A's entitlement together with the audit trail of its actual use. The reviewer-informed by usage evidence rather than only by the original justification, per the feedback path described in Section III.C-recertifies the entitlement for another cycle. Governance flow, revocation: four months later, Contractor A's engagement with the client organization ends. The manager submits a revocation; the entitlement is removed from the catalog, the policy store is updated, and the change propagates to the policy decision point's cache. The next request presenting Contractor A's still-unexpired token is evaluated against the updated policy and denied at the service-level enforcement point (stage 4)-the token itself was never revoked or reissued, which is the specific sense in which Section III's central claim holds: the entitlement was governed independently of the credential. Section VIII.A reports simulated timing for this propagation step under stated assumptions.

### *E. Governance Operations at Volume*

Bulk operations such as organization migration, mass de-provisioning at the end of a business relationship, and periodic access certification across large populations do not fit a single request-response cycle. These operations are handled asynchronously, through event queues, retry with idempotency guarantees, reconciliation to detect records left inconsistent, and logging that ties each asynchronous step to the originating event. The governance implications of processing at this volume are developed in Section VII.

## IV. Secure API Authorization and Token-Based Access Control

### *A. Protocol Roles and What Each Mechanism Establishes*

Modern enterprise platforms expose most business functions through APIs, so access control that stops at a login screen is incomplete. OAuth 2.0 is an authorization framework: it defines how a client obtains limited access to a protected resource on a resource owner's behalf, and deliberately does not define how the resource server should reach fine-grained decisions once a valid token is presented; it is also not an authentication protocol. The OAuth 2.0 security best current practice consolidates the experience accumulated since the original specifications, updates their threat model, and deprecates modes of operation now considered insecure [5].

OpenID Connect adds an identity layer on top of OAuth 2.0, returning information about the authentication event itself in an ID Token and standardizing the identity claims a relying party can expect [6]. The distinction is practical: OAuth 2.0 establishes that a caller holds delegated authority, whereas OpenID Connect establishes who authenticated, when, and by what means.

The JSON Web Token (JWT) specification defines a compact, URL-safe format for representing claims, which may be signed as a JSON Web Signature or encrypted as a JSON Web Encryption structure [7]. An ID Token is by definition a JWT, whereas an OAuth 2.0 access token is not required to be one and its format is in principle opaque to the client. The format specification does not by itself make an implementation secure: correct handling requires verifying the signature against a current key, rejecting unexpected or absent algorithms, and validating the issuer, audience, expiry, and not-before claims. The JWT best current practice documents these pitfalls and the corresponding implementation guidance [8].

### *B. Layered Authorization and Object-Level Access*

Authentication together with a broad role assignment is not equivalent to authorization for a specific action on a specific object. An actor may be correctly authenticated and correctly assigned an administrator role and still have no legitimate basis for viewing every organization on the platform, editing every record, or reaching every administrative function the underlying services expose. The OWASP API Security Top 10 lists broken object-level authorization first in its 2023 edition, and lists broken function-level authorization separately, because implementations commonly verify authentication and a broad role and then stop, without checking whether the specific object requested falls within the actor's authorized scope or whether the specific operation is permitted [10].

This is why enforcement belongs close to the protected resource and not only at the API gateway. Gateways are well suited to coarse-grained, uniform controls, and poorly suited to decisions depending on tenant identity, organizational scope, record ownership, or the specific operation attempted, because the gateway does not generally hold business-object, tenant, or record-level context. Regulated platforms therefore combine gateway validation with object, action, and field-level authorization evaluated at the service, drawing on entitlement data held in the governance plane; the mapping study of cloud-native access control reports that placement of enforcement is itself a recurring source of difficulty [19]. Table IV states each layer, the question it answers, and where that question is evaluated.

**TABLE IV.** LAYERED AUTHORIZATION MODEL FOR CLOUD-NATIVE APIS

| Layer | Question answered | Where it is evaluated |
|---|---|---|
| Authentication | Is this actor the identity it claims to be? | Human or workload identity plane, verified at the API gateway |
| Role authorization | What general capability does this role permit? | API gateway or service middleware |
| Organization authorization | Is this action scoped to the correct organization or tenant? | Policy decision point |
| Object authorization | Does this actor hold rights over this specific object? | Service-level enforcement point |
| Action authorization | Is this specific operation permitted, such as read, write, or delete? | Service-level enforcement point |
| Field-level authorization | Which data fields are visible or editable for this actor? | Application data layer |
| Evidence capture | What was decided, for which actor, and on what basis? | Evidence plane, receiving records from every layer above |

*No layer substitutes for another. The object and action layers correspond to the two authorization failure classes listed separately in the OWASP API Security Top 10 [10].*

### *C. Token Exchange and Delegation*

Delegation requires explicit design rather than treatment as an edge case. Where a downstream call is made on a user's behalf while the original credential remains valid, an authorization server or security token service can exchange the user token for a narrower token representing delegation or impersonation, following the pattern defined in OAuth 2.0 token exchange, which also defines the actor and subject claims recording on whose behalf a call is made [9]. The exchanged token should restrict audience, scope, and lifetime, and the audit record should preserve the distinction between an action taken directly by a human user and one taken by a workload acting on that user's behalf. A scheduled batch job requires a time-limited credential scoped to the operation it performs within its window rather than a standing credential valid indefinitely. Both cases are governed by the same principles: least privilege, short credential lifetimes, and a delegation trail that can be reconstructed.

## V. Cloud-Native Workload Identity and Federation

Human identity governance addresses only part of the attack surface. Containers, scheduled jobs, serverless functions, message consumers, and deployment pipelines make privileged calls to other services, and where these workloads authenticate using long-lived shared secrets, the platform becomes progressively harder to secure and to audit as the number of workloads grows. A container running for months with a credential that never rotates presents a comparable governance problem to a human account never reviewed after a change of role.

### *A. Workload Identity as a Governed Lifecycle*

Workload identity requires the same lifecycle discipline as human identity, expressed in workload-specific terms: registration, an assigned owning team, placement within a defined trust domain, attestation rules describing how the workload proves its identity, credential issuance and rotation, an approval record for the permissions granted, periodic review of whether those permissions remain appropriate, and a defined revocation and decommissioning path when the workload is retired. An organization applying this discipline should be able to establish, for any workload identity in production, which team owns it, which services it may reach, who approved those permissions, and whether the identity was removed when the workload was decommissioned. The systematization of Kubernetes security practice reports that broad service-account permissions and unrotated credentials remain recurring and avoidable weaknesses [17], which is a lifecycle failure rather than a mechanism failure.

### *B. Verifiable Workload Credentials*

SPIFFE frames workload identity around cryptographically verifiable, short-lived credentials rather than network location. It defines a workload identity as a SPIFFE ID, a uniform resource identifier naming a trust domain and a path within it, and defines the SPIFFE Verifiable Identity Document (SVID) as the document through which a workload proves that identity, issued in either an X.509 or a JWT form [11]. The SPIFFE Workload API defines the interface through which a workload obtains its SVID and the trust bundles needed to verify its peers, without the workload holding a long-lived secret [12]. SPIRE implements these specifications, using node attestation to establish the identity of the machine and workload attestation to establish the identity of the process before issuing an SVID [13]. Work extending SPIRE to confidential workloads demonstrates that these mechanisms integrate with trusted execution environments [18].

In Kubernetes, a service account represents a workload's identity to the Kubernetes API. Projected service-account tokens are audience-bound and time-limited and are the appropriate mechanism where such access is required; manually created, long-lived service-account token secrets should be avoided [14]. A service account is not by itself a complete business-level identity for every internal call a workload makes, and cluster role bindings should grant only the permissions the workload requires. Table V traces the workload identity flow from provisioning through to the resulting audit record, using generic labels so that the pattern generalizes across cloud providers and orchestration platforms.

**TABLE V.** WORKLOAD IDENTITY FLOW FOR SERVICE-TO-SERVICE AUTHORIZATION

| Step | Action | Responsible component |
|---|---|---|
| 1 | Workload is provisioned with a service account and granted the minimum required cluster permissions [14] | Service account, cluster role-based access control |
| 2 | A registration rule maps the eligible workload to a SPIFFE ID [11], [13] | SPIRE server, registration policy |
| 3 | The identity of the node and then of the workload process is attested at runtime [13] | SPIRE agent, node and workload attestation |
| 4 | An X.509 or JWT SVID representing the verified identity is issued [11] | Workload identity plane |
| 5 | The workload obtains its SVID and peer trust bundles through the Workload API, directly or through a service mesh sidecar [12] | SPIFFE Workload API, optional service mesh sidecar |
| 6 | Both peer identities are verified, and the receiving service evaluates whether the requested action is permitted | Mutual TLS enforcement, policy decision point |
| 7 | The identity, policy decision, requested action, and outcome are recorded | Evidence plane, with correlation identifiers |

### *C. Transport Authentication and Service-Level Authorization*

Mutual Transport Layer Security (mTLS) authenticates both ends of a service-to-service connection and encrypts the traffic between them. Its guarantee should be stated precisely: mTLS establishes that the peer holds a credential for a particular identity, not that the identity is permitted to invoke the endpoint being called. Service mesh authorization policies supply that second decision by restricting which workloads may call which endpoints, and policy engines can further evaluate contextual attributes such as namespace, service-account identity, request path, and the sensitivity of the data involved, consistent with the cloud-native Zero Trust guidance in SP 800-207A [2]. The systematic review of inter-service security threats documents this pattern of gaps directly, finding transport protection comparatively well covered while service-to-service authorization remains less developed, which leaves a workload able to reach an endpoint it authenticated to but was not intended to call [20].

## VI. Application to Regulated Settings

The following three applications are illustrative rather than drawn from a studied deployment. Each shows which planes and which authorization layers carry the load in a setting with a distinct regulatory character.

### *A. Healthcare*

Consider a clinician requesting a patient record. The governance plane holds an entitlement tied to care-team assignment rather than to job title alone, with an approval record and a review cadence. The authorization plane issues a token carrying organizational and care-team context. At runtime, role authorization establishes that a clinician may read records in general, but object authorization is what determines whether this specific patient falls within this clinician's assignment, and that check must occur at the service-level enforcement point because the API gateway does not hold care-team assignment. Field-level authorization suppresses categories of information the role does not require. Emergency access is handled as a governance-plane construct rather than an enforcement bypass: a time-limited entitlement, granted through a defined path, that generates a decision record and a mandatory post-hoc review. The evidence plane is what allows a later review to establish whether the assignment justifying the access was current at the time it was used.

### *B. Financial Services*

Consider a platform servicing multiple financial institutions, where an administrator at one client must not gain visibility into another-the setting traced end to end in Section III.D. Organization authorization is the load-bearing layer, and the failure mode is a correctly authenticated administrator with a correct role reaching a second organization's data because tenant scope was assumed rather than evaluated. The governance plane holds separate entitlements per organization for the same person; the token carries organization scope; the policy decision point evaluates the tenant boundary on every request. Segregation of duties is expressed as a governance-plane constraint on which entitlement combinations may be approved for one individual, and the evidence plane supplies the records an examiner requests to confirm it held. Mass de-provisioning at the end of a client relationship is a governance-plane operation executed through the asynchronous path of Section III.E, with idempotency and reconciliation, because a partially completed de-provisioning is a compliance finding rather than an operational inconvenience.

### *C. Insurance*

Consider claims routing and adjuster access. Action authorization carries the load here: reading a claim, adjusting a reserve, and approving a settlement are distinct operations, and approval authority above a monetary threshold is properly an attribute evaluated at decision time rather than a role held indefinitely. An automated claims-routing service acts under its own workload identity, scoped to the topics and operations it requires. Where that service performs an immediate downstream call on an adjuster's behalf, token exchange yields a narrowly scoped delegated credential and the audit record distinguishes the adjuster from the executing workload [9]. Field-level authorization applies to medical evidence held within a claim file, which is accessible to a claims handler for some purposes and not for others.

## VII. Operational Scalability and Auditability

Identity platforms in regulated industries process lifecycle activity at volumes that a manual or purely synchronous design cannot support: bulk provisioning during an onboarding wave or merger, mass de-provisioning, organization migration, and scheduled entitlement review across a large population. Parallel and asynchronous processing improves throughput, but identity operations carry a requirement that generic batch processing does not always share: every operation must remain traceable, recoverable after partial failure, and idempotent, so that a retried operation produces the same end state rather than a duplicated or conflicting one. A job that partially completes and retries without idempotency guarantees can deactivate an account twice or apply an entitlement change inconsistently across services, and reconstructing what occurred afterward becomes difficult. Asynchronous audit delivery carries the same requirement: the platform needs monitoring and reconciliation to detect that an expected audit record did not arrive, rather than treating silence as success.

Each access decision should record the identity of the actor or workload, the organizational context, the resource affected, a policy identifier and version, a decision identifier, the outcome, a timestamp, and a correlation identifier tying the decision to the wider request or workflow, for the reasons set out in Section III.C. This evidence supports incident investigation, internal risk reporting, formal audit, and root-cause analysis; an architecture that produces it as a by-product of normal operation is more defensible than one requiring reconstruction after an incident. Where regulatory requirements call for it, audit storage should be append-only or otherwise tamper-resistant, access to audit data should itself be controlled and recorded, and retention and deletion should follow a defined schedule.

A tension arises between detail and exposure, and the architecture should resolve it explicitly. Audit records must be detailed enough to support reconstruction while avoiding the capture of more sensitive data than necessary. Recording full request payloads may appear safest for later investigation, but it creates its own exposure by placing personal data, secrets, or protected business information in a store that was not designed with the access controls of the primary system of record. The alternative is to record correlation, event, policy, and resource identifiers in place of payload content, which preserves reconstruction while keeping access tokens, secrets, and complete sensitive payloads out of the audit trail. An architecture built along these lines produces the evidence that regulators and auditors expect; it does not by itself establish regulatory compliance, which depends on organizational processes, personnel, and controls beyond the technical design.

## VIII. Simulation, Evaluation Approach, and Limitations

### *A. Discrete-Event Simulation of Policy Evaluation and Revocation Propagation*

The architecture has not been implemented as a production system, and the figures in this subsection are simulated, not measured. A discrete-event simulation was built for this article to give the latency and revocation claims a quantitative basis rather than leaving them as unqualified assertions. The simulation and its parameters are reported here in full so the estimate can be reproduced or challenged; they should not be read as production benchmark results.

**Policy-evaluation overhead.** The simulation models 2,000 requests under two conditions: a baseline representing authentication and coarse role checking only, and a fine-grained-authorization condition representing the layered model of Table IV, with an 85% policy-decision-point cache-hit rate (a cache hit is evaluated locally against a cached policy; a cache miss incurs a simulated round trip to a remote policy decision point). Both conditions draw base request latency from the same distribution, so the reported difference isolates the modeled authorization overhead rather than unrelated request-time variance. Under these assumptions, mean added latency was 1.5 ms and 95th-percentile added latency was 7.9 ms, driven by the minority of requests modeled as cache misses. This indicates that the overhead of fine-grained authorization is small when most decisions are served from a local cache and dominated by remote policy-decision-point calls when they are not-which is consistent with, though not a substitute for, the qualitative placement argument in Section IV.B, and is the reason Evaluation item 6 below calls for measuring cache-hit-dependent overhead specifically in a real deployment.

**Revocation propagation.** The simulation models revocation events for two resource sensitivity classes against the propagation objectives introduced in Section III.B: a high-sensitivity class (500 ms objective, illustrated by the financial-services tenant boundary of Section III.D and VI.B) and a standard class (5,000 ms objective, illustrated by general application access). Under the modeled propagation-delay distributions (mean 249 ms, 95th percentile 352 ms for the high-sensitivity class; mean 1,803 ms, 95th percentile 2,509 ms for the standard class, each over 500 simulated events), both classes met their stated objectives at the 95th and 99th percentile in the simulation. This is a statement about the simulated model meeting the objectives it was parameterized to meet, not evidence that a real message queue and policy-store update path would behave identically; Evaluation item 3 below specifies how that would need to be measured against an actual implementation.

The simulation code, parameters, and random seed are available from the corresponding author on request, consistent with the article's reproducibility position: no figure in this subsection is asserted without the model that produced it.

### *B. How the Full Architecture Could Be Evaluated*

The simulation in Section VIII.A addresses two of the seven measurements below; the remaining five require an implementation this article does not provide. It is nonetheless useful to state all seven, both because the client of such an architecture is entitled to know what evidence would support it and because the proposed tests indicate what the design is claiming.

**1)Structural conformance.** Can each of the six planes be located in the implementation, and is the governance plane genuinely separable from the authorization plane, in the sense demonstrated conceptually in Section III.D-an entitlement can be revoked without reissuing credentials?

**2)Authorization-layer tests.** With a valid token and a correct broad role, attempt access to an object outside the actor's scope, to a second organization's data, to an operation outside the permitted set, and to a restricted field. Each attempt should be denied and should produce a decision record. These tests correspond directly to the layers of Table IV.

**3)Revocation propagation (implementation measurement).** Measure elapsed time from revocation in the governance plane to first denial at the enforcement plane, per resource class, in a deployed system, and compare against the objective defined for that class and against the simulated figures in Section VIII.A.

**4)Evidence completeness**. Generate a known number of decisions, then reconcile against the records held, confirming that audit records are not sampled and that any gap is detected rather than inferred.

**5)Certification effectiveness.** Measure the proportion of entitlements reviewed within their defined cadence and the proportion revoked as a result, which distinguishes a review process that changes outcomes from one that confirms the existing state.

**6)Overhead (implementation measurement).** Measure added request latency and policy-evaluation load against a baseline without fine-grained authorization, in a deployed system, separating decisions evaluated locally against cached entitlements from those requiring a call to the policy decision point, and compare against the simulated figures in Section VIII.A.

**7)Workload identity hygiene.** For a sample of workload identities in production, establish the owning team, the services each may reach, the approval record, credential age, and whether identities belonging to decommissioned workloads were removed.

These are stated as an evaluation design against which the simulation in Section VIII.A is a first, partial step. No implementation results are reported, and the article makes no claim about how a given deployment would perform against items 3, 4, 5, 6, or 7.

*C. Limitations*

Five limitations follow from the nature of the work. The architecture is conceptual and has not been implemented as a complete production system; the simulation in Section VIII.A models the behavior the architecture specifies under stated assumptions and is not a substitute for measuring an actual deployment. The comparisons in Tables I to IV are qualitative characterizations drawn from the cited standards and literature and from the structure of the mechanisms themselves; they are not measurements and should not be read as predictions for a specific deployment. The simulated overhead and propagation figures depend on the cache-hit rate, network-latency, and queueing assumptions stated in Section VIII.A, and a real deployment's cache-hit rate, network topology, and policy-store implementation could shift them substantially in either direction. The model supports the production of compliance evidence but does not establish regulatory compliance, which depends on process, personnel, and controls outside the technical architecture. And the scope is confined to the access path: identity proofing procedures, cryptographic key management, and third-party risk assessment are treated as adjacent concerns rather than developed here. The three applications in Section VI, and the worked entitlement lifecycle in Section III.D, are illustrative constructions based on common patterns rather than observations of a studied deployment, and should be read as demonstrations of how the model applies rather than as evidence that it works in a specific installation.

## IX. Conclusion

The architecture described here treats identity governance as infrastructure on which a regulated platform depends, rather than as an administrative function operating behind it. Its central structural claim is that entitlement state belongs in a governance plane separate from the authorization plane that issues credentials and evaluates requests, because that separation is what makes an entitlement reviewable, recertifiable, and revocable independently of the credentials carrying it-a claim traced concretely in Section III.D and given a preliminary quantitative treatment in Section VIII.A. Around that separation, the model places human and workload identity in one control plane, distinguishes six layers of authorization from authentication through field-level access, treats workload identity as a governed lifecycle, and makes the evidence plane a first-class element whose records feed the next access review.

The model is intended to enable a regulated enterprise to explain, after the fact, why access existed, under which policy it was exercised, and when it was reviewed or removed. Whether it does so in a given deployment is an empirical question about that deployment rather than a property established by the architecture or its simulation, and Section VIII.B sets out what evidence would answer it. Future work could extend the model toward continuous access certification driven by usage telemetry rather than fixed review cycles, toward interoperability between workload identity federation schemes and the policy engines used for human-facing authorization, and toward the field measurements-items 3 through 7 of Section VIII.B-that a production implementation would make possible.

## References


[1] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Special Publication 800-207, Aug. 2020, doi: 10.6028/NIST.SP.800-207.

[2] R. Chandramouli and Z. Butcher, "A zero trust architecture model for access control in cloud-native applications in multi-location environments," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Special Publication 800-207A, Sep. 2023, doi: 10.6028/NIST.SP.800-207A.

[3] D. Temoshok, Y.-Y. Choong, R. Galluzzo, C. LaSalle, A. Regenscheid, D. Proud-Madruga, S. Gupta, and N. Lefkovitz, "Digital identity guidelines," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Special Publication 800-63-4, Jul. 2025, doi: 10.6028/NIST.SP.800-63-4.

[4] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Special Publication 800-162, Jan. 2014, includes updates as of Aug. 2, 2019, doi: 10.6028/NIST.SP.800-162.

[5] T. Lodderstedt, J. Bradley, A. Labunets, and D. Fett, "Best current practice for OAuth 2.0 security," Internet Engineering Task Force, RFC 9700 (BCP 240), Jan. 2025, doi: 10.17487/RFC9700.

[6] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 2," OpenID Foundation, final specification, Dec. 2023. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html (accessed Aug. 4, 2026).

[7] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," Internet Engineering Task Force, RFC 7519, May 2015, doi: 10.17487/RFC7519.

[8] Y. Sheffer, D. Hardt, and M. Jones, "JSON Web Token best current practices," Internet Engineering Task Force, RFC 8725 (BCP 225), Feb. 2020, doi: 10.17487/RFC8725.

[9] M. Jones, A. Nadalin, B. Campbell, Ed., J. Bradley, and C. Mortimore, "OAuth 2.0 token exchange," Internet Engineering Task Force, RFC 8693, Jan. 2020, doi: 10.17487/RFC8693.

[10] OWASP Foundation, "OWASP API Security Top 10 – 2023," 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0x00-header/ (accessed Aug. 4, 2026).

[11] SPIFFE Project, "SPIFFE identity and verifiable identity document," The SPIFFE Standard, v1.15.2. [Online]. Available: https://spiffe.io/docs/latest/spiffe-specs/spiffe-id/ (accessed Aug. 4, 2026).

[12] SPIFFE Project, “SPIFFE Workload API,” The SPIFFE Standard, v1.15.2. [Online]. Available: https://spiffe.io/docs/latest/spiffe-specs/spiffe_workload_api/ (accessed Aug. 4, 2026).

[13] SPIFFE Project, “SPIRE concepts,” SPIRE Documentation, v1.15.2. [Online]. Available: https://spiffe.io/docs/latest/spire-about/spire-concepts/ (accessed Aug. 4, 2026).

[14] Kubernetes Authors, “Managing service accounts,” Kubernetes Documentation. [Online]. Available: https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/ (accessed Aug. 4, 2026).

[15] N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, “Zero trust architecture (ZTA): A comprehensive survey,” IEEE Access, vol. 10, pp. 57143–57179, 2022, doi: 10.1109/ACCESS.2022.3174679.

[16] Y. He, D. Huang, L. Chen, Y. Ni, and X. Ma, “A survey on zero trust architecture: Challenges and future trends,” Wireless Commun. Mobile Comput., vol. 2022, art. no. 6476274, Jun. 2022, doi: 10.1155/2022/6476274.

[17] M. S. I. Shamim, F. A. Bhuiyan, and A. Rahman, “XI commandments of Kubernetes security: A systematization of knowledge related to Kubernetes security practices,” in Proc. IEEE Secure Develop. Conf. (SecDev), Atlanta, GA, USA, Sep. 2020, pp. 58–64, doi: 10.1109/SecDev45635.2020.00025.

[18] E. Falcão, M. Silva, A. Luz, and A. Brito, “Supporting confidential workloads in SPIRE,” in Proc. IEEE 14th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom), Bangkok, Thailand, Dec. 2022, pp. 186–193, doi: 10.1109/CloudCom55334.2022.00035.

[19] M. S. Rahaman, S. N. Tisha, E. Song, and T. Cerny, “Access control design practice and solutions in cloud-native architecture: A systematic mapping study,” Sensors, vol. 23, no. 7, art. no. 3413, Mar. 2023, doi: 10.3390/s23073413.

[20] P. Haindl, P. Kochberger, and M. Sveggen, “A systematic literature review of inter-service security threats and mitigation strategies in microservice architectures,” IEEE Access, vol. 12, pp. 90252–90286, 2024, doi: 10.1109/ACCESS.2024.3406500.