Risk-Adaptive Authorization for Hybrid Cloud IAM: Coupling Behavioral Anomaly Scoring with Permission-Flow Reachability

DRAFTED BY:
Aalok Bhandari, Ashal Pandey, Famous Dhungana
Department of Computer Science and Information Technology
Madan Bhandari Memorial College, Tribhuvan University
Kathmandu, Nepal

Abstract

Misconfigured identity and access management is a leading cause of cloud compromise, and privilege escalation is its most damaging consequence. An adversary holding a low-privilege credential chains together individually permitted control-plane operations until administrative authority is obtained; every call is authenticated and authorised, and only the sequence is malicious. Current defences address this partially. Static configuration analysis and model checking establish that an escalation path exists in a policy set, but run offline and cannot distinguish an attacker traversing that path from an administrator performing the same operation in the course of legitimate work. Signature-based cloud detection matches documented API patterns in audit logs after the operation has completed, with no model of how the acting identity normally behaves. Behavioral analytics models principal activity well, though largely on endpoint and directory telemetry rather than control-plane action sequences, and its output is an alert for human triage rather than an authorization verdict. The Zero Trust reference architecture names historical subject behavior as an input to the trust algorithm without specifying that algorithm for cloud identity. This paper proposes a runtime integration of two signals that are individually established but, in the literature surveyed here, not combined at an enforcement point. An inline API broker intercepts every control-plane request, emits normalized telemetry, and serves as the policy enforcement point. A behavioral analytics engine maintains a per-principal baseline over action sequences. A permission-flow graph service computes, for the specific operation requested, the proportion of currently unreachable permissions that the operation would unlock. Both signals and a contextual deviation term form a bounded composite risk score, which a policy engine evaluates against thresholds and resource sensitivity to return allow, step-up authentication or deny before the operation executes. An evaluation program is specified on a fully open-source emulated hybrid testbed reproducible at no infrastructure cost, using a corpus generated from scripted benign and adversarial workloads, four baseline configurations of which two are ablations isolating each risk term, and metrics covering detection, authorization latency and resource overhead. *The program has been designed but not executed. Every result cell is marked as pending execution; no detection, latency or resource figure in this paper is an obtained measurement, and the case study of Section 6 is analytical rather than empirical.*

Keywords—Zero Trust Architecture, continuous authorization, hybrid cloud, identity and access management, privilege escalation, permission-flow graph, behavioral analytics, anomaly detection, risk-adaptive access control, policy decision point, policy enforcement point, isolation forest, Open Policy Agent, MITRE ATT&CK

1. INTRODUCTION

Enterprise infrastructure is now provisioned, modified and destroyed through APIs. The network perimeter has stopped delimiting trust: workloads run across provider regions and on-premises datacentres at once, service identities outnumber human ones, and every administrative act of consequence is an authenticated call to a control-plane endpoint. What an actor can do follows from identity, not from where the packet originated.

Hybrid deployment sharpens the problem. An organization running both a cloud tenancy and on-premises services federates one identity across two administrative domains that differ in policy language, audit format and enforcement semantics. Authority legitimate in one plane frequently carries into the other, and the union of permissions available to a federated principal is seldom visible to any single administrator. Responsibility for the correctness of those permission configurations rests with the customer rather than the provider [2].

Permissions accumulate. Entitlements provisioned for one deployment task are rarely revoked; roles become assumable by more principals than intended; a policy written to unblock an incident survives the incident. Configurations of any age therefore tend to contain latent paths along which a low-privilege entity can reach high-privilege authority without any software vulnerability being exploited.

This paper is concerned with escalation performed through the control plane itself. An adversary who obtains a low-privilege credential enumerates their own entitlements, locates an operation that propagates permissions, and invokes it. Systematic study of permission acquisition in a major provider identifies a small number of recurring categories, unified by the observation that in each one permissions spread between entities [6]. The Capital One compromise remains the canonical instance of the pattern: an over-permissive role, reached through a legitimate mechanism, converted into access far beyond what any single grant intended.

Existing defences fail in complementary directions. Static entitlement analysis and bounded model checking determine whether an escalation path exists in a configuration [7], [10]; both run offline and observe no one traversing the path. Enumerated escalation catalogs [8], [9] are concrete and operationally validated, but fixed, and they emit no runtime signal. Signature matching over audit logs [11] fires once the operation has succeeded. Behavioral analytics captures how a principal normally acts [13], [14], though its development has centered on endpoint and directory telemetry, and its output is an alert rather than a verdict. NIST SP 800-207 [1] prescribes continuous re-evaluation and names behavior among the trust algorithm's inputs, while leaving the algorithm itself abstract.

The observation motivating this work is that two of these strands are complementary rather than competing. Reachability analysis knows which operations are dangerous in a given configuration but nothing about who is invoking them. Behavioral modeling knows what is unusual for a principal but nothing about what a particular operation would unlock. Evaluated together, at a point on the request path that can refuse, they yield a decision neither supports alone.

This paper makes the following principal contributions:

Runtime coupling of behavior to enforcement: An architecture aligned to the logical decomposition of NIST SP 800-207 [1] and SP 800-207A [2] places an inline API broker on the control-plane request path, so that the component observing behavior is also the component able to refuse it. The verdict precedes execution, which removes the remediation step that detection-oriented designs require. Section 3 specifies each component's inputs, outputs, synchrony and behavior under failure.

A composite risk function over behavior and permission flow: A bounded score combines a normalized behavioral anomaly term over per-principal action sequences, a permission-flow gain measuring what the requested operation would unlock, and a contextual deviation term. Variables, normalization, domain constraints, degenerate cases and thresholds are stated in full, and the mapping from score to verdict is given as an explicit matrix. The reachability formulation underlying the second term is due to prior work [6], [10]; its evaluation per request as an input to an authorization decision, rather than periodically as a configuration audit, is what is proposed here.

An evaluation design that is reproducible at zero cost: The program runs on open-source components with no cloud account. Its corpus is generated inside the testbed from scripted benign and adversarial workloads, so training and target domains coincide and labels are exact by construction. Two of the four baseline configurations are ablations that zero one risk term each, which is what makes the claimed contribution falsifiable rather than merely plausible.

Section 2 establishes the foundations and derives the research gap from a comparative analysis. Section 3 specifies the architecture. Section 4 develops the authorization mechanism. Section 5 presents the evaluation methodology and its threats to validity. Section 6 works through an escalation scenario analytically. Section 7 examines security properties, threat model and limitations. Section 8 concludes.

2. THEORETICAL FRAMEWORK

2.1 Zero Trust Architecture and the Unspecified Trust Algorithm

NIST SP 800-207 defines zero trust as a set of paradigms that move defence from static network perimeters onto users, assets and resources, assuming no implicit trust from network location or asset ownership [1]. An implementation decomposes into a policy decision point, comprising policy engine and policy administrator, and a policy enforcement point that establishes, monitors and terminates the subject-resource connection. SP 800-207A extends this to cloud-native applications spanning multiple cloud and on-premises locations, treating service identity and the placement of enforcement in API-mediated environments [2].

Two properties of this guidance shape the present work. It is architectural rather than algorithmic: input categories for the policy engine are enumerated, but not how heterogeneous inputs are quantified, combined, or mapped to an action. And although re-evaluation is continuous in principle, deployments commonly re-evaluate on coarse triggers such as token expiry or network change. The interval between triggers is the window in which an escalation sequence completes. Surveys of the area report continuous authentication and risk-aware access control as recurring recommendations while noting the scarcity of concrete, reproducible instantiations of the trust algorithm [3], [4].

2.2 Hybrid Deployment and the Shared Responsibility Boundary

Combining private infrastructure with public cloud services under unified management produces more than an additive security problem. Each environment keeps its own authorization semantics — directory group membership on one side, policy documents attached to principals and roles on the other — while federation lets a subject authenticated in one act in the other. Monitoring, by contrast, is usually built separately for each.

The shared responsibility model assigns physical infrastructure and managed service implementations to the provider, and identities, permissions, configuration and data to the customer. Provider audit logs capture control-plane API activity: operations that create, modify, enumerate or delete resources and identity objects. They do not capture activity inside a compute instance below the API layer. Any architecture built on this telemetry therefore inherits a definite visibility boundary, treated in Section 7.4 as a limitation on what the design can be said to cover. Within that boundary the telemetry suits behavioral analysis well, being structured, attributed to a named principal, and inclusive of exactly the operations by which permissions change [21].

2.3 Privilege Escalation and Permission Flow

Cloud IAM authorises through policy documents that enumerate permitted actions on specified resources, evaluated against every request [21]. Escalation becomes possible when a principal's existing permissions allow it to modify the permission model or to assume an identity with greater authority. Documented primitives include creating policy versions, attaching managed policies to principals, writing inline role policies, passing a privileged role to a service that will assume it, and creating additional credentials for another principal [8].

Analysis of permission acquisition in a major provider identifies five categories — permission request, permission delegation, permission inheritance, code execution and credential takeover — sharing the property that permissions flow between entities, a relation representable as a graph and analyzable by reachability [6]. Related work encodes IAM state as a finite Boolean model and applies model checking to find multi-step transitive escalation [7]; graph-based tooling computes transitive authentication chains from a configuration snapshot [10]. All of these are offline. They establish that a path exists, not that a principal is walking it,

and they cannot tell an attacker from an administrator whose job requires the same operation. The MITRE ATT&CK cloud matrix [5] supplies the technique taxonomy used to structure the threat model in Section 7.2; being an enumeration of observed techniques, it cannot by construction characterize novel ones, which is one argument for pairing pattern logic with behavioral modeling.

2.4 Behavioral Analytics on Control-Plane Telemetry

Behavioral analytics builds a statistical profile of normal activity per principal and scores subsequent activity by its deviation. On identity telemetry the relevant features concern which operations a principal invokes, in what volume and sequence, at what hours, from which origins, and against which resource classes. The approach does not require that a malicious action have been anticipated, only that it be unusual for the principal performing it.

Labeled data is the binding constraint. Escalation is rare relative to routine administration, producing severe class imbalance, and organizations do not publish labeled records of their own compromises. Unsupervised methods are the practical default: isolation forest [15] and one-class support vector machines [16] train on predominantly benign activity and score deviation without labeled positives. Work on the principal public insider-threat corpus [13] establishes that the granularity at which activity is aggregated materially affects detection performance [14], which informs the windowing choice in Section 4.1. Three weaknesses recur across this literature and carry into the present design: false positives when a principal's legitimate role changes; degradation when adversary activity is present during training and is absorbed into the baseline; and susceptibility to gradual evasion by an adversary pacing activity slowly enough to be learned as normal. Section 7.4 addresses each.

2.5 Comparative Analysis and Research Gap

Table I compares representative work across these foundations, with each entry assessed against runtime prevention of control-plane escalation rather than against its own stated goal.

TABLE I — COMPARATIVE ANALYSIS OF EXISTING APPROACHES AND DERIVED RESEARCH GAP

Work	Approach	Env.	Limitation for runtime prevention
Rose et al. [1]	ZTA reference model	Enterprise	Descriptive; trust algorithm left abstract; no evaluation
Chandramouli, Butcher [2]	ZTA for cloud-native, multi-location	Hybrid	Guidance; no implemented or evaluated system
Gietzen [8]; Pacu [9]	Enumerated escalation patterns	Cloud IAM	Fixed catalog; offline; no runtime signal
PMapper [10]	Permission graph reachability	Cloud IAM	Configuration-time snapshot; one acquisition category
Shevrin, Margalit [7]	Bounded model checking of IAM state	Cloud IAM	Hand-crafted encodings; offline; no behavioral input
Hu et al. [6]	Permission-flow graph; five-category taxonomy	Cloud IAM	Static; attacker and administrator are indistinguishable
Splunk [11]	Signature rules over audit logs	Cloud	Fires after execution; no model of the acting identity
CERT-based UEBA [13], [14]	*Behavioral modeling of user activity*	*Endpoints*	*Endpoint telemetry; output is an alert, not a verdict*
This proposal	Behavior and permission-flow reachability at an inline PEP	Hybrid (emulated)	Control-plane visibility only; corpus generated, not observed

Three literatures appear in Table I and they do not meet. Reachability analysis proves paths exist without observing their use. Signature detection observes known operations without modeling the identity invoking them, and reports after the fact. Behavioral analytics models principals but on the wrong telemetry and to the wrong output. Zero Trust guidance specifies where behavior should enter the decision without specifying what to measure.

Stated precisely, the gap is one of combination and operationalisation rather than of absence. Each ingredient exists: behavioral anomaly scoring [13]–[16], permission-flow reachability [6], [10], policy-as-code enforcement [18], and the architectural pattern that would host them [1], [2]. What the surveyed literature does not provide is an account in which a behavioral score over control-plane action sequences and a reachability computation performed for the specific operation requested are combined into a single bounded quantity, evaluated at the granularity of the individual request by an enforcement point positioned before execution, and assessed on a testbed another researcher can rebuild. The claim is confined to that integration and its evaluation. No claim is made that the underlying techniques are new, and the survey in Table I is not offered as an exhaustive proof that no such system exists anywhere.

3. SYSTEM ARCHITECTURE

3.1 Architectural Overview

The design places an inline broker on the control-plane request path. Requests from human and service principals terminate at the broker, which authenticates the caller, normalizes the request into an identity-event record, obtains a verdict, and forwards or refuses accordingly. Two analyzers consume the intercepted stream: one scores behavior, the other computes what the requested operation would unlock. A risk engine combines their outputs with a contextual term, and a policy decision point maps the composite to a verdict. Figure 1 shows the composition.

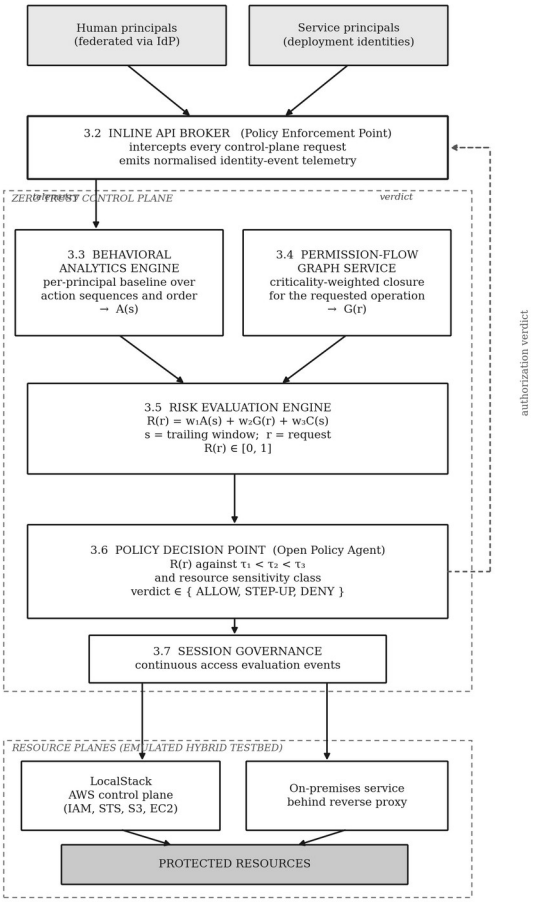


Figure 1: End-to-end architecture. Every control-plane request traverses the inline broker, which acts as both telemetry source and policy enforcement point, so that the authorization verdict precedes execution of the requested operation.

Component boundaries follow the decomposition of NIST SP 800-207 [1]. The behavioral and permission-flow services jointly constitute the policy information function; the policy decision point corresponds to the policy engine and administrator; the broker is the policy enforcement point. Positioning enforcement inline rather than downstream is the decision from which the architecture's principal property follows, and also its principal risk: a refused operation never executes and requires no remediation, but a broker that is bypassed or unavailable removes the control entirely. Section 3.8 treats that dependency.

3.2 Inline API Broker

The broker intercepts control-plane API calls, meaning operations against identity, configuration and resource-management endpoints. Data-plane traffic — reads and writes of object content — is outside its scope and outside the telemetry this design consumes. Authentication is performed at the broker against the federated identity provider [20], establishing the calling principal. Authorization is not performed at the broker: the broker requests a verdict from the policy decision point and applies it. Keeping the two separate matters because it allows policy to be audited independently of the proxy implementation.

For each request the broker emits a normalized record carrying principal identifier, timestamp, action name and family, target resource and sensitivity class, source origin, and originating plane. It then applies the returned verdict: forward unchanged, hold pending a step-up challenge, or refuse. Because the broker sits on the latency path of every request, authorization overhead is measured as a first-class quantity in Section 5.6 rather than reported as an afterthought.

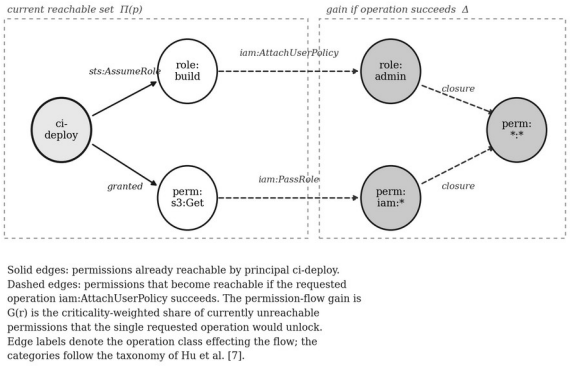
3.3 Behavioral Analytics Engine

The engine runs asynchronously. Blocking a request until a fresh window score is computed would couple request latency to the window duration, which is untenable for windows of any useful length. Instead the engine consumes the normalized event stream continuously, maintains a per-principal baseline profile, and writes the current anomaly score for each principal to a store the risk engine reads. Scores are recomputed on a fixed cadence and on the arrival of an identity-mutating event, whichever comes first.

The consequence is that the behavioral term is always slightly stale relative to the request being decided. That staleness is bounded by the recomputation cadence, it is measured in Section 5.6 as a reported quantity, and it is a genuine cost of the design: an escalation completed entirely within one cadence interval is decided on a behavioral score that predates it. The permission-flow term, which is computed synchronously, is what covers that interval.

3.4 Permission-Flow Graph Service

The service maintains a directed graph whose nodes are principals, roles and permission sets, and whose edges are operations that propagate permissions between them, following the acquisition categories established in prior work [6]. A principal's reachable permission set is the transitive closure from its node. Queried with a specific requested operation, the service computes the closure that would obtain were the operation to succeed and returns the difference. Figure 2 illustrates the structure.



Solid edges: permissions already reachable by principal ci-deploy. Dashed edges: permissions that become reachable if the requested operation iam:AttachUserPolicy succeeds. The permission-flow gain is G(r) is the criticality-weighted share of currently unreachable permissions that the single requested operation would unlock. Edge labels denote the operation class effecting the flow; the categories follow the taxonomy of Hu et al. [7].

Figure 2: Permission-flow graph fragment. The principal ci-deploy currently reaches a limited permission set; the requested operation would extend the closure to administrative authority. The normalized size of that extension is the permission-flow gain defined in Section 4.3.

This query is synchronous and blocking, because its answer is specific to the request under decision and cannot be

precomputed. The graph is rebuilt whenever an identity-mutating operation is permitted, so it reflects the configuration in force rather than a periodic snapshot; between rebuilds it is exact, because the only operations that can invalidate it are precisely those that trigger a rebuild. This invariant holds only while every mutating path passes through the broker, which is the assumption Section 3.8 examines. Rebuild cost bounds throughput and is measured in Section 5.6.

3.5 Risk Evaluation Engine

The engine reads the current behavioral score for the calling principal, receives the permission-flow gain for the requested operation, derives the contextual term from the trailing window, and emits the bounded composite defined in Section 4.4. It exists as a separate component because deviation and consequence are different quantities. An unusual but inconsequential action and a routine but permission-unlocking one call for different treatment, and a single model output would make that distinction inexpressible in policy.

3.6 Policy Decision Point

The decision point evaluates the composite against configured thresholds and the sensitivity class of the target resource, returning allow, step-up or deny together with the rule and inputs that produced it. Policy is expressed as code in a general-purpose policy engine [18], making it version-controlled, reviewable and testable independently of the analytics pipeline. Decision logs are retained for audit. The matrix is given in Table V.

Step-up needs concrete definition, since the term is used loosely across the literature. Here the broker rejects the pending request with a challenge response and holds the operation rather than discarding it. The principal re-authenticates to the identity provider [20] with a second factor, a time-based one-time password registered against the account, and the token subsequently issued carries an elevated authentication context reference together with an authentication methods claim recording that the second factor was presented. The broker admits the held operation only against a token bearing that claim, and the elevation attaches to the session for a configured lifetime rather than to the single request. In the terms of NIST SP 800-63B [17], this raises the authenticator assurance level of the session.

Service principals cannot satisfy an interactive challenge, and the asymmetry matters. Where the caller is non-interactive, step-up is unavailable and the decision point substitutes deny: a machine identity behaving anomalously against a sensitive resource has no in-band way to prove itself, and recovery becomes an operator action. The design is therefore stricter for service identities than for human ones, which suits the threat model, since the compromised credential in Section 6 belongs to a deployment principal.

3.7 Session Governance

Where a verdict invalidates authority already granted within the session, the governance component propagates a revocation signal to the identity provider so that existing sessions terminate rather than merely failing at the next request. Every decision is recorded with its score, its three contributing terms and the governing rule, so an enforcement action can be reconstructed and contested afterwards. Revocation is rate-limited per principal, preventing a false positive from becoming sustained denial of service against a legitimate operator.

3.8 Failure Modes, Degraded Operation and Bypass

Three failure conditions require a defined response. If the behavioral store is unavailable or its score for a principal exceeds the staleness bound, the risk engine substitutes the population mean anomaly score and marks the decision as degraded; the permission-flow and contextual terms continue to contribute, so escalation-relevant operations remain covered. If the permission-flow service is unavailable, the decision point fails closed for operations against high-sensitivity resources and falls back to behavioral scoring alone for the remainder, on the reasoning that the term being lost is precisely the one that protects identity objects. If the decision point itself is unreachable, the broker fails closed. Each degraded decision is labeled as such in the audit record, so that periods of reduced assurance are visible after the fact rather than indistinguishable from normal operation.

Bypass is the more serious concern, and the architecture does not solve it alone. Every property claimed here holds only while the broker is the sole path to the control plane, and that invariant must be established by the provider rather than by this design. On a major provider it would rest on three controls used together. An organization-level service control policy denies all identity, key-management and audit-configuration actions unless the requesting principal is the broker's own role, expressed as a Deny statement conditioned on the principal ARN, which no account-level policy can override. A condition key restricting control-plane calls to the broker's private endpoint refuses requests arriving over the public API surface. And long-lived access keys are disabled in favor of short-lived session credentials issued only through the broker, which removes the credential type the adversary of Section 6 is holding.

Two gaps survive even that arrangement. The broker's role becomes the highest-value credential in the environment, so compromising it is equivalent to compromising the control plane. And the emulator used in Section 5 implements none of these controls, since it provides neither organization-level policy nor endpoint conditions. The mechanism is therefore specified here without evidence that it holds, and validating it on a real provider account is named in Section 8 as the prerequisite to any deployment claim.

4. CONTINUOUS RISK-ADAPTIVE AUTHORIZATION MECHANISM

4.1 Decision Unit and Session Representation

The unit of decision is a single request. Let r denote a request issued by principal p at time t, requesting operation o against resource q. Let s(r) denote the session window for that request: the events attributed to p in the interval of duration W ending at t. Windows advance by a stride of W/2 so that consecutive windows overlap and a sequence spanning a boundary is not split across independently scored units. W is fixed across all configurations and its value is recorded in Table VIIII; prior work establishes that

aggregation granularity materially affects detection performance [14], so treating it as a free parameter tuned per configuration would confound the comparison.

Distinguishing r from s(r) matters for the formulation that follows. The behavioral and contextual terms are properties of the window; the permission-flow term is a property of the individual operation requested. Conflating them would leave the model unable to express the case that motivates the design, in which a principal behaving unremarkably requests one operation of large consequence. Table II gives the feature set over s(r). Continuous features are standardized using statistics computed on the training partition alone. The order family deserves comment. Counting which actions appear in a window discards the sequence in which they appeared, yet sequence is what distinguishes reconnaissance followed by escalation from the same operations performed in an unrelated order during routine work. Three features recover it: the distribution of action-family bigrams, the mean log-probability of the observed transitions under a first-order transition matrix estimated per principal during training, and a binary indicator that an enumeration call was followed within the window by a permission-propagating one. The third is deliberately coarse and interpretable, so that a verdict driven by it can be explained to an operator.

TABLE II — SESSION-WINDOW FEATURE SET

Family	Feature	Type	Rationale
Volume	Event count in window	Continuous	Enumeration raises activity above a principal's norm
Volume	Distinct actions invoked	Continuous	Permission discovery invokes a broad range of reads
Volume	Failed-authorization count	Continuous	Permission probing produces access-denied volume
Category	Action-family distribution	Vector	Shifts in read, write and identity operation mix
Category	Identity-mutating action present	Binary	Signals capability to alter the permission model
Category	Unseen action for principal	Binary	Action absent from the baseline profile
Category	Resource sensitivity class	Ordinal	Distinguishes low- from high-value targets
Temporal	Hour-of-day deviation	Continuous	Off-hours activity is a recurring compromise indicator
Temporal	Inter-event interval statistics	Continuous	Tooling shows regularity unlike human operators
Order	Action-family bigram distribution	Vector	Captures which transitions occur, not merely which actions
Order	Transition likelihood under baseline	Continuous	Mean log-probability of observed transitions given the principal's historical transition matrix
Order	Enumeration-to-mutation transition	Binary	The ordering that characterizes escalation: discovery followed by a permission-propagating call
Context	Unseen source origin	Binary	New origin for an established principal
Context	Cross-plane activity	Binary	On-premises authentication then cloud action in one window
Context	Role-assumption depth	Ordinal	Chained assumption characterizes lateral movement

4.2 Baseline Profiling and Anomaly Detection

Each principal's baseline profile is built from a training partition containing benign activity only, recording the empirical distribution of every continuous feature and the observed support of every categorical one. Partitioning is specified in Section 5.2 so that adversarial activity cannot enter training. Principals without sufficient history are treated as cold-start rather than assigned an average score. Substituting the population mean would give an unrecognized identity the treatment of a typical one, which is the wrong default when the identity may not be legitimate at all. Instead A(s(r)) is set to the seventy-fifth percentile of the population anomaly distribution, and Section 4.5 imposes a verdict floor that prevents a cold-start principal from being silently allowed against a sensitive resource. Cold-start decisions are flagged in the audit record so that assurance taken on weak evidence remains distinguishable afterwards.

An isolation forest [15] serves as the primary model: it needs no labeled positives, scales to the corpus sizes involved, and exposes few hyperparameters, which matters when reproducibility is an objective. A one-class support vector machine [16] is retained for comparison. Their relative performance is an experimental question that Section 5 is designed to answer and that this paper does not prejudge.

4.3 Permission-Flow Gain

Let $\Pi(p)$ be the set of permissions reachable by principal p under the configuration in force, computed as the transitive closure over the permission-flow graph, and let $\Pi'(p, o)$ be the closure that would obtain were operation o to succeed. Let Π_{max} be the permission universe of the environment.

Counting permissions treats them as interchangeable, which they are not. Acquiring a single action that can rewrite the permission model is more consequential than acquiring twenty read-only descriptive calls, and a gain defined on set cardinality alone would score the second higher. Each permission π therefore carries a criticality weight $\kappa(\pi)$ drawn from the classes in Table III, and the gain is computed over weighted mass rather than count. The permission-flow gain of request r is

$$G(r) = \Sigma\, \kappa(\pi) \text{ over } \Pi'(p,o) \setminus \Pi(p) \;\div\; \Sigma\, \kappa(\pi) \text{ over } \Pi_{max} \setminus \Pi(p)$$

that is, the criticality-weighted share of currently unreachable permissions that this one operation would unlock. The numerator is zero for any operation propagating no permissions, so $G(r) = 0$ across the majority of routine traffic. It equals the denominator when the operation opens the full closure, giving $G(r) = 1$. Monotonicity holds by construction:

$\Pi(p) \subseteq \Pi'(p, o)$ and every $\kappa(\pi)$ is positive, so the numerator is non-negative and cannot exceed the denominator, and $G(r) \in [0, 1]$ without clamping.

TABLE III — PERMISSION CRITICALITY CLASSES (PROTOTYPE WEIGHTS)

Class	Membership	κ(π)	Rationale
Critical	Identity and access management writes, key-management operations, audit-configuration changes	25	Can rewrite the permission model or conceal its own use
Elevated	Compute control, role passing, data-store writes	5	Enables lateral movement or data modification without directly altering entitlement
Routine	Read, list and describe operations	1	Informative to an adversary but cannot itself change authority

The values 25, 5 and 1 are prototype settings, chosen so that acquiring one critical permission outweighs acquiring roughly twenty routine ones. They are configuration rather than findings, and they belong to the parameter set examined by the sensitivity analysis of Section 5.6. What the ratio should be in a production environment is an empirical question this work does not answer.

Algorithm 1 states the computation. Both closures are ordinary graph reachability, so the cost is linear in the edges of the subgraph reachable from p rather than in the size of the whole graph, and the second closure differs from the first only by the edges the requested operation would add.

Algorithm 1 Permission-flow gain for a requested operation

```
input : graph D, principal p, requested operation o,
        criticality weights kappa
output: G in [0,1]

1  P   <- Reach(D, p)          // current closure
2  D'  <- D + EdgesAdded(o)    // hypothetical
3  P'  <- Reach(D', p)         // closure after o
4  gain <- Mass(P' \ P, kappa)
5  room <- Mass(Umax \ P, kappa)
6  if room = 0 then return 0   // saturated
7  return gain / room

Reach(D, p):
8  S <- {}; Q <- {p}
9  while Q not empty:
10     n <- pop(Q)
11     for each edge n -> m in D:
12         if m not in S: add m to S; push m to Q
13 return permissions in S

Mass(X, kappa): return sum of kappa(x) for x in X
```

Two degenerate cases require definition. When $\Pi(p) = \Pi_{max}$ the denominator vanishes; G(r) is defined as zero in this case, since a principal that already reaches every permission cannot escalate further, and the operation should be governed by the behavioral and contextual terms alone. When p is unknown to the graph, G(r) cannot be computed at all, and the request is handled by the cold-start rule of Section 4.5 rather than by substituting a numeric stand-in. Assigning such a request an average score would treat an unrecognized identity as ordinary, which inverts the appropriate default.

One property of this construction constrains interpretation. G(r) is normalized against the permission universe of the environment in which it is computed, so its values are comparable within a deployment but not across deployments of different size. A gain of 0.4 in a small testbed and a gain of 0.4 in a large production tenancy do not denote the same absolute consequence. Thresholds calibrated on one environment therefore do not transfer to another without recalibration, which Section 7.4 records as a limitation on generalizability.

4.4 Composite Risk Formulation

Writing A(s(r)) for the normalized behavioral anomaly score of the window and C(s(r)) for its contextual deviation, the composite risk of request r is

$$R(r) = w_1 A(s(r)) + w_2 G(r) + w_3 C(s(r))$$

A(s(r)) is obtained by mapping the raw model output onto the unit interval using the fifth and ninety-fifth percentiles of the anomaly-score distribution on the training partition, with values outside that range clamped. Percentile mapping is preferred to min-max scaling, which a single extreme training value would dominate. C(s(r)) is the arithmetic mean of the triggered binary contextual features of Table II, and therefore lies in [0, 1] by construction; using the mean rather than a count keeps the term independent of how many contextual features the feature set happens to contain, so that adding a feature does not silently reweight the composite.

The weights are non-negative and satisfy $w_1 + w_2 + w_3 = 1$. Since each term lies in [0, 1], $R(r) \in [0, 1]$ follows as a convex combination, and no clamping is applied at this stage. The weights are not fixed a priori: their selection is an experimental parameter, and Section 5.6 specifies a sweep across the weight simplex reporting how detection quality and false-positive rate vary. Fixing weights by assertion would place the formulation beyond falsification, which is the failure this design is meant to avoid. A starting point is nonetheless required to run the prototype at all, and the settings used are $w_1 = 0.4$, $w_2 = 0.4$, $w_3 = 0.2$, with $\tau_1 = 0.35$, $\tau_2 = 0.55$ and $\tau_3 = 0.75$. These are prototype values, selected so that the behavioral and permission-flow terms carry equal influence and the contextual term acts as a modifier rather than a driver. They are not tuned, not derived from data, and carry no claim of optimality; the sweep of Section 5.6 exists to replace them.

TABLE IV — RISK SCORE COMPONENTS AND DETERMINATION

Symbol	Quantity	Domain and normalization	Determination
A(s(r))	Behavioral anomaly score of window	[0,1]; percentile map using P5/P95 of training distribution, clamped	Isolation forest; asynchronous, staleness-bounded

Symbol	Quantity	Domain and normalization	Determination
G(r)	Permission-flow gain of requested operation	[0,1] by construction; zero if $\Pi(p) = \Pi_{max}$	Transitive closure, computed synchronously per request
C(s(r))	Contextual deviation of window	[0,1]; mean of triggered binary contextual features	Table II context family
w_1, w_2, w_3	Component weights	Non-negative, summing to 1	Weight-simplex sweep (Section 5.6)
τ_1, τ_2, τ_3	Decision thresholds	$0 < \tau_1 < \tau_2 < \tau_3 < 1$	Precision–recall operating curve
W	Session window duration	Fixed across all configurations	Value recorded in Table VIII
Prototype settings	$w_1, w_2, w_3 = 0.4, 0.4, 0.2$; $\tau_1, \tau_2, \tau_3 = 0.35, 0.55, 0.75$	Starting values only	To be replaced by the Section 5.6 sweep

4.5 Policy Evaluation and Decision Logic

The decision point maps R(r) to a risk band using the three thresholds and combines the band with the sensitivity class of the target resource. Allow forwards the request. Step-up holds it pending an additional authentication factor and marks the session provisional. Deny refuses the operation and, where authority was granted earlier in the same window, triggers revocation through the governance component. Coupling band to sensitivity rather than acting on the band alone enforces proportionality: identical deviation warrants different responses depending on what is being reached for.

TABLE V — AUTHORIZATION DECISION MATRIX (RISK BAND × RESOURCE SENSITIVITY)

Risk band	Low sensitivity	Medium sensitivity	High sensitivity
$R(r) < \tau_1$	Allow	Allow	Allow
$\tau_1 \le R(r) < \tau_2$	Allow	Step-up	Step-up
$\tau_2 \le R(r) < \tau_3$	Step-up	Step-up	Deny
$R(r) \ge \tau_3$	Step-up	Deny	Deny

The matrix is monotone non-decreasing along both axes: raising the risk band or the sensitivity class never returns a weaker action. Low-sensitivity resources are capped at step-up and never reach deny, which is a policy choice rather than an oversight — refusing read-only descriptive operations imposes a continuity cost that the threat model does not justify, since such operations cannot themselves alter the permission model. An administrator who disagrees can change this in policy without touching the analytics, which is one reason for expressing the matrix as code.

Two overrides sit above the matrix. A cold-start principal — one with no baseline profile, or one absent from the permission-flow graph — receives at minimum a step-up verdict for any medium- or high-sensitivity target, whatever R(r) evaluates to. The matrix can raise that floor but not lower it. This closes the case in which an unrecognized identity would otherwise be allowed through on a score computed from absent evidence. Where the cold-start principal is non-interactive, the substitution rule of Section 3.6 converts the floor to deny.

Sensitivity classes are assigned at configuration time: identity and access management objects, key management resources and audit configuration are high; data stores and compute resources medium; descriptive and read-only endpoints low. The classification is a policy input rather than a model output, and is therefore reviewable independently of anything the analytics produces.

5. PROPOSED PERFORMANCE EVALUATION

This section specifies methodology, testbed, corpus, baselines, metrics and threats to validity. The program has been designed and not executed. Result tables appear with their structure and metric definitions fixed and their values marked as pending execution, so that the evaluation can be carried out and reported without adjusting the criteria after seeing the outcome.

5.1 Testbed Specification

The testbed is a container composition on a single commodity workstation, requiring no cloud account and incurring no cost. A local emulator with policy enforcement enabled supplies cloud IAM semantics [19]; an open-source identity provider supplies federation [20]; a general-purpose policy engine supplies decision [18]. The on-premises plane is a containerised service behind a reverse proxy. Two distinct authorization domains under one federated identity is the minimum configuration exhibiting the property the architecture addresses, and it is what the testbed provides.

TABLE VI — TESTBED SPECIFICATION

Element	Specification
Cloud plane	Local AWS emulator with IAM policy enforcement enabled [19]
Identity provider	Open-source IdP federating both planes [20]
Policy engine	Open Policy Agent, decision logs enabled [18]
Inline broker	Python reverse proxy on the control-plane request path
On-premises plane	Containerised service behind reverse-proxy enforcement point
Analytics	Python; scikit-learn isolation forest and one-class SVM
Host	Commodity workstation; exact specification recorded at experiment time
Composition	Docker Compose with pinned image digests, released as artifact

5.2 Corpus Generation and Partitioning

No public corpus supplies labeled cloud control-plane telemetry with an adequate benign baseline. The two candidates were examined and rejected for metric computation. The CERT insider threat corpus carries exact labels but records on-premises endpoint activity [12], so figures measured on it would not transfer to the control plane. The publicly released flaws.cloud CloudTrail corpus is genuine cloud telemetry [11], but consists predominantly of attack traffic against a deliberately vulnerable training environment and carries no per-event labeling, supplying

neither the benign baseline that profiling requires nor the ground truth that precision and recall presuppose. Both are retained for validating the action taxonomy rather than for producing numbers. The corpus is therefore generated inside the testbed. Benign workload scripts emulate routine operator and deployment activity across a population of principals over a simulated period, with inter-event timing drawn from parameterised distributions under fixed seeds. Adversarial scenarios are injected at known times, making ground-truth labels exact by construction rather than annotated after the fact.

Partitioning is temporal and strict. The corpus divides into a training partition containing benign activity only and an evaluation partition containing benign activity with adversarial scenarios injected. Baseline profiles and anomaly models are fitted on the training partition alone; feature standardisation statistics and anomaly-score percentiles are computed there and applied unchanged to the evaluation partition. No adversarial event appears in training, so the baseline-poisoning failure mode is excluded by design in this evaluation — which is a property of the experiment, not of the deployed system, and Table X records poisoning as uncovered for that reason. Scenario labels are used only to score verdicts after the fact; no label is available to any component at decision time.

Two of the five acquisition categories are covered, with one scenario chaining them within the cloud plane and one chaining them across planes. Reducing scope was necessary to permit adequate repetitions and statistical treatment within the project period; the remaining three categories are identified in Section 8. Table VII lists the catalog.

Because the population of principals is generated rather than observed, the class balance is a design parameter rather than a discovered property. Adversarial requests are injected at a low proportion of total request volume, matching the imbalance characteristic of the problem, and the exact ratio is recorded with the results. Reporting a precision–recall curve rather than a single operating point is what makes the metric interpretable under that imbalance; no resampling is applied, since the models are unsupervised and never see labels.

TABLE VII — ATTACK SCENARIO CATALOG

ID	Category [6]	Primitive	Description
S1	Permission delegation	AttachUserPolicy	Compromised principal attaches an administrative managed policy to itself
S2	Permission delegation	PutUserPolicy	Inline policy written granting identity-mutating actions
S3	Credential takeover	CreateAccessKey	Additional access key created for a higher-privileged principal
S4	Credential takeover	UpdateLoginProfile	Console credential set on a principal held by another party
S5	Multi-step chain	Enumeration then S1	Permission discovery followed by delegation within one session
S6	Cross-plane chain	On-premises logon then AssumeRole then S1	Directory credential used on-premises, federated into the cloud plane, then delegation attempted
B0	Benign control	Mixed operations	Legitimate administrator performing S1-equivalent operations in routine work

Scenario B0 carries the weight of the argument. It is the case that separates this design from static analysis, which cannot distinguish an attacker from an administrator performing the identical operation. A system flagging B0 at the rate it flags S1 has demonstrated nothing beyond what a configuration scanner already provides, and the B0 false-positive rate is therefore reported as a headline usability metric rather than a supplementary one.

5.3 Baselines and Ablations

Four configurations are compared against the proposal. B1, static entitlement, applies IAM policy with no re-evaluation and fixes the floor. B2, signature detection, matches known escalation patterns in the telemetry stream. B3 sets $w_2 = 0$, removing the permission-flow term and leaving behavior and context. B4 sets $w_1 = 0$, removing the behavioral term and leaving an approximation of configuration-time reachability analysis operating at request time.

The ablations are what make the contribution falsifiable. If the full system fails to outperform both B3 and B4, then combining the two signals adds nothing over whichever performs better alone, and the central claim of the paper does not hold. They cost nothing beyond re-running with one weight set to zero, so there is no efficiency argument for omitting them. All five configurations use identical data partitions, identical seeds and the same tuning procedure, and that procedure is reported alongside the results.

5.4 Metrics and Unit of Classification

The unit of classification is the individual request. A request belonging to an injected adversarial scenario that receives a step-up or deny verdict counts as a true positive; a benign request receiving either verdict counts as a false positive. Fixing the unit explicitly matters because per-window and per-scenario definitions yield different numbers from the same run, and comparisons across published work are frequently incommensurable for exactly this reason.

Detection is reported by precision, recall, F1 and area under the precision–recall curve, with recall broken out per scenario category alongside the aggregate. The precision–recall curve is preferred to the receiver operating characteristic because under severe imbalance a high ROC area coexists comfortably with an unusable alert volume. Accuracy is not reported, being uninformative here. The B0 false-positive rate is reported separately. Authorization overhead is wall-clock time added to the request path, at median, ninety-fifth and ninety-ninth percentiles, measured against a baseline condition in which the broker proxies without risk evaluation, so that the cost of the risk pipeline is separated from the cost of proxying. Behavioral staleness and graph closure time are reported as component quantities. Resource overhead is mean CPU utilisation and peak memory.

5.5 Statistical Treatment and Reproducibility

Each configuration runs five times under distinct seeds governing benign workload generation, adversarial timing and model initialisation, with results reported as mean and standard deviation. Comparisons against each baseline use the Wilcoxon signed-rank test on paired per-run results, a non-parametric procedure suited to this sample size and to distributions that cannot be assumed normal, with effect sizes reported alongside significance so that a detectable but negligible difference is not presented as substantive. Five runs is a modest number and bounds the strength of any inference drawn; it is chosen for feasibility within the project period and stated as such rather than presented as sufficient.

The released artifact comprises the container composition with pinned digests, the workload and adversarial scripts, the seeds, the policies, the feature extraction and modeling code, the generated corpus, and a single command reproducing the full sequence.

5.6 Result Instruments

Tables VIII and IX are the instruments into which measurements will be entered. Sensitivity is examined by sweeping the three thresholds and the weight simplex on a fixed grid and reporting how F1 and false-positive rate vary across it, which establishes whether any operating point eventually reported occupies a stable region or is an artefact of one parameter choice.

TABLE VIII — PLANNED EVALUATION METRICS: DETECTION

Metric	Definition	How obtained	Status
Precision	TP / (TP + FP) over requests receiving step-up or deny	Verdict log scored against injection labels	Pending execution
Recall	TP / (TP + FN) over adversarial requests	Verdict log scored against injection labels	Pending execution
F1 score	Harmonic mean of precision and recall	Derived from the two above	Pending execution
False positive rate	FP / total benign requests	Verdict log over the benign partition	Pending execution
PR-AUC	Area under the precision–recall curve	Curve traced by sweeping τ_2	Pending execution
Per-category recall	Recall computed separately for S1–S2, S3–S4, S5, S6	Labels carry scenario identity	Pending execution
B0 false positives	Benign-administrator requests denied or challenged	Scenario B0 verdicts	Pending execution
Adversarial share	Injected requests as a proportion of total volume	Recorded at generation time	Pending execution
Session window W	Duration of the behavioral window	Fixed configuration value	Pending execution

Experimental execution is future work. The metric definitions and measurement procedures above are fixed; the values they will take are obtained only by running the program of this section, and each configuration in Section 5.3 is measured against every row.

TABLE IX — PLANNED EVALUATION METRICS: LATENCY AND OVERHEAD

Metric	Definition	Condition	Status
Authorization latency	Wall-clock time added to the request path, at p50, p95 and p99	Full system vs broker-only	Pending execution
Graph closure time	Time to compute both reachability closures for one request, p95	Full system only	Pending execution
Graph rebuild time	Time to rebuild the graph after a permitted mutating operation, p95	Full system only	Pending execution
Behavioral staleness	Age of the anomaly score at the moment of decision, p95	Full system only	Pending execution
Policy evaluation time	Time spent inside the policy engine, p95	Full system vs broker-only	Pending execution
Mean CPU utilization	Average across all containers during a run	Full system vs broker-only	Pending execution
Peak memory	Maximum resident set across all containers	Full system vs broker-only	Pending execution
Sustained throughput	Requests per second at which latency p95 remains stable	Full system vs broker-only	Pending execution

Measuring against a broker-only condition matters because an inline enforcement point is deployable only if the latency it adds is tolerable. A design that detects well and doubles request latency has not solved the problem it set out to solve, and a single combined figure would conceal which component was responsible. Experimental execution is future work for this table as for Table VIII.

5.7 Threats to the Validity of This Design

The corpus is generated by the same authors who designed the detector, and the regularity of the benign workload directly determines how difficult detection is. A workload more regular than real administrative activity would inflate every figure the program produces. Two measures constrain this without eliminating it: the benign workload parameters are fixed and documented before any detector tuning takes place, and scenario B0 is constructed specifically to be adversarially hard, containing legitimate administrative work that exercises the same operations as S1. Neither measure substitutes for evaluation on observed enterprise telemetry, which remains the appropriate next step and is not attempted here.

The emulator approximates rather than reproduces the IAM semantics of a production provider, and its performance characteristics differ from a real service. Any latency or throughput figure the program yields describes the testbed. Treating it as a prediction of production overhead would not be warranted, and Section 7.4 records this as a boundary on what the evaluation can establish.

6. CASE STUDY: ESCALATION VIA A COMPROMISED DEPLOYMENT CREDENTIAL

The following traces the specified decision logic against a scenario. Every statement in this section is architectural behavior or analytical reasoning derived from Sections III and IV. None of it is a measurement, and none of it should be read as evidence that the system performs as described; that question is what Section 5 is designed to settle.

6.1 Threat Scenario

An adversary obtains a long-lived access key belonging to a deployment principal with modest permissions, corresponding to scenario S5 of Table VII. The key is valid, so every request is authenticated and, under static entitlement, permitted up to the principal's configured limits. The objective is administrative authority. No software vulnerability is exploited, and the operations invoked are ones legitimate tooling invokes daily [6], [8].

6.2 Detection Sequence

Figure 3 traces the sequence. Credential compromise precedes t_0 and is not itself observable to this design. At t_0 the adversary performs identity introspection, and at t_1 enumerates attached policies and assumable roles. Neither operation propagates permissions, so G(r) is zero for both, and the behavioral term carries the decision alone. What the engine registers is novelty rather than malice: action families absent from this principal's baseline, activity outside its established hours, an origin with no history. C(s(r)) rises with the unseen-origin and off-hours indicators. The composite crosses τ_1, and because the target of the enumeration is an identity object and therefore high sensitivity, Table V returns step-up. The session is marked provisional.

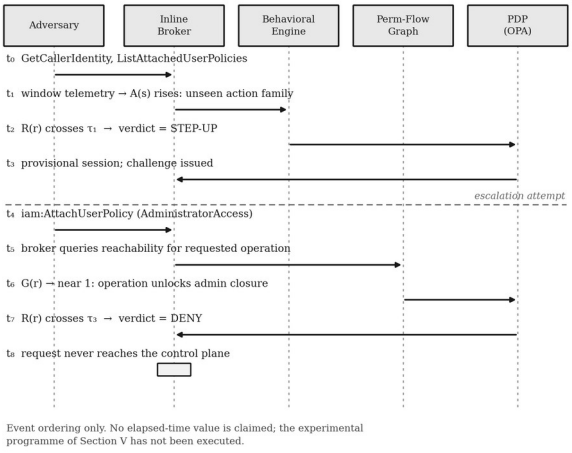


Figure 3: Detection sequence for the case-study scenario. Event ordering only; no elapsed-time value is claimed. The behavioral term raises risk during enumeration, and the permission-flow term, evaluated for the specific operation requested, carries the composite past the deny threshold at the escalation attempt.

At t_4 the adversary attempts the escalation itself, requesting attachment of an administrative managed policy. Here the two terms separate informatively. The behavioral term treats this as one further unusual action, contributing no more than the enumeration did. The permission-flow service, queried at t_5 for this specific operation against the configuration in force, returns at t_6 a gain approaching unity: succeeding would extend the principal's closure to administrative authority, so the numerator of G(r) approaches its denominator. The composite crosses τ_3. The target is an identity object, so the matrix returns deny.

6.3 Enforcement Outcome

Because enforcement is inline, the operation does not reach the control plane. No policy is attached and nothing requires undoing. The verdict, the composite, its three contributing terms and the governing rule are written to the decision log, and revocation propagates to the identity provider, terminating the provisional session opened at t_3. This is the property that separates the design from detection systems: the response is refusal rather than notification followed by cleanup.

6.4 Cross-Plane Variant

Scenario S6 exercises the property that motivates the hybrid framing, and it is worth tracing because a single-plane scenario would not demonstrate it. A directory credential is compromised on the on-premises side and used to authenticate against the identity provider. That authentication is legitimate and, considered alone, unremarkable: the principal has logged in before. The adversary then exchanges the resulting assertion for cloud session credentials and, within the same window, begins enumerating cloud entitlements.

Neither plane monitored alone would register anything. The on-premises directory sees one successful logon. The cloud control plane sees a federated principal doing what federated principals do. What the broker sees, because it normalizes both streams into one identity-event schema keyed on the federated principal, is a window containing an on-premises authentication followed by cloud enumeration from an origin with no history for that principal — the cross-plane contextual feature of Table II together with the enumeration-to-mutation order feature. The composite rises on the contextual and order terms before any permission-propagating operation is requested, and the subsequent delegation attempt is then evaluated against an already-elevated score.

The claim here is narrow: the architecture makes the cross-plane sequence visible as one behavioral unit, which per-plane monitoring does not. Whether that visibility yields a detection advantage in practice is what the S6 rows of Table VIII will show once the program is run.

6.5 Comparative Reasoning

Traced against the same scenario, the four baselines behave differently. Under B1 the sequence completes, since every call falls within configured permissions and nothing is re-evaluated. Under B2 the attachment matches a rule and an alert fires, but the attachment has already taken effect and the outcome depends on how quickly an analyst responds. Under B3 the enumeration raises risk, yet the escalation itself carries no more weight than any other unusual action, so whether τ_3 is crossed depends on accumulated behavioral deviation alone. Under B4 the escalation is caught, while the enumeration phase passes unnoticed and a legitimate administrator performing the same attachment is treated identically — the failure that scenario B0 exists to expose.

This reasoning explains why the architecture is expected to behave as claimed. It does not establish that it does. The comparison assumes the models behave as designed on data that has not yet been generated, and its conclusions are conditional on the outcome of Section 5.

7. SECURITY ANALYSIS

7.1 Security Properties and Their Conditions

Four properties are claimed by design, each holding under stated conditions rather than unconditionally. Continuous verification holds in that authorization is recomputed per request rather than granted once per session, subject to the behavioral term's staleness bound. Pre-execution enforcement holds provided the broker is the sole path to the control plane, which Section 3.8 identifies as an assumption external to this design. Proportionality holds in that the response scales with both deviation and the consequence of the specific operation. Accountability holds in that every verdict is recorded with its inputs, its governing rule and a marker where the decision was degraded. These are properties of the specification. Whether an implementation exhibits them is what Section 5 is designed to test.

7.2 Threat Model

The adversary holds valid credentials for a low-privilege principal, can issue arbitrary API calls within that principal's entitlement, knows the provider's permission model, and can pace activity to evade detection. The adversary does not control the broker, the analytics services or the policy decision point, cannot reach the control plane except through the broker, cannot suppress telemetry without generating a record, and does not hold administrative credentials initially. The third of these assumptions is the load-bearing one and depends on controls outside this design. Table X states coverage, including what the architecture does not address.

TABLE X — ATTACK COVERAGE ASSESSMENT

Attack behavior	Coverage	Basis and residual risk
Permission enumeration	Detected	Novel action families and volume relative to baseline; behavioral term only, as G(r) = 0
Delegation escalation (S1, S2)	Detected	G(r) approaches unity for the requested operation
Credential takeover (S3, S4)	Detected	Operation extends closure to another principal's authority
Multi-step chain (S5)	Detected	Both terms contribute at different stages of the sequence
Slow, low-volume escalation	Partial	Behavioral term may stay within tolerance; the permission-flow term still fires on the mutating operation itself
Escalation by a legitimately privileged insider	Partial	Behavior matches baseline; only the permission-flow and contextual terms contribute
Baseline poisoning during training	Not detected	Adversary activity present during profiling is absorbed into the baseline
Gradual evasion by paced activity	Partial	Baseline adapts to the adversary; the permission-flow term is unaffected by pacing
In-instance activity below the API layer	Not detected	Outside control-plane telemetry
Data-plane access without permission change	Not detected	G(r) = 0; only the behavioral term can contribute
Broker bypass or direct control-plane access	Not detected	Violates the assumption on which every property rests
Compromise of broker, analytics or PDP	Not detected	Adversary is outside the stated threat model

7.3 Limitations

The broker is a single point of both enforcement and failure. If a request path exists that does not traverse it, the design provides nothing. Section 3.8 specifies fail-closed behavior for the failure case; the bypass case requires provider-side controls that the emulated testbed does not model and about which this work offers no evidence.

The corpus is synthetic and self-generated, so its benign workload encodes assumptions about normal activity that may be more regular than real behavior. The emulator approximates production IAM semantics and differs in performance characteristics, which confines every latency figure to the testbed. The permission-flow gain is normalized against the environment's permission universe and so is not comparable across deployments of different size, meaning thresholds require recalibration when the environment changes. Coverage extends to two acquisition categories of five.

Behavioral weaknesses persist. Baseline poisoning is excluded by the experimental partitioning but not by the deployed design. Gradual evasion remains available against the behavioral term, mitigated only by the permission-flow term's indifference to pacing. Graph rebuild on every identity-mutating operation bounds throughput, and where that bound falls for realistic configuration sizes is measured in Section 5.6 but not resolved by it. Five runs per configuration limits the strength of any statistical inference.

7.4 Security–Continuity Trade-off

Lower thresholds intercept escalation earlier and interrupt legitimate work more often; higher thresholds reverse both. This design does not resolve the tension, and no threshold setting is claimed optimal in general. The sensitivity analysis of Section 5.6 exists to characterize the trade-off across the parameter space rather than to conceal it behind one reported operating point. The same reasoning governs the decision to cap low-sensitivity resources at step-up: where an operation cannot alter the permission model, the continuity cost of refusing it outweighs the security benefit.

8. CONCLUSION AND FUTURE WORK

This paper addressed privilege escalation in hybrid cloud environments, where an adversary holding a compromised low-privilege credential reaches administrative authority through a sequence of individually legitimate control-plane

operations. The prevailing controls each miss part of the problem: static entitlement cannot respond to a credential stolen after configuration, offline reachability analysis proves a path exists without observing anyone using it, and signature matching reports once the operation has completed.

The proposal places an inline broker on the request path and combines two complementary signals evaluated there. A behavioral engine scores how unusual a principal's recent activity is against its own baseline, asynchronously and with a bounded staleness. A permission-flow service computes synchronously, for the specific operation requested, the share of currently unreachable permissions that operation would unlock. Neither suffices alone: behavior cannot say what an operation would enable, and reachability cannot distinguish an attacker from an administrator. Their convex combination, evaluated against thresholds and resource sensitivity, yields a verdict returned before execution.

The contribution is confined to that integration. Zero Trust is established, unsupervised anomaly detection is established, and the reachability formulation is due to prior work [6], [10]. What is offered here is their coupling at an enforcement point, a decision function specified completely enough to be criticised, and an evaluation design executable at no cost on open-source components. Where the trust algorithm is usually named as a component rather than published as a procedure, writing it down is the step this paper takes.

The evaluation program of Section 5 has been designed and not executed. This paper therefore contributes an architecture, a mechanism and a methodology, and no empirical finding. Every result cell is marked as pending execution, and experimental execution is future work.

Five directions follow. Executing the specified program is the immediate one. Coverage should then extend from two acquisition categories to the five identified in prior work [6], which requires further adversarial scenarios but no architectural change. Validating the testbed against a real provider account matters more than it might appear: emulator IAM semantics only approximate production behavior, and the bypass-prevention controls on which every security property depends can only be tested where they exist. Adversarial robustness deserves empirical study, particularly the rate at which paced activity can be absorbed into a baseline. Finally, graph rebuild cost bounds throughput, and locating that bound for realistic configuration sizes is a prerequisite to any claim that the design is deployable at scale.

AUTHOR CONTRIBUTIONS

Author	Main responsibility	Evidence
Aalok Bhandari	Inline broker and emulated cloud plane integration	Source code, container composition, request and verdict logs
Ashal Pandey	Behavioral analytics, permission-flow graph service, risk computation	Corpus generation scripts, model code, feature pipeline output
Famous Dhungana	Policy engine rules, identity provider configuration, testing and documentation	Policy files, federation configuration, test results and run records

All three authors contributed to the threat model, the case-study analysis and the manuscript. The evaluation program described in Section 5 had not been executed at the time of writing.

REFERENCES

[1] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST Special Publication 800-207, National Institute of Standards and Technology, Gaithersburg, MD, 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," NIST Special Publication 800-207A, National Institute of Standards and Technology, Sep. 2023. doi: 10.6028/NIST.SP.800-207A.

[3] 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.

[4] L. Ferretti, F. Magnanini, M. Andreolini, and M. Colajanni, "Survivable zero trust for cloud computing environments," Computers & Security, vol. 110, art. 102419, Nov. 2021. doi: 10.1016/j.cose.2021.102419.

[5] MITRE Corporation, "ATT&CK Matrix for Enterprise: Cloud." [Online]. Available: https://attack.mitre.org/matrices/enterprise/cloud/

[6] Y. Hu, W. Wang, S. Khurshid, and M. Tiwari, "TAC: Hybrid IAM Privilege Escalation Detection," arXiv:2304.14540, version 8, Mar. 2026. [Online]. Available: https://arxiv.org/abs/2304.14540

[7] I. Shevrin and O. Margalit, "Detecting Multi-Step IAM Attacks in AWS Environments via Model Checking," in Proc. 32nd USENIX Security Symposium, Anaheim, CA, Aug. 2023, pp. 6025–6042.

[8] S. Gietzen, "AWS IAM Privilege Escalation — Methods and Mitigation," Rhino Security Labs, 2018. [Online]. Available: https://rhinosecuritylabs.com/aws/aws-privilege-escalation-methods-mitigation/

[9] Rhino Security Labs, "Pacu: The Open Source AWS Exploitation Framework." [Online]. Available: https://github.com/RhinoSecurityLabs/pacu

[10] E. Steringer, "Principal Mapper (PMapper)," NCC Group. [Online]. Available: https://github.com/nccgroup/PMapper

[11] Splunk Threat Research Team, "AWS IAM Privilege Escalation," Splunk Security Content, 2021. [Online]. Available: https://research.splunk.com/stories/aws_iam_privilege_escalation/

[12] S. Piper, "Public dataset of CloudTrail logs from flaws.cloud," Summit Route, Oct. 2020. [Online]. Available: https://summitroute.com/blog/2020/10/09/public_dataset_of_cloudtrail_logs_from_flaws_cloud/

[13] J. Glasser and B. Lindauer, "Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data," in Proc. 2013 IEEE Security and Privacy Workshops, San Francisco, CA, 2013, pp. 98–104. doi: 10.1109/SPW.2013.37.

[14] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Analyzing Data Granularity Levels for Insider Threat Detection Using Machine Learning," IEEE Transactions on Network and Service Management, vol. 17, no. 1, pp. 30–44, Mar. 2020.

[15] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. 8th IEEE International Conference on Data Mining (ICDM), 2008, pp. 413–422.

[16] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001.

[17] P. A. Grassi, J. L. Fenton, E. M. Newton, et al., "Digital Identity Guidelines: Authentication and Lifecycle Management," NIST Special Publication 800-63B, National Institute of Standards and Technology, 2017. doi: 10.6028/NIST.SP.800-63b.

[18] Open Policy Agent Project, "Open Policy Agent Documentation," Cloud Native Computing Foundation. [Online]. Available: https://openpolicyagent.org/docs/

[19] LocalStack, "LocalStack Documentation." [Online]. Available: https://docs.localstack.cloud/

[20] Keycloak Project, "Keycloak Documentation," Red Hat. [Online]. Available: https://www.keycloak.org/documentation

[21] Amazon Web Services, "Policy Evaluation Logic," AWS IAM User Guide. [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html