International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

Applying Zero Trust to Financial Microservices through Default Deny Authorization

Surendra Kumar Valavala

Software Engineer at SS&C Company, United States

Abstract: Financial platforms often authenticate requests at an external gateway while distributing the information needed for authorization across account, holdings, reporting, and transaction services. This separation creates a persistent risk: a valid identity can reach an operation without having authority over its resource or financial effect. This article develops a default-deny authorization architecture that evaluates workload identity, represented user authority, institutional ownership, and relevant business state at the service responsible for execution. It distinguishes delegated user requests from separately approved service computations and binds sensitive approvals to the resource state actually reviewed. The design combines centralized policy governance with distributed enforcement, including tenant restrictions in queries, caches, exports, and asynchronous processing.

A payment-approval example and four financial scenarios translate the architecture into testable requirements. The analysis addresses unavailable dependencies, stale permissions, credential misuse, evidence durability, and incremental migration. A proposed evaluation compares unauthorized effects and legitimate completion alongside enforcement coverage, revocation delay, and decision overhead. The contribution is a financial-domain synthesis of established security mechanisms and an evaluation method for examining their combined behavior. The study reports no implementation results, measured performance improvement, or compliance certification.

KEYWORDS: Zero Trust architecture; financial microservices; default-deny authorization; workload identity; tenant isolation; transaction approval.

I. INTRODUCTION

A payment request may arrive with a valid access token and an authenticated service connection yet still require rejection. The requester may lack authority over the account, exceed an approval limit, or attempt to approve a payment they created. A beneficiary change after review introduces a further problem: an earlier authorization decision may no longer describe the instruction about to execute. These conditions cannot be resolved by identifying the caller alone.

Microservices distribute these decisions across components with different responsibilities. A gateway evaluates incoming credentials and routes. An account service maintains ownership relationships. A transaction service controls approval stages. A data-access path determines which records are disclosed or changed. NIST identifies authentication, access control, communication security, and monitoring as central concerns in microservice systems [1]. FFIEC guidance similarly extends access and authentication considerations beyond customers to employees, third parties, and system-to-system communication [2].

This article examines how authority can retain its meaning across those boundaries. It proposes an authorization contract that identifies the permitted financial action, the resource, the execution purpose, and the conditions that must hold when the operation takes effect. Enforcement remains with the service accountable for that effect. The design applies the resource-oriented principles of Zero Trust Architecture [3] without treating network segmentation or gateway authentication as substitutes for financial permissions.

The scope is a conceptual architecture for institutions operating account, holdings, valuation, reporting, and transaction services. The financial examples are hypothetical. The analysis preserves the distinction between a proposed security property and evidence that an implementation achieves it.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

II. TECHNICAL FOUNDATIONS AND CONTRIBUTION

Default-deny authorization follows the established principles of fail-safe defaults and complete mediation described by Saltzer and Schroeder [4]. Service-mesh security and attribute-based decisions are developed in NIST SP 800-204A and SP 800-204B [5], [6]. The April 2023 initial public draft of SP 800-207A extends this discussion to application identities [7]. Zero Trust surveys and migration research also address the organizational and operational work required to adopt these controls [8], [9]. These publications provide the technical foundation; they do not validate the particular financial design presented here.

The contribution is the combination of workload permission, user delegation, authoritative ownership, transaction state, and enforceable obligations within a reviewable decision model. The model is paired with an evaluation plan that tests actual disclosures and financial changes, including failures between a policy decision and execution. It introduces no new cryptographic mechanism and makes no claim of superiority over every existing financial reference architecture.

The threat model includes authenticated users requesting prohibited resources, a compromised workload using its valid credential, caller-controlled identifiers, stolen tokens, and authorization dependencies that become unavailable or stale. It assumes that designated ownership records and credential issuers remain authoritative and that each protected operation reaches an enforcement point the attacker cannot bypass. Compromise of those foundations requires separate controls and changes the assurance the contract can provide. Within these assumptions, the required property is that a protected financial effect occurs only under an applicable decision whose conditions still hold. The evaluation must challenge those conditions individually, rather than infer security from successful authentication or the presence of a policy engine.

Figure 1 contrasts two authorization assumptions. In the simplified baseline, an internal credential or network position carries excessive authority. In the proposed design, the resource service requires an explicit permit. Mature perimeter-oriented systems may already implement comparable controls. The comparison concerns the basis of a decision, rather than the age or marketing category of the underlying infrastructure.

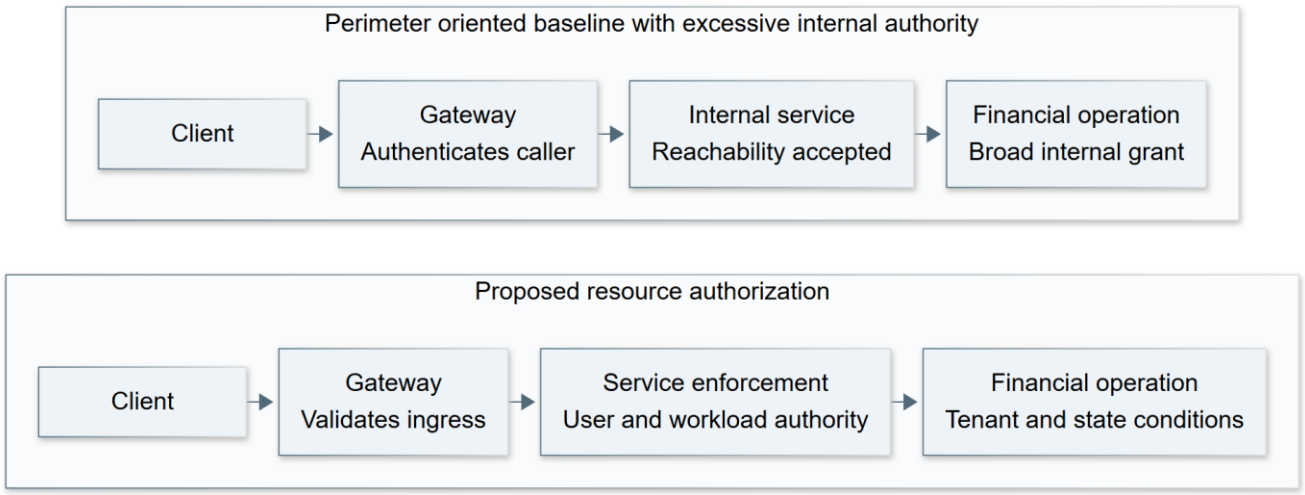


Figure.1. Authorization assumptions at the internal service boundary

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

Table.1. Authorization assumptions and operating responsibilities

Dimension	Simplified perimeter baseline	Proposed resource authorization
Source of authority	Admission or internal reachability may imply broad access.	An explicit permit covers a named resource and action.
Internal identity	A shared credential may identify several services.	A named workload acts through verified delegation or an approved purpose.
Financial restrictions	A route role may omit ownership or workflow conditions.	Tenant, field, amount, approval, and resource-state checks apply.
Enforcement	Gateway admission may be treated as sufficient.	The resource service and data path enforce required conditions.
Decision evidence	Entry logs may not identify the deciding boundary.	A decision reference connects permission with the financial outcome.
Operating obligations	Implicit grants can conceal gaps and dependency assumptions.	Policy lifecycle, freshness, recovery, and test coverage require ownership.

III. IDENTITY AND DELEGATED AUTHORITY

OAuth 2.0 provides a framework for limited API access [10]. OpenID Connect adds end-user authentication to that framework [11]. An ID token issued to a client must not be treated as an access token for an unrelated resource server. Single Sign-On establishes a session; the financial API must still determine whether the requested operation falls within the caller's authority.

For JWT access tokens, the receiving API validates the signature against trusted issuer configuration, restricts acceptable algorithms, checks issuer and audience, and enforces applicable time constraints. A deployment adopting RFC 9068 must also enforce that access-token profile [12]. Separate validation rules and trusted key selection reduce cross-JWT substitution and algorithm-confusion risks described in RFC 8725 [13]. A caller-controlled claim or arbitrary key location must not redefine the verifier's trust configuration.

Opaque tokens may instead be checked through authenticated introspection, which supplies token status and associated metadata [14]. The choice between introspection and local validation affects dependencies and freshness. Neither approach independently establishes current ownership of a portfolio. Browser-facing authorization should address authorization-code interception, token disclosure, and excessive token privileges using the OAuth threat model [15].

Workload authentication establishes which service is calling. Mutual TLS requires both certificate validation and verification of the expected peer identity. Certificate-bound access tokens can constrain token use to the authenticated client certificate [16]. For public OAuth clients, PKCE addresses authorization-code interception [17]; it does not bind an issued access token to its presenter. A compromised legitimate endpoint may still exercise its own permissions. Repeated financial instructions also require business idempotency controls.

SPIFFE provides a model for verifiable workload identities [18]. Its usefulness depends on trustworthy issuance, attestation, and credential isolation. Credentials require protected storage, automated rotation, and exclusion from application logs. A credential shared by many services weakens attribution and complicates selective revocation. Legacy shared accounts should therefore retain only the narrow permissions required during migration, with explicit ownership and a retirement plan.

Delegated operations require evidence of both the represented user and the acting workload. An approved token-exchange process can issue an audience-specific downstream credential and express actor and subject information [19]. The receiving service verifies the credential, the current transport peer, and the relationship permitted between them. Unsigned user headers and client-selected tenant values cannot establish that relationship. Where a proxy terminates transport authentication, it must convey verified context through a protected mechanism and remove spoofable incoming identity headers.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

IV. A FINANCIAL AUTHORIZATION CONTRACT

The contract begins with a business action. Reading a payment, creating it, approving it, and releasing it involve different authorities even when they share an API or database. Broad roles can establish responsibilities, while resource attributes and workflow conditions narrow a particular grant. Role-based access control and attribute-based access control are therefore complementary. Independent approval and least privilege also correspond to access-control objectives in NIST SP 800-53 [20].

A decision input identifies the authenticated workload, any represented user, delegation evidence, the registered action, a canonical resource identifier, authoritative tenant ownership, relevant resource state, and decision context. The evaluator obtains ownership and current workflow facts from institution-designated sources. Client-supplied identifiers can help locate a resource but cannot prove entitlement to it. Attribute resolution should retrieve only the metadata needed for the decision and must not disclose protected financial information before authorization succeeds.

Conceptually, permission requires verified identities, an approved execution mode, an applicable resource grant, satisfied tenant and workflow restrictions, sufficiently fresh policy inputs, and obligations that the service can enforce. If any required condition is unresolved, no permit is issued. A mandatory prohibition, such as self-approval, takes precedence over a broad approver role. Conflict handling must be specified explicitly rather than delegated to whichever rule happens to run first.

4.1 Delegated requests and approved computations

For delegated approval, the workload must be permitted to invoke the approval function and the user must hold authority over the account and amount. One grant cannot compensate for the absence of the other. Where required, the service also verifies independent approval, authentication strength, and session freshness using trusted evidence.

A scheduled valuation or regulatory report may legitimately read raw positions that an individual recipient cannot inspect. Applying every recipient's raw-data permissions to the computation would block valid aggregation. The alternative is a separately registered service purpose with defined resource scope, output restrictions, and an accountable owner. The institution approves this execution mode through policy and configuration. A failed user request must not be retried under a generic service-only identity to evade the original restriction.

4.2 Binding decisions to business state

Suppose an approver reviews payment P at version 7. The decision covers its institution, account, amount, beneficiary, and approval stage. Another request changes the beneficiary and advances the payment to version 8 before approval commits. The permit for version 7 must not authorize the changed instruction. The transaction service should require a conditional state transition, or an equivalent transaction mechanism, that checks the reviewed input state before committing approval.

The version is an input precondition. A legitimate approval may itself advance version 7 to version 8; the prohibited case is an intervening change that invalidates the reviewed facts. On a mismatch, the service rejects that approval attempt. A material change requires renewed review and a fresh decision. Automatically reevaluating policy without renewed review could authorize details the approver never examined.

Version checks address a resource-state race, while external entitlements present a different consistency problem. The institution must define which facts are rechecked authoritatively at execution, which may be cached, and the maximum permitted age of each. Immediate revocation across independent systems requires coordination capable of providing that guarantee. Short-lived tokens and invalidation events alone do not establish simultaneous global state.

A permit therefore includes a decision identifier, policy version, validity boundary, resource-state binding, and applicable obligations. Obligations can require a response projection or durable evidence associated with the transition. If the service cannot fulfill a mandatory condition, the permit is unusable. A successful evaluator response is only one part of an authorized operation.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

V. ENFORCEMENT ACROSS SERVICE AND DATA BOUNDARIES

The representative platform includes an API Gateway, Identity Provider, Account Service, Holdings Service, Valuation Service, Transaction Service, Reporting Service, Reference Data Service, Administrative Service, and Audit and Monitoring Service. These names identify responsibilities rather than requiring a separate deployment for every function. Administrative and reference-data paths require explicit enforcement whenever their operations are protected. The gateway checks external credentials, request structure, quotas, and coarse access rules. Each resource service applies the policy for its financial action. A policy enforcement point intercepts the operation and applies the decision; a policy decision point evaluates rules and authoritative attributes. These are logical roles. NIST further distinguishes the policy engine and policy administrator within its control model [3]. This article uses the decision-point term generically and does not collapse those responsibilities into the gateway.

Central governance maintains approved rules, policy versions, and release ownership. Runtime evaluation may use a local engine with an integrity-protected bundle or a remote authorization service. A shared policy model does not require one network call for every decision. Figure 2 shows this division between policy governance and enforcement at the financial service and data path.

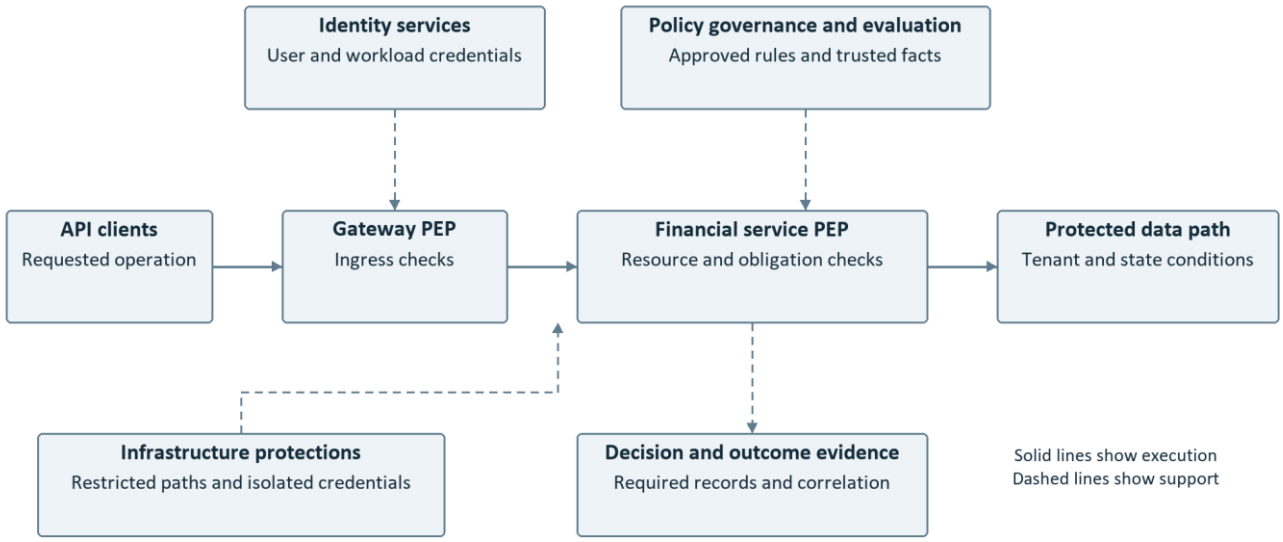


Figure.2. Resource enforcement and supporting control services

A sidecar can authenticate peers and evaluate rules for which it has sufficient inputs. Account ownership, restricted fields, and transaction conditions also require application knowledge and enforcement in the business or data-access path. A proxy that lacks those facts cannot resolve the complete authorization question.

Tenant constraints must survive queries, caches, exports, and message consumption. The Holdings Service resolves the requested portfolio to a canonical resource and applies trusted ownership and entitlement predicates before retrieving protected records. Filtering an unrestricted result only at the response boundary can expose data through intermediate processing or logs. Cache keys must distinguish institutions even when local account identifiers collide. Aggregation and repeated queries also require output controls where they could reveal restricted positions.

Network restrictions and credential protection support this boundary. Kubernetes NetworkPolicy controls supported communication paths when an enforcing network implementation is present [21]; it does not authorize a financial object. CNCF lifecycle guidance and NSA and CISA Kubernetes hardening guidance address complementary concerns such as deployment integrity, restricted credential access, and protected storage [22], [23]. Those controls constrain opportunities for misuse while the financial service determines the operation's authority.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

VI. PAYMENT APPROVAL FROM REVIEW TO EXECUTION

Consider an approver submitting payment P for approval in institution A. The gateway validates the incoming credential and obtains an approved delegated credential for the Transaction Service. That service authenticates the calling workload, verifies the represented user and delegation, and retrieves the payment's current ownership and workflow facts. A decision then evaluates approval authority, applicable amount limits, and creator–approver separation.

An illustrative permit D41 identifies P at input version 7, its protected payment details, the approved policy release, and an expiry. Before applying the transition, the service verifies that the permit still belongs to the current caller, action, resource, and execution mode. It checks the required freshness conditions and confirms that the protected state has not changed. Figure 3 summarizes the sequence; it describes a proposed design, rather than a production trace.

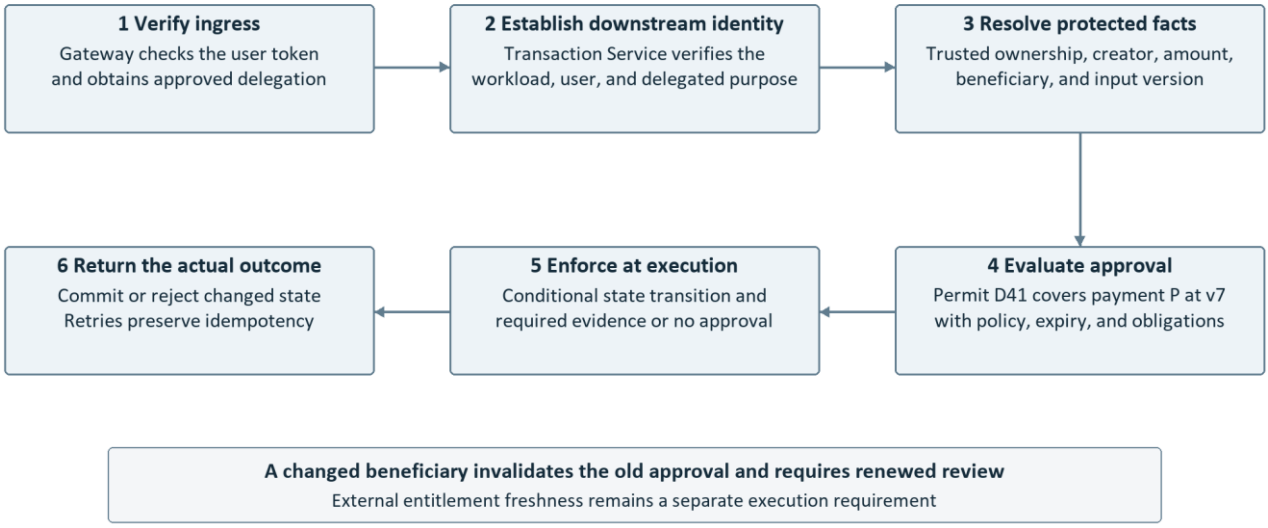


Figure.3. Payment approval with delegated authority and a state precondition

The business-state update must enforce the precondition. Merely recording that version 7 was approved does not prevent version 8 from being executed. Where approval emits a downstream release instruction, the implementation must also prevent partial success in which the caller receives an error after an instruction has already escaped. A transactionally coordinated journal or equivalent mechanism is one implementation option. The choice requires validation against the platform's storage and messaging guarantees.

Evidence obligations apply at the same boundary. If durable evidence is mandatory before a sensitive transition, a remote logging outage must not silently waive it. The service needs a durable local record, another approved mechanism, or a defined failure outcome. Less sensitive paths may permit bounded buffering under explicit policy, provided that loss and recovery are detectable. Retries must preserve idempotency so that recovery from an authorization failure does not create a second financial effect.

VII. FINANCIAL SCENARIOS AND TESTABLE PROPERTIES

7.1 Unauthorized portfolio access

An authenticated customer substitutes another customer's portfolio identifier. The Holdings Service evaluates the requested resource rather than relying on gateway admission. It verifies user entitlement, the workload's permitted function, and institutional ownership before applying a tenant-scoped query and approved field projection. The test succeeds only if protected holdings remain undisclosed. Enumeration, pagination, exports, and error messages must also be examined for information that reveals another customer's resources.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

7.2 Misuse of reporting credentials

A compromised reporting workload attempts to modify a transaction using a valid service credential. The Transaction Service authenticates that workload but rejects the modification because its registered reporting purpose provides no transaction-management authority. The same restriction applies to direct service calls that bypass the gateway. This contains the unauthorized change; it does not eliminate misuse of data within the reporting workload's legitimate scope. Narrow reporting grants and output restrictions therefore remain important.

7.3 Cross institution valuation requests

A caller supplies another institution's identifier or an account number that exists in two institutional namespaces. The Valuation Service treats the supplied values as lookup inputs and derives ownership from trusted records. Downstream holdings reads, cached valuations, exports, and background jobs preserve that ownership context. Shared infrastructure establishes no cross-institution grant. Deliberate shared analytics requires an explicit purpose and restricted output policy, with tests for both approved and prohibited combinations.

7.4 Unauthorized or outdated approval

A user attempts self-approval, exceeds an approval limit, or submits an approval after a beneficiary change. The Transaction and Administrative services enforce the required authority and independence. The resource-state precondition prevents approval against outdated details. Tests inspect stored state, queued instructions, and decision records, rather than treating an HTTP error as proof that no financial effect occurred. A valid retry must require the necessary review and must not duplicate an earlier committed operation.

These scenarios cover object, field, function, tenant, and workflow constraints. The OWASP 2019 edition addresses object-level and function-level authorization, excessive data exposure, and mass assignment [24]. Table 2 relates these risks to broader threats and failure conditions. Its controls are proposed requirements; effectiveness must be established by observing an implementation.

Table.2. Threats controls and decisive observations

Threat or failure	Required control	Residual risk or verification
Object and field disclosure	Authoritative entitlement, tenant-scoped queries, and approved projections.	Test identifiers, pagination, exports, and reconstruction through repeated queries.
Function abuse and excess privilege	Explicit action grants, approved execution mode, and independent approval.	Attempt self-approval and reporting-to-transaction escalation.
Credential theft and lateral movement	Verified workload identity, protected keys, constrained tokens, and restricted paths.	A compromised endpoint can still exercise its legitimate authority.
Cross-institution leakage	Trusted ownership in queries, cache keys, exports, and consumers.	Use colliding identifiers and shared analytics paths.
API and workflow abuse	Per-subject quotas and operation-aware controls on sensitive flows.	Valid permissions may still enable harmful automated behavior.
Unsafe policy administration	Independent review, integrity-protected releases, and policy invariants.	A signed policy can be too broad; introduce and detect a faulty grant.
Unavailable or stale decisions	Fresh approved local evaluation or prevention of the protected effect.	Interrupt dependencies and revoke access between decision and commit.
State races and lost evidence	Conditional transitions, coordinated instructions, and durable required records.	Inspect committed state, emitted instructions, and evidence during faults.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

VIII. AVAILABILITY FRESHNESS AND OPERATIONAL COST

An evaluator should distinguish a permit, an explicit denial, and an indeterminate result caused by missing or unverifiable inputs. Both denial and indeterminacy prevent the protected effect, but their operational treatment differs. A denied entitlement calls for permissions review; an unavailable ownership source calls for dependency recovery. Combining these outcomes obscures both reliability and security behavior.

Local evaluation can sustain operations during a network interruption if approved policy, credentials, and required attributes remain valid within their assigned freshness limits. This is normal enforcement under a defined policy. Reusing an expired decision or omitting an unavailable ownership check would introduce permission that the evaluator has not established. Replication and bounded deadlines can reduce interruption, but retries, circuit breakers, and load shedding must not convert authorization uncertainty into business success.

Caching requires a security budget tied to the financial action. A decision key must represent every input that can change the outcome, including workload, represented user, execution mode, tenant, action, resource state, and policy version. A cached grant must expire when any required fact becomes too old. Invalidation can shorten exposure, although delayed or lost events remain part of the failure model. A reference-price read and a payment release may reasonably require different freshness guarantees.

Performance analysis should separate token validation, attribute retrieval, policy evaluation, state checks, and evidence persistence. Fan-out can multiply these costs even when an individual rule is inexpensive. Reports should include key rotation, policy changes, cache misses, and degraded dependencies. Optimizing a policy engine does not establish acceptable end-to-end financial performance.

Emergency access requires a separately governed policy with defined actions, independent approval where appropriate, limited validity, and accountable evidence. A hidden fallback that activates an unrestricted service account during a timeout defeats the architecture's authority model. Recovery procedures must state what operators can restore and which restrictions continue to apply.

IX. ADOPTION GOVERNANCE AND CONTROL EVIDENCE

Migration should begin with a bounded workflow and a resource inventory. The inventory includes API versions, administrative tools, batch jobs, message consumers, database procedures, and direct service paths. An undocumented path remains a gap even when every known route has an authorization guard. The action catalog records each resource owner, accepted issuer, permitted workload, execution mode, required attributes, tenant rule, approval conditions, and failure response.

Historical access logs can reveal dependencies, but they do not prove that observed permissions are legitimate. Business and security owners must review the intended authority and specify prohibited cases. A controlled adapter may translate legacy authentication into constrained internal assertions. It should restrict source, audience, and lifetime, remove caller-supplied identity context, and receive direct testing because it concentrates trust during migration.
Shadow evaluation compares proposed decisions with independently reviewed expectations while existing controls remain active. A canary deployment limits the impact of policy errors before wider enforcement. Temporary exceptions need an owner, narrow scope, expiry, and compensating control. Rollback restores an approved restrictive policy rather than removing authorization. This sequence is consistent with treating Zero Trust adoption as an organizational and technical process [9].

The policy control plane requires independent review of expanded grants, prohibited role combinations, and changes to approved service purposes. Integrity-protected releases, policy-difference review, invariant tests, and monitored deployment reduce the risk of an authorized administrator publishing an unsafe rule. A correctly signed policy can still grant excessive authority.

Evidence should connect decisions with actual operations, including requests rejected before business logic runs. Useful fields include workload and user references, action, resource reference, outcome, reason, policy version, and correlation identifier. Raw tokens, credentials, and unnecessary financial details should be excluded. Access restrictions and retention controls must protect the resulting records.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

FFIEC guidance supports risk-based authentication, layered controls, and monitoring without endorsing this architecture [2]. NIST access-control and audit families provide a way to organize supporting evidence [20]. For cardholder-data environments within PCI DSS scope, access restriction, authentication, and logging requirements are relevant [25]. Applicability and compliance require separate assessment; adopting this framework does not certify an institution or satisfy an entire standard.

X. EVALUATION DESIGN

A prototype should implement a bounded workflow with account ownership, holdings retrieval, reporting, transaction initiation and approval, and tenant-scoped storage. Synthetic data should include overlapping roles, restricted fields, duplicate local account identifiers, service computations, and changing entitlements. An independently specified entitlement matrix provides the expected-decision oracle. Generating expectations from the policy under test would allow a mistaken rule to validate itself.

The comparison should keep business logic, credential strength, infrastructure, and workload constant while changing the authorization enforcement model. A gateway-authenticated baseline must be defined precisely. If the proposed configuration also introduces a mesh, a different database, and stronger credentials, the observed difference cannot be attributed solely to resource-level authorization. An intermediate configuration can help isolate transport identity from financial policy enforcement.

Table 3 defines the measurements. Rejection rates alone are insufficient: a system that denies every request can appear secure while failing its business purpose. Every adversarial batch therefore needs authorized positive controls, examination of returned data, and inspection of committed state and emitted instructions.

Table.3. Proposed measurements and validation methods

Measure	Definition	Validation method
Unauthorized effects	Disclosures or financial changes outside the independent entitlement matrix.	Inspect returned data, stored state, and emitted instructions.
Unauthorized rejection	Explicitly denied unauthorized requests divided by unauthorized requests sent.	Separate denials, credential failures, and dependency errors.
Tenant isolation	Cross-tenant attempts without disclosure or unauthorized change, divided by attempts.	Vary owners, identifier collisions, caches, exports, and jobs.
Enforcement coverage	Inventoried protected operations with tested enforcement, divided by the inventory.	Reconcile API, administrative, message, and database paths.
Workload verification and escalation	Verified internal calls divided by observed calls; escalation attempts prevented without effects divided by attempts.	Probe direct paths, role combinations, and service-purpose bypasses.
Latency and API overhead	Complete decision time and matched response-time difference from the baseline.	Report p50, p95, and p99 by load, cache state, location, and outcome.
Revocation delay	Elapsed time until tested enforcement points consistently reject revoked access.	Probe replicas, cached grants, and in-flight operations.
Evidence completeness	Recoverable required decision and outcome records divided by expected records.	Reconcile independent test identifiers, including early denials and sink faults.
Failure behavior	Count of protected operations without valid permission, with timeout and recovery behavior.	Interrupt identity, attributes, policy delivery, and remote evaluation.
Legitimate completion	Correctly completed authorized operations divided by authorized requests.	Run positive controls with each adversarial batch; report false denials.

Fault injection should interrupt identity issuance, attribute retrieval, policy distribution, remote evaluation, and the evidence sink. The test plan also introduces expired bundles, delayed invalidation, key rotation, and a resource change after permission but before commit. For payment P, execution pauses after the permit for version 7; the beneficiary

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

changes; execution resumes. The original approval must fail to commit the changed instruction. A separate entitlement-revocation test examines the promised freshness guarantee.

Mutation tests deliberately remove a tenant condition, broaden a workload grant, or bypass an enforcement point. The suite should detect each introduced defect. Otherwise, a high pass rate provides little assurance that the controls being claimed were actually exercised. Coverage must reconcile tests with the inventories of routes, service edges, jobs, and data paths.

Latency should be reported by percentile and separated by evaluation location, cache state, load, and decision outcome. Repeated trials should document software configuration, warm-up, sample sizes, and uncertainty. Authentication failures, policy denials, and dependency errors require separate counts. Revocation tests should probe replicas and in-flight work and identify the last observed successful use after authoritative withdrawal of permission.

Acceptance criteria should require no unauthorized disclosure or state change within the defined test corpus, active tested enforcement on inventoried protected operations, and legitimate completion within agreed service objectives. Passing supports the tested implementation, assumptions, and workload. It establishes neither universal security nor acceptable performance at an untested institutional scale.

XI. LIMITATIONS AND RESEARCH PRIORITIES

The architecture depends on trustworthy ownership data, reliable credential issuance, and enforcement paths that cannot be bypassed. A compromised workload can still misuse its legitimate privileges. Authentication does not establish correct behavior, and authorization does not replace fraud detection, financial reconciliation, data validation, or idempotent processing. Legacy shared accounts and undocumented database access may also prevent complete enforcement until their dependencies are understood.

Distributed entitlements limit what a local transaction check can guarantee. Payment versioning can prevent approval of changed details without making external policy updates instantaneous. Cross-system approval, clearing, and settlement may require additional coordination and renewed authorization at later irreversible stages. Institutions must select freshness and availability guarantees that the implementation can actually meet.

Further work should evaluate event-driven execution when authority changes between publication and consumption, workload attestation changes after credential issuance, and federation across trust domains. Policy verification should examine combinations of permissions and service purposes. Adaptive risk signals require explainable effects and measurement of false denials. AI-assisted policy review could support analysis, but proposed changes would still require deterministic checks and accountable approval.

Portability also requires evidence. Two institutions may use different account hierarchies, delegation rules, and approval tiers. A useful next study would implement both ownership models under the same decision contract, then examine whether adapters preserve the required restrictions without hidden exceptions. Only implementation and comparative evaluation can establish the design's operating cost, reusable scope, and practical security value.

XII. CONCLUSION

Default-deny authorization requires a financial service to establish permission for the operation it is about to perform. The proposed contract connects verified workload and user identities with authoritative ownership, approved execution purpose, relevant business state, and enforceable obligations. Resource-level checks preserve these conditions beyond gateway admission and into data access and transaction execution.

The design's central requirements are distinct authority for delegated and service computations, approval bound to reviewed state, tenant restrictions throughout processing, and explicit behavior when required evidence or fresh inputs are unavailable. Its evaluation must test unauthorized financial effects and legitimate service performance together. An independently evaluated implementation is the next step toward determining whether the proposed architecture delivers these properties under realistic workload and failure conditions.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 5, Issue 3, May - June 2023 |

DOI:10.15662/IJEETR.2023.0503010

REFERENCES

[1] R. Chandramouli, Security Strategies for Microservices-based Application Systems, NIST SP 800-204. August 2019. doi: 10.6028/NIST.SP.800-204
[2] Federal Financial Institutions Examination Council, Authentication and Access to Financial Institution Services and Systems. August 2021. Official publication
[3] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, Zero Trust Architecture, NIST SP 800-207. August 2020. doi: 10.6028/NIST.SP.800-207
[4] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308. 1975. doi: 10.1109/PROC.1975.9939
[5] R. Chandramouli and Z. Butcher, Building Secure Microservices-based Applications Using Service-Mesh Architecture, NIST SP 800-204A. May 2020. doi: 10.6028/NIST.SP.800-204A
[6] R. Chandramouli, Z. Butcher, and A. Chetal, Attribute-based Access Control for Microservices-based Applications Using a Service Mesh, NIST SP 800-204B. August 2021. doi: 10.6028/NIST.SP.800-204B
[7] R. Chandramouli and Z. Butcher, A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments, NIST SP 800-207A, initial public draft, April 2023. Initial public draft
[8] 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
[9] P. Phiayura and S. Teerakanok, "A comprehensive framework for migrating to Zero Trust architecture," IEEE Access, vol. 11, pp. 19487-19511. February 2023. doi: 10.1109/ACCESS.2023.3248622
[10] D. Hardt, Ed., The OAuth 2.0 Authorization Framework, IETF RFC 6749. October 2012. doi: 10.17487/RFC6749
[11] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, OpenID Connect Core 1.0 Incorporating Errata Set 1, OpenID Foundation, November 2014. Official specification
[12] V. Bertocci, JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens, IETF RFC 9068. October 2021. doi: 10.17487/RFC9068
[13] Y. Sheffer, D. Hardt, and M. Jones, JSON Web Token Best Current Practices, IETF RFC 8725, BCP 225. February 2020. doi: 10.17487/RFC8725
[14] J. Richer, Ed., OAuth 2.0 Token Introspection, IETF RFC 7662. October 2015. doi: 10.17487/RFC7662
[15] T. Lodderstedt, M. McGloin, and P. Hunt, OAuth 2.0 Threat Model and Security Considerations, RFC 6819, January 2013. doi: 10.17487/RFC6819
[16] B. Campbell, J. Bradley, N. Sakimura, and T. Lodderstedt, OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens, IETF RFC 8705. February 2020. doi: 10.17487/RFC8705
[17] N. Sakimura, J. Bradley, and N. Agarwal, Proof Key for Code Exchange by OAuth Public Clients, RFC 7636, September 2015. doi: 10.17487/RFC7636
[18] D. Feldman et al., Solving the Bottom Turtle A SPIFFE Way to Establish Trust in Your Infrastructure via Universal Identity, first edition, 2020. Official book
[19] M. Jones, A. Nadalin, B. Campbell, J. Bradley, and C. Mortimore, OAuth 2.0 Token Exchange, IETF RFC 8693. January 2020. doi: 10.17487/RFC8693
[20] Joint Task Force, Security and Privacy Controls for Information Systems and Organizations, NIST SP 800-53, rev. 5. September 2020. doi: 10.6028/NIST.SP.800-53r5
[21] Kubernetes Authors, Enforcing Network Policies in Kubernetes, October 2017. Official publication
[22] Cloud Native Computing Foundation, TAG Security, Cloud Native Security Whitepaper, ver. 2.0. May 2022. Official publication
[23] National Security Agency and Cybersecurity and Infrastructure Security Agency, Kubernetes Hardening Guidance, version 1.2, August 2022. Official guidance
[24] OWASP Foundation, OWASP API Security Top 10, 2019 edition. Official publication
[25] PCI Security Standards Council, Payment Card Industry Data Security Standard Requirements and Testing Procedures, version 4.0, March 2022, Requirements 7, 8, and 10. Official standard