International Journal of Cyber Threat Intelligence and Secure Networking (IJCTISN)
eISSN: 3087-4297
Volume03 Issue10 | Page No: - 08-21

Received: 28 Sep 2026, Revised: 14 Oct. 2026, Accepted: 25 October 2026, Published: 30 October 2026

DOI: https://doi.org/10.55640/ijctisn-v03i10-02

Non-Human Identity as the Primary Attack Surface: A Governance Architecture for AI Agent Credentials in Regulated Multi-Cloud Enterprises

Parijat Sharma
Independent Researcher, USA

ABSTRACT: Non-human identities now outnumber human identities in many enterprise environments and frequently hold persistent, poorly reviewed privilege. AI agents amplify the exposure, because an agent selects tools, retrieves data, and executes actions without continuous human involvement, which means an identity can be technically valid while the action it chooses is outside its original business purpose. This paper proposes a governance architecture that gives every service, workload, and agent a named owner, a verifiable identity, constrained authorization, short-lived credentials, continuous behavioral monitoring, and rapid revocation. The architecture treats a non-human identity as a governed security object carrying a lifecycle, a dependency graph, a risk tier, an action boundary, and an evidence record. For AI agents it separates the identity of the agent from the authority of each tool call, so that authenticating as an agent does not by itself authorize what the agent proposes to do. Governance is unified across clouds while provider-native enforcement is preserved rather than replaced, which lets an organization answer both who a workload is and what it may do at this moment. The architecture is proposed and synthesized from production control experience; it is not presented as a deployed enterprise control plane, and the paper states which quantities remain unmeasured.

KEYWORDS: Non-human identity; machine identity; AI agent identity; workload identity; service account; cloud IAM; multi-cloud security; identity governance; secrets management; credential rotation; workload federation; zero trust; least privilege; just-in-time access; continuous authorization; agentic AI security; entitlement graph; credential revocation; regulated enterprise.

1. Introduction

The population of identities inside an enterprise has inverted, and most identity programmes have not.

Identity governance as practised was designed around a person. It assumes an employee with a manager, a job role, an employment status, an interactive login, and a periodic access review someone can be held to. The controls that grew from those assumptions are genuinely mature: multi-factor authentication, conditional access, joiner-mover-leaver processes driven from a human resources system, and certification campaigns with named approvers.

None of those assumptions hold for the identity of a container. A workload has no manager. It does not resign. It authenticates without a human present, often thousands of times an hour, and it may be created automatically by a deployment pipeline that no one reviews in the way a new hire is reviewed. It may exist for four seconds or four years. And the mechanism that governs the human population, which is that leaving the company terminates access, has no analogue: a service account's credential does not expire when the engineer who created it changes teams.

The result is accumulation. Machine identities are created as part of applications, scripts, cloud resources, deployment pipelines, integrations, and emergency fixes, and they persist after the project that created them has changed because they were never bound to a lifecycle process in the first place. Ownership decays, documentation decays faster, and teams become reluctant to disable anything because the business dependency is unknown. The reluctance is rational. Disabling an unknown dependency causes an outage, and the person who disables it owns that outage, whereas leaving it in place distributes the risk across everyone and attributes it to no one.

What an attacker gains from this is a trusted execution

path. A compromised machine identity may bypass interactive multi-factor authentication, sit outside user behavior analytics that were tuned for humans, and require no human approval, and depending on its privilege it can retrieve data, create infrastructure, call internal APIs, alter configuration, move laterally, and establish persistence while appearing to be a legitimate workload doing legitimate work. The empirical record supports the concern without needing to be overstated: credential abuse remains the most pervasive technique across full breach chains in the current reporting even as vulnerability exploitation has overtaken it as the leading initial access vector [2].

The claim of this paper is therefore not that every breach begins with a machine identity. That claim is false and would be easy to disprove. The claim is narrower and harder to dismiss: unmanaged non-human access has become one of the largest and least-governed attack surfaces in the enterprise, and it is the surface where the gap between the number of identities and the number of governed identities is widest.

AI agents make this materially worse in a way that is qualitatively new. Every previous class of machine identity executed a fixed program. A service account did what its code did, and if you knew the code you knew the reachable action set. An agent plans. It selects tools at runtime based on context, and its reachable action set is bounded by its permissions rather than by its instructions. This introduces a failure mode with no prior analogue: the identity is valid, the credential is legitimate, the authentication is genuine, and the action is nonetheless unsafe or outside the original business purpose. Every control built to answer who is this fails to detect it, because the answer to who is this is correct.

This paper proposes a governance architecture addressing both the accumulated problem and the new one. Its contributions are four.

First, it treats a non-human identity as a governed security object with a lifecycle, dependency graph, risk tier, action boundary, and evidence record, rather than as an account in a directory.

Second, and centrally, it separates the identity of an agent from the authority of each tool call, so that authenticating as an agent authorizes nothing by itself.

Third, it specifies a cross-cloud entitlement and access-path graph that resolves inherited, delegated, and agent-to-tool authorization into a single answerable question, while preserving provider-native enforcement rather than replacing it.

Fourth, it specifies revocation as a sequence with an explicit and quantified residual-exposure window, on the grounds that revocation is routinely described as instantaneous and routinely is not.

Note on scope, evidence, and data handling

This paper is a reference architecture. The distinction between what is running and what is designed is maintained throughout.

The underlying practical work developed progressively through the author's security architecture responsibilities from approximately 2024 through July 2026, with the agent-specific governance model consolidated during 2026.

What is implemented. The author has reviewed and designed controls for human and non-human access across enterprise cloud and SaaS environments, covering service accounts, workload roles, application identities, enterprise applications, privileged cloud identities, API credentials, emergency access, and machine-to-machine integrations. This work included identity governance reviews, least-privilege analysis, application-consent governance, cloud account controls, access-path analysis, credential-risk identification, and architecture recommendations across more than one cloud platform.

What is proposed. The complete lifecycle architecture presented here, and in particular the unified governance layer spanning autonomous AI-agent identities across multiple clouds, is a proposed reference architecture synthesized from those controls and the gaps observed between them. No claim is made that a single enterprise-wide platform currently governs every AI agent, workload identity, service account, and credential through one production control plane.

What is not claimed. There is no verified enterprise-wide count of non-human identities or AI agents, no measured average credential lifetime, no universal revocation-time benchmark, and no defensible number of incidents prevented by this architecture. These are not estimated or modelled anywhere in the paper. Section 8 states what a production implementation would have to measure.

Data handling. Two practitioner observations appear in Section 3.4. Both are deliberately generalized to order of magnitude and are described by function. No employer, product, vendor, tenant, account count, control gap, or internal control status is disclosed, and the observations are used to illustrate a class of governance failure rather than to characterize any organization.

2. Background and related work

2.1 The machine identity problem as documented

The practitioner literature has converged on the shape of the problem. The OWASP Non-Human Identities Top 10 [1] enumerates the recurring failures, and the ones that matter most for this architecture are improper

offboarding, secret leakage, over-privileged non-human identities, and long-lived secrets. The pattern across all four is the same: the failure is not that the identity was created wrongly, but that nothing owns the question of whether it should still exist with the privilege it has.

Secret leakage has an additional property that makes it structurally different from human credential exposure. A leaked password is invalidated by a password change and its exposure window closes. A leaked machine credential is frequently still valid years later, because nothing forces rotation and no person's departure triggers review. Published scanning of public repositories continues to find very large volumes of newly exposed secrets annually, with a substantial fraction of historically exposed secrets remaining valid [1].

2.2 Workload identity and attestation

The strongest available answer to the credential problem is to stop having credentials. Workload identity frameworks issue short-lived identity documents on the basis of attested runtime properties rather than a stored secret, so the workload proves what it is and receives a credential valid for a bounded period. NIST SP 800-207A [4] names application identity infrastructure of this kind as a component of zero trust for cloud-native applications in multi-cloud environments, and its identity-based segmentation model is the one this architecture adopts.

Attestation is the mechanism that carries the weight. It binds the identity to the workload's actual execution context, so that the issuing system mints identity material only when the attested runtime matches policy. That property is what allows the architecture to answer who is this workload with something better than possession of a secret, and it is why the architecture prefers federation and provider-managed workload identity over static keys wherever the platform permits.

The limitation is coverage. Attestation-based workload identity is well supported in orchestrated compute and increasingly in cloud-managed services, and it is poorly supported or absent in SaaS platforms, legacy applications, appliances, and much of the integration estate. Any architecture claiming to unify machine identity governance has to survive that unevenness rather than assume it away.

2.3 Agent authorization

The 2026 literature on agent authorization has produced the mechanism this architecture needs for its central contribution, arriving from several directions at once.

Tallam [5] examines authorization propagation in multi-agent systems and makes the observation that anchors this paper's design: the naive approach in which an agent inherits the full authority of the human it acts for violates least privilege, and no widely deployed authorization system yet provides a complete solution for autonomous agent-to-agent delegation. Uchibeke [8] argues for deterministic pre-action authorization evaluated before the tool call. Zhu and Wang [9] define intent-tool mismatch as an access-control failure and bound privilege to the request. Kodathala [10] places the authorization decision off the agent host so a compromised agent cannot authorize itself. Patil [7] addresses delegation chains with intent verification, and Yu et al. [6] specify a delegation protocol for secret use in agentic systems. Avirneni [11], [12] develops the identity control plane as a unifying layer and applies intent-aware authorization to pipeline identities. Rajendran et al. [13] treat identity federation as the basis for zero trust in microservices. Sharma [14] takes the runtime enforcement problem down to a hardware-rooted foundation. Lazer et al. [15] survey the field.

What is missing across this body of work is the enterprise governance frame. These papers specify how an agent's action should be authorized. They do not address how an organization enumerates every agent it has, assigns each one an accountable human owner, determines what each can reach across three cloud providers with different inheritance models, proves the answer to an auditor, and revokes it inside a defined time when something goes wrong. That is a governance problem rather than a protocol problem, and it is what this paper contributes.

2.4 The gap

Existing work covers the failure catalogue [1], the identity primitive [4], and the per-action authorization mechanism [5]-[12]. No published work joins them into a lifecycle architecture that treats agents as one class within a governed non-human identity estate, preserves provider-native enforcement across heterogeneous clouds, and produces the evidence a regulated enterprise must present.

3. The problem in detail

3.1 How the estate accumulates

Machine identities enter the estate through at least seven routes, and only one of them resembles a governed request. They are created as part of application deployment, by infrastructure-as-code, by pipeline automation, by SaaS integration configuration, by data platform connections, by robotic process automation, and by emergency remediation during incidents. The last of these is worth naming because it produces the worst instances: created under time pressure, granted broad privilege to guarantee the fix works, and never revisited once the incident closes.

The failures that follow are consistent across environments: long-lived access keys, embedded secrets, shared service accounts, wildcard permissions, unused credentials, inconsistent naming that defeats inventory, weak or absent rotation, missing owners, absent expiration, duplicate identities, and no monitoring of actual use as distinct from monitoring of existence.

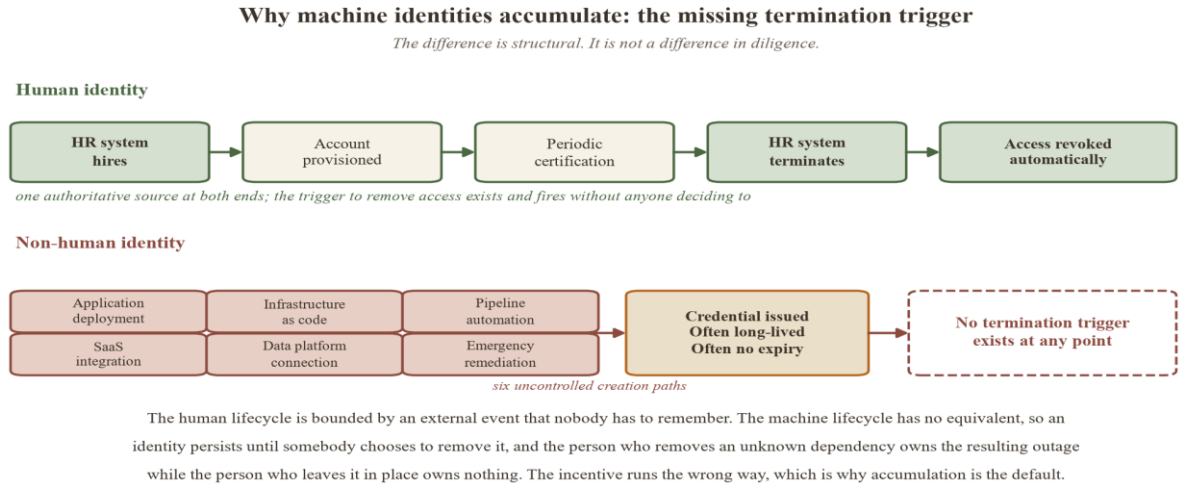


Figure 1. Human and non-human identity lifecycles compared. The human lifecycle has an authoritative external trigger at both ends, which is what makes deprovisioning reliable. The non-human lifecycle has multiple uncontrolled creation paths and, critically, no external termination trigger at all, which is the structural reason machine identities accumulate rather than a failure of diligence.

3.2 Why human-centered governance does not transfer

Table 1 sets the two governance models against each other. The point of the comparison is not that machine identity governance is harder, which is obvious, but that the specific control primitives do not have machine equivalents, so the gap cannot be closed by extending existing tooling.

Governance primitive	Human identity	Non-human identity	Consequence
Authoritative source	HR system; joiner-mover-leaver events	None. Creation is distributed across pipelines, code, consoles and integrations	No trigger exists for deprovisioning
Accountability	The person, plus their manager	Frequently unassigned, or assigned to a team that has since reorganized	Nobody can be asked to approve removal
Authentication assurance	MFA, conditional access, device posture	Possession of a key or token; sometimes attestation	Theft of the credential is equivalent to being the workload
Termination	Employment ends; access is revoked	Credential remains valid indefinitely	Exposure has no natural end

Access review	Periodic certification with a named approver	Existence check rather than privilege justification	Accumulated privilege is never removed
Behavior baseline	User behavior analytics tuned to human rhythms	Continuous, machine-speed, high-volume; human baselines do not apply	Anomalies are invisible to human-tuned analytics
Delegation	Rare and explicit	Routine, chained, and often implicit through service-to-service trust	Attribution across the chain is lost
Action set	Bounded by the interface the person uses	For agents, bounded only by permission, selected at runtime	A valid identity may take an unforeseen action

Table 1. Why human identity governance does not transfer to non-human identities. The final row is the one with no historical analogue and is the reason the architecture separates identity from per-action authority.

3.3 The nine classes and how they differ

The architecture governs nine classes of non-human identity. They differ in how the identity is proven, how credentials are obtained, and where trust material lives, and those three differences determine which controls are available for each.

Class	**How identity is proven**	**Where credential or trust lives**	**Governing constraint**
Service accounts	Password, key, certificate, token, or platform-managed credential	Directory, application config, vault, host, database, or SaaS platform	Long-lived by default; the hardest class to make ephemeral
Cloud workload identities	Provider-issued token bound to resource identity, metadata, certificate, or federation	Cloud control plane; ideally no static secret	Strongest primitive available; provider-specific semantics
CI/CD and automation identities	Federated token, signed workload assertion, API token, certificate, or stored secret	Pipeline platform, secrets manager, federation config, or runner	Broad blast radius; often holds deployment privilege into production
API clients and integrations	OAuth client credential, signed assertion, API key, mTLS certificate, or delegated token	Identity provider, API gateway, application vault, integration platform	Frequently crosses an organizational boundary
Containers and	Workload service account, projected token, platform	Orchestrator and cloud identity provider;	Population churns faster

orchestrated workloads	attestation, or federated identity	credentials should be ephemeral	than any review cycle
Data and analytics identities	Cloud role, database credential, certificate, token, or managed service identity	Data platform, vault, cloud IAM, or database	Reaches the most sensitive data by design
Robotic process automation	Bot account, certificate, vault-injected secret, or delegated identity	RPA control plane, enterprise directory, credential vault	Interacts through user-facing interfaces; often holds human-equivalent privilege
AI agents	Agent identity plus short-lived tool-specific authorization, delegated user context, workload attestation, or signed capability token	Agent platform, identity provider, gateway, policy engine, tool broker	Action set selected at runtime; identity validity does not imply action validity
Emergency and break-glass	Strongly protected or hardware-backed credential under dual control	Isolated vault or secure recovery process	Must survive failure of the identity system that governs everything else

Table 2. The nine governed classes. The AI agent row differs from every row above it in one respect that drives the architecture: for the other eight classes, knowing the identity substantially bounds the action set. For agents it does not.

3.4 Two observations from governance review

Two practitioner observations motivate a specific architectural requirement. Both are generalized to order of magnitude and described by function, with no organization, platform, or product identified.

The first concerns application consent. A single identity-governance review of an enterprise application estate surfaced dozens of historical administrative consent grants that had never been subject to governance review. Each represented an application holding delegated or application-level permission into enterprise data, granted at some point in the past by someone with authority to grant it, and thereafter unexamined. None was anomalous individually. Collectively they constituted an unreviewed permission surface that no existing process owned.

The second concerns the reliability of control-state reporting. In a cloud account control programme covering several hundred accounts, a substantial share of accounts had in fact been remediated while the security reporting continued to show the control as outstanding. The gap was in the reporting path rather than in the control itself.

The second observation is the more important of the two, and its lesson is uncomfortable. If the reporting layer can be wrong about the state of a simple, binary, well-understood control across a known account population, then a governance architecture cannot treat any single reporting source as authoritative for the far harder question of what an agent can currently reach. The requirement this produces is independent reconciliation: control state must be verified against the underlying platform rather than accepted from the layer that summarizes it. That requirement is carried into Section 4.7 and into the metrics of Section 8.

4. The governance architecture

The architecture governs a non-human identity through nine ordered stages. The stages are presented in order, though the lifecycle is not strictly linear, because Stage 8 returns to Stage 4 whenever review finds excess privilege.

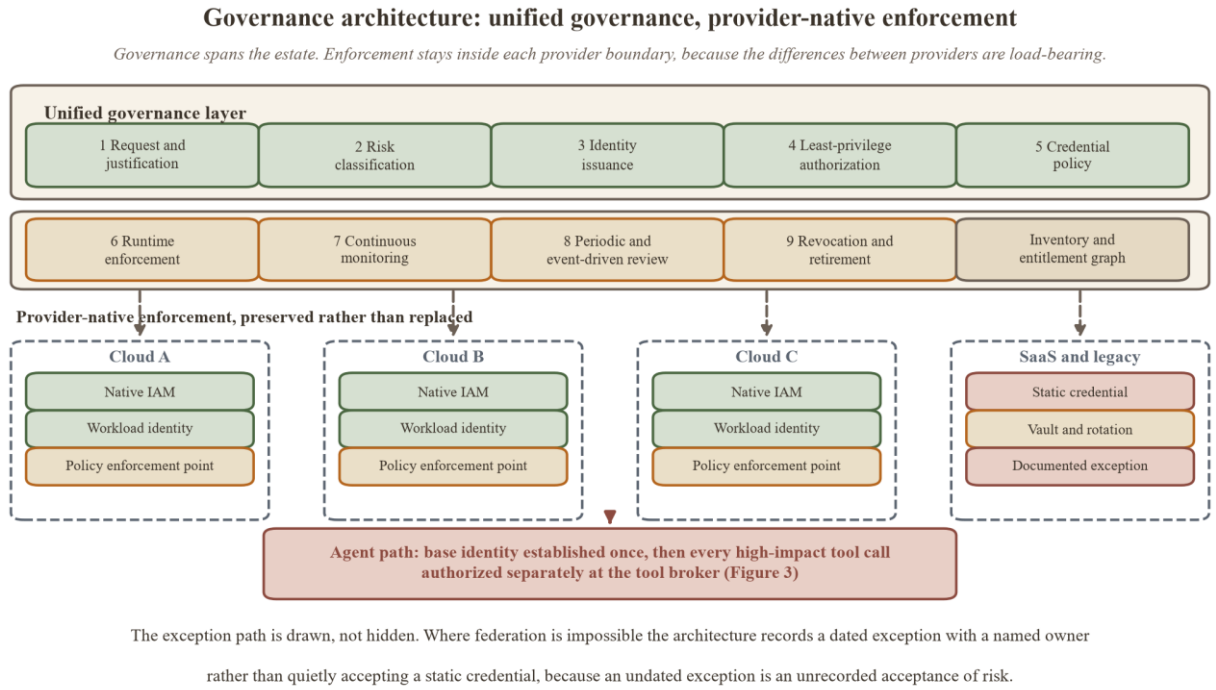


Figure 2. The governance architecture across the identity lifecycle. The unified governance layer spans the estate while enforcement remains provider-native, shown by the enforcement points sitting inside each cloud boundary rather than above it. The agent action-control path, drawn separately from the base identity path, is the architecture's central structural claim.

4.1 Request and business justification

A service owner requests a non-human identity and states its purpose, data classification, environments, expected lifetime, dependencies, and accountable human owner.

The accountable human owner is the load-bearing field and it must be a named person rather than a team alias. A team alias survives reorganization by pointing at nobody in particular, which is the observed mechanism by which owned identities silently become unowned.

Expected lifetime is the second field that matters, because it converts the default from permanent to temporary. An identity requested for a ninety-day migration should expire at ninety days and require justification to continue, and this single change removes a large share of the accumulation problem at the point of creation rather than through review years later.

4.2 Risk classification

The identity is classified by privilege, reachable assets, regulatory impact, autonomy, external connectivity, and potential blast radius.

Autonomy is included as a dimension because it is the property that distinguishes an agent from a script with the same permissions. Two identities with identical entitlements present different risk if one executes fixed logic and the other selects its actions at runtime, and a classification scheme that scores only on permission cannot see the difference.

4.3 Identity issuance

A provider-native workload identity or federated identity is created. Static credentials are permitted only where federation or managed identity is not technically possible, and that determination is recorded as an exception with a named owner and a review date rather than accepted silently.

The ordering here is a policy statement. Federation is the default and a static secret is a documented failure to achieve the default. Stated the other way round, which is how most estates actually operate, static credentials become the norm and federation becomes a project.

4.4 Least-privilege authorization

Permissions are derived from approved tasks and expressed as narrowly scoped roles, resource policies, tool permissions, and data-level controls. Wildcards and standing administrative access require exception approval.

Deriving permission from tasks rather than from convenience is the difference between an identity that can do its job and an identity that can do its job plus everything adjacent to it. In practice the adjacent set is where an attacker operates, because the intended function is rarely the useful one.

4.5 Credential policy

Credentials are short-lived, automatically rotated, bound to workload context where possible, and stored in an approved secrets system only where persistence is unavoidable.

Sender-constrained and proof-of-possession token mechanisms are preferred over bearer tokens for this reason: a bearer token is a credential that anyone holding it may use, so its theft is equivalent to compromise of the workload, whereas a sender-constrained token requires the holder to demonstrate possession of a key the token is bound to. Standardized mechanisms exist for both certificate-bound access tokens [16] and application-layer proof of possession [17], and the architecture requires one of them for any identity classified above the lowest risk tier.

4.6 Runtime enforcement and agent action control

Authentication context, workload attestation, environment, resource sensitivity, action type, and risk signals are evaluated before access is granted.

For agents, one additional rule applies and it is the architecture's central contribution.

The agent's base identity does not authorize its tool calls. Authenticating as the agent establishes only that the agent is the agent. Each high-impact action is separately evaluated by a policy enforcement point against purpose, scope, data sensitivity, transaction limit, and required human approval, and the evaluation happens before the call rather than being inferred from the agent's stated intent [8], [9].

The reason for separating these is not defence in depth. It is that they answer different questions and the first has no bearing on the second. Identity answers who is calling. Authorization answers whether this specific call, against these specific resources, at this moment, for this stated purpose, is within the boundary that was approved for this agent. An agent whose identity is entirely legitimate can propose an action that is entirely outside its purpose, which is precisely the failure mode that has no analogue in fixed-function machine identity, and no amount of identity assurance detects it.

The policy enforcement point is placed at the tool broker rather than inside the agent, so that a compromised agent cannot self-authorize [10]. Delegation chains preserve the original principal, the agent identity, the authorization scope, the purpose, and the action history at every hop, without which attribution across a multi-agent path is lost [5], [7].

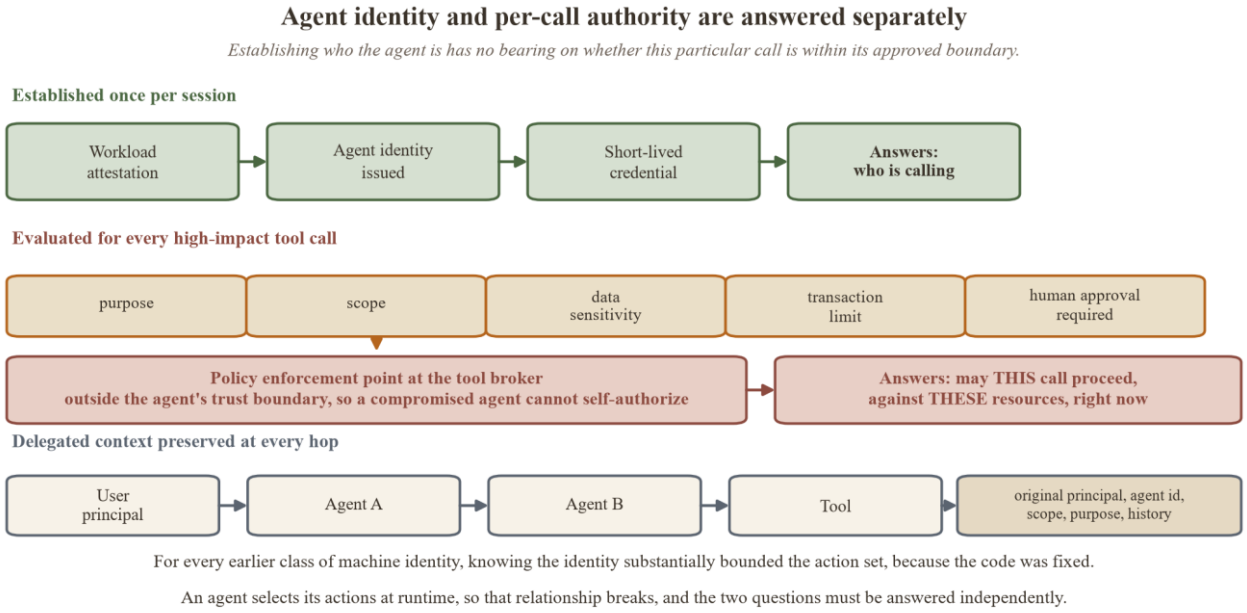


Figure 3. Separation of agent identity from per-call authority. The upper path establishes who the agent is, once per session. The lower path evaluates each proposed tool call independently against the action boundary, and it is the path that catches a valid identity proposing an invalid action. The delegated-context fields carried at each hop are what preserve attribution when one agent invokes another.

4.7 Continuous monitoring

Authentication, token issuance, secret retrieval, privilege use, tool calls, data access, policy denials, and anomalous behavior are logged and correlated.

Policy denials deserve particular attention as a signal. A denial is the system working correctly, so it generates no incident, and consequently denial patterns are rarely examined. But a rising denial rate for one agent is among the earliest available indicators that the agent is attempting actions outside its boundary, whether through compromise, a model or prompt change, or drift between the approved purpose and the deployed

behavior.

The independent-reconciliation requirement from Section 3.4 applies here. Monitoring that consumes only the summarizing layer inherits that layer's errors, so control state is verified against the underlying platform on a defined cycle.

4.8 Periodic and event-driven review

Access is reviewed on a schedule and on change of ownership, application architecture, environment, risk, or usage. Unused and excessive permissions are removed.

Event-driven review is the more effective of the two, because a scheduled review examines the estate at an arbitrary moment while an event-driven review examines an identity at the moment its context changed, which is when the answer is most likely to have changed.

4.9 Revocation and retirement

Revocation is a sequence, not an event, and treating it as an event is the most common error in incident response involving machine identities.

The sequence has seven steps: freeze or disable the identity in the authoritative control plane; deny new token issuance and remove role, group, trust, and resource-policy bindings; invalidate sessions or tokens where the platform supports it; rotate or revoke persistent secrets, certificates, API keys, and signing material; block the workload at the gateway, network, orchestration, or tool layer where tokens cannot be immediately revoked; quarantine the workload and preserve logs; and verify that downstream access is no longer possible, recording the actual revocation time.

Step three carries a qualification that governs the whole design. Many signed tokens remain valid until they expire, irrespective of what happens to the identity that was issued them. Disabling an identity does not necessarily stop a token already in circulation. The residual exposure window is therefore bounded by token lifetime rather than by the speed of the response, which is the single strongest practical argument for short lifetimes: not that short-lived credentials are harder to steal, but that they bound the damage when revocation is slower than everyone assumes.

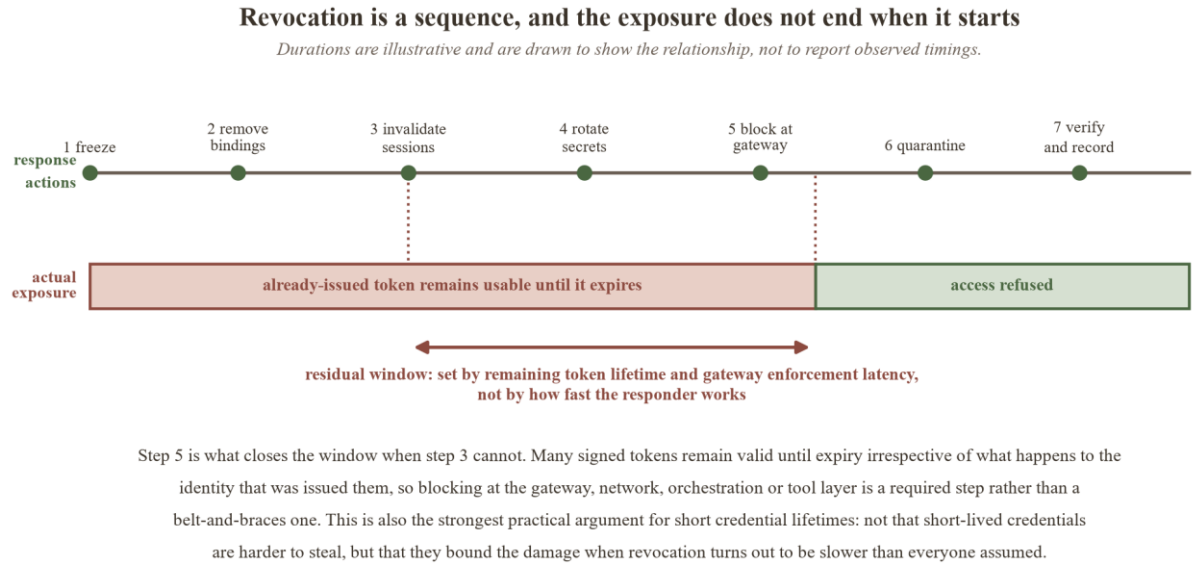


Figure 4. Revocation sequence and residual exposure. The upper track shows the response actions; the lower track shows the interval during which an already-issued token remains usable. The residual window is set by the remaining token lifetime and by gateway enforcement latency, not by response speed, which is why the compensating block at the tool and gateway layer is a required rather than optional step. Durations shown are illustrative to demonstrate the relationship.

Step	Mechanism	Takes effect	Residual risk after this step
1. Freeze identity	Disable in authoritative	Immediate for new	Existing tokens remain

	control plane	authentication	valid
2. Remove bindings	Delete role, group, trust and resource-policy bindings	Immediate to minutes, provider-dependent	Cached authorization decisions may persist
3. Invalidate sessions	Session and token revocation where supported	Immediate where supported; not supported for many signed tokens	Unrevocable tokens valid until expiry
4. Rotate secrets	Rotate or revoke secrets, certificates, keys, signing material	Immediate on rotation; dependent systems may need reconfiguration	Copies of the old secret held elsewhere
5. Block at gateway	Deny at API gateway, network, orchestration or tool broker	Seconds to minutes	Paths that bypass the enforcement point
6. Quarantine workload	Isolate compute and preserve logs	Minutes	Actions already completed
7. Verify and record	Confirm downstream access is refused; record actual revocation time	Verification-dependent	Undiscovered access paths

Table 3. The revocation sequence with residual risk at each step. Step 3 is where the assumption of instant revocation usually fails. The purpose of stating residual risk per step is that an incident responder needs to know what remains possible after each action, not only what has been done.

5. Answering the question at scale

An architecture is only governable if the question what can this identity reach right now can be answered at the moment it is asked. Three mechanisms make that possible.

5.1 The inventory

A continuously updated inventory links every identity to its owner, purpose, creation source, credential type, privilege tier, resources, data, environments, and dependencies. Creation source is included because it is what allows an unrecognized identity to be traced to the pipeline, template, or process that produced it, and therefore to the class of identity rather than the instance.

5.2 The cross-cloud entitlement graph

The entitlement and access-path graph resolves direct permissions, group membership, inherited roles, resource policies, trust relationships, delegated permissions, and agent-to-tool authorization into a single traversable structure.

The graph is necessary rather than convenient because effective permission is not readable from any single policy document. A workload's actual reach is the composition of what its role grants, what the resource policies on the targets allow, what trust relationships permit it to assume elsewhere, and, for agents, what tools it may invoke and what those tools may in turn reach with their own identities. That composition frequently produces reach that no individual policy author intended and that no individual policy review would surface.

The cross-cloud dimension adds a second difficulty. Providers differ in inheritance models, in whether resource policies can grant access independently of identity policies, in how trust is expressed, and in the semantics of deny. A unified graph must therefore normalize the relationships while preserving provider-specific evaluation semantics, because a normalization that flattens a provider's deny precedence produces an answer that is wrong in exactly the cases that matter.

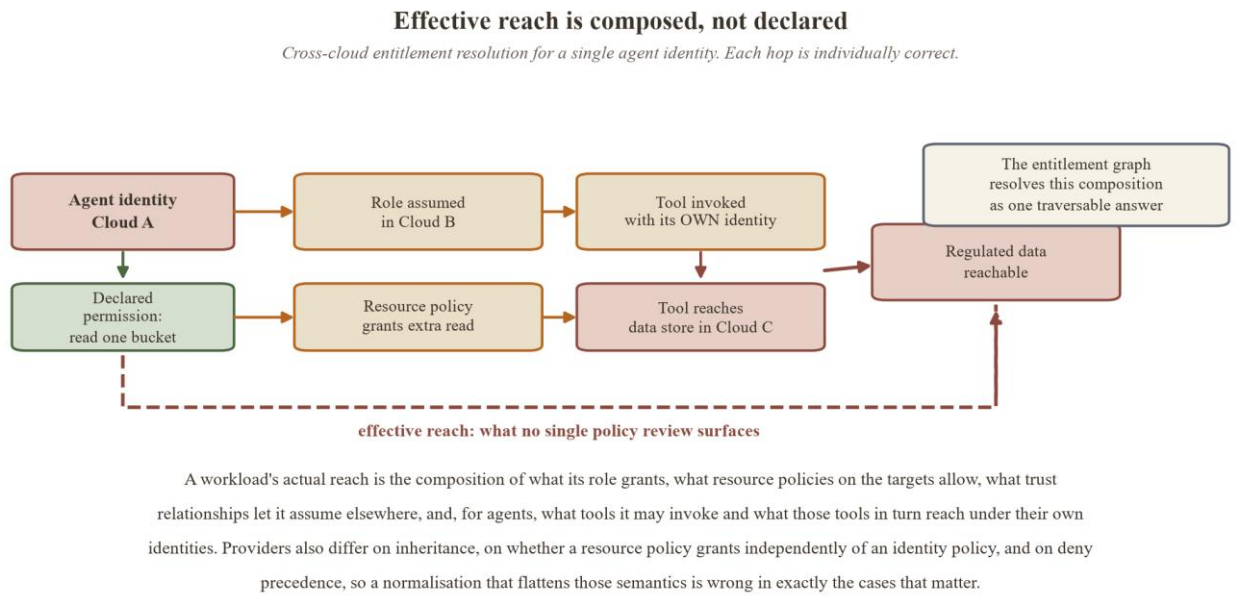


Figure 5. Cross-cloud entitlement and access-path resolution for an agent. The declared permission is narrow; the effective reach, composed through role assumption, resource policy, and tool-mediated access using the tool's own identity, is materially wider. The dashed path is the one no single policy review surfaces, because each hop is individually correct.

5.3 Preferring federation and bounding lifetime

Federation and provider-managed workload identity are preferred so tokens are issued only when the workload proves its identity [4]. Token lifetimes are set from risk, and high-risk actions receive task-specific or transaction-specific authorization rather than a broad reusable token. Rotation is automated, and alerting covers stale, unrotated, unused, exposed, and ownerless credentials.

The emergency revocation function disables the identity, denies new token issuance, removes role bindings, rotates secrets, blocks tool access, and isolates the workload, as a single operation rather than as seven manual steps performed under incident pressure.

6. Where the architecture breaks down

6.1 Agents that need broad access

Some agents require access to many systems to perform orchestration or analysis at all, and excessive restriction makes them useless. This is a genuine tension rather than an implementation detail, and the architecture's answer is to decompose rather than to loosen: separate discovery from execution, grant read-only where possible, issue task-scoped capabilities rather than standing ones, constrain transactions, and require human authorization for irreversible or high-impact actions. Broad read with narrow write is a defensible posture. Broad write is not, regardless of how much easier it makes the agent to build.

6.2 Incompatible identity models

Clouds and SaaS platforms differ in federation support, token semantics, role inheritance, revocation capability, audit detail, and workload attestation. A unified governance layer cannot erase those differences and should not try. It preserves provider-native enforcement while normalizing ownership, risk, evidence, and access relationships, which means the architecture's guarantees are bounded by the least capable participant in any cross-cloud trust relationship.

6.3 Legacy and unsupported systems

Some systems support only usernames, passwords, shared accounts, or long-lived API keys. These produce irreducible residual risk. Compensating controls are vaulting, automated rotation, network restriction, proxy-based access, session recording, and accelerated modernization, and the exception must be dated and owned so that it does not quietly become permanent.

6.4 Central-service outage

A central identity or policy service is a single point of failure and the architecture must assume it will fail. Cached low-risk decisions, provider-native fallback controls, regional redundancy, immutable deny rules, and separately protected break-glass access all apply.

Fail-open behavior is not permitted for high-impact actions. This is the rule most likely to be relaxed under pressure, because the failure it produces is visible and immediate while the risk it prevents is invisible and probabilistic. It should be enforced technically rather than by policy.

6.5 Impersonation and token theft

A valid token does not prove that the intended workload is presenting it. Workload attestation, sender-constrained tokens [16], mutual TLS, signed requests, token binding where available, runtime identity, and behavioral analysis all reduce replay and impersonation exposure, and none eliminates it.

6.6 Governance limits

Inventories and entitlement graphs may be incomplete. Revocation may be delayed by token caching. Permissions may be embedded in application logic where no identity system can see them. Risk scoring may simply be wrong. The design must therefore remain auditable, reversible, and conservative for destructive actions, and it must not present its own coverage as complete when completeness is unverifiable.

7. Contribution boundary

A companion manuscript in preparation by the author addresses security telemetry engineering and the conditions under which a pipeline delivers dependable detection evidence. This paper does not restate that material. Where monitoring in Section 4.7 depends on telemetry being complete, fresh, and fail-visible, that property is a dependency of this architecture rather than a contribution of it.

8. What a production implementation would measure

No enterprise-wide figures are claimed. Table 4 gives the instrument that a production implementation would produce, which is the honest substitute for results that do not yet exist.

Metric	What it establishes
Total non-human identities by class and cloud	The denominator. Every ratio below is meaningless without it
Share with a named accountable owner	Whether the estate is governable at all
Share using federated or managed identity	Progress away from static credentials
Share holding static credentials	The residual exception population
Median and maximum credential age	Whether rotation policy is real or nominal
Unused identities and unused permissions	The removable surface
High-risk permission count	Concentration of blast radius
Review completion rate	Governance coverage, distinct from governance quality
Secrets past rotation policy	Policy-to-practice gap

Time to revoke, and time for revocation to take effect	Two separate metrics. The gap between them is the residual window of Section 4.9
Agent tool calls by risk tier	Whether agent activity matches its approved purpose
Policy denials per agent, trended	Earliest indicator of boundary drift or compromise
Human approvals requested and granted	Whether the approval gate is a control or a formality
Anomalous-use alerts	Behavioral coverage
Access paths to regulated data	The regulator's question, answered from the entitlement graph
Control-state reconciliation variance	Divergence between reported and actual control state, per Section 3.4

Table 4. Metrics a production implementation would produce. Targets should be adopted only after measurement. Reasonable design objectives, stated as objectives rather than as achieved results, are complete ownership for production identities, no unmanaged persistent administrative credentials, short-lived tokens by default, automated rotation for unavoidable secrets, and tested emergency revocation for all high-risk identities.

The next component to build is a machine-identity control plane combining identity discovery, entitlement graphing, workload attestation, policy-based credential issuance, agent tool authorization, continuous access evaluation, and tested cross-cloud revocation. It should expose open interfaces rather than replace provider-native identity and access management, for the reason given in Section 6.2: the differences between providers are load-bearing, and a layer that hides them produces answers that are confidently wrong.

9. Conclusion

Non-human identities became the larger population without becoming the governed one. The controls that make human identity manageable, and they are genuinely good controls, depend on an authoritative lifecycle source, a person who can be held accountable, and a termination event. Machine identities have none of the three, which is why they accumulate and why the accumulation is a structural property rather than a consequence of insufficient care.

The architecture proposed here treats a non-human identity as a governed security object rather than as a directory account: it carries a lifecycle, a dependency graph, a risk tier, an action boundary, and an evidence record, and it has a named human owner from creation. Governance is unified across clouds while enforcement stays provider-native, because the differences between providers are load-bearing and a layer that flattens them answers the wrong question confidently.

The contribution that matters most is the separation of agent identity from per-call authority. For every earlier class of machine identity, establishing who is calling substantially bounded what could be called. Agents break that relationship, and once it is broken, identity assurance and action authorization must be answered independently, before the call, by a component the agent cannot influence. An architecture that authenticates an agent well and authorizes its actions implicitly has solved the easier problem and left the one that matters.

Two honest qualifications close the paper. The

architecture is proposed rather than deployed as a single control plane, and the metrics in Section 8 are an instrument rather than a result. And revocation, which every incident-response plan treats as the moment the exposure ends, ends the exposure only as fast as the shortest token lifetime allows. Designing as though that were not true is the most common and most consequential error in machine identity response.

References

1. OWASP Foundation, "OWASP Top 10 Non-Human Identity Risks," 2025.
2. Verizon, "Data Breach Investigations Report," 2026.
3. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST SP 800-207, Aug. 2020. DOI: 10.6028/NIST.SP.800-207.
4. 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, Sep. 2023. DOI: 10.6028/NIST.SP.800-207A.
5. K. Tallam, "Authorization Propagation in Multi-Agent AI Systems: Identity Governance as Infrastructure," arXiv:2605.05440, 2026.
6. X. Yu, H. Geng, X. Zeng, and W. Knottenbelt, "SUDP: Secret-Use Delegation Protocol for Agentic Systems," arXiv:2604.24920, 2026.
7. K. Patil, "SentinelAgent: Intent-Verified Delegation Chains for Securing Federal Multi-Agent AI Systems," arXiv:2604.02767, 2026.
8. U. Uchibeke, "Before the Tool Call: Deterministic Pre-Action Authorization for Autonomous AI Agents," arXiv:2603.20953, 2026.
9. G. Zhu and C. Wang, "Intent-Governed Tool Authorization for AI Agents," arXiv:2606.22916, 2026.
10. S. V. Kodathala, "aiAuthZ: Off-Host, Identity-Bound Authorization for AI Agents," arXiv:2607.05518, 2026.
11. S. T. Avirneni, "Identity Control Plane: The Unifying Layer for Zero Trust Infrastructure," arXiv:2504.17759, 2025.
12. S. T. Avirneni, "Intent-Aware Authorization for Zero Trust CI/CD," arXiv:2504.14777, 2025.
13. R. N. Rajendran, S. K. Anumula, D. K. Rai, and S. Agrawal, "Zero Trust Security Model Implementation in Microservices Architectures Using Identity Federation," arXiv:2511.04925, 2025.
14. R. M. Sharma, "Ethical Hyper-Velocity (EHV): A Hardware-Rooted Zero-Trust Runtime Enforcement Architecture for Agentic AI Systems," arXiv:2605.17909, 2026.
15. S. J. Lazer, K. Aryal, M. Gupta, and E. Bertino, "A Survey of Agentic AI and Cybersecurity: Challenges, Opportunities and Use-case Prototypes," arXiv:2601.05293, 2026.
16. B. Campbell, J. Bradley, N. Sakimura, and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens," RFC 8705, Feb. 2020. DOI: 10.17487/RFC8705.
17. D. Fett, B. Campbell, J. Bradley, T. Lodderstedt, M. Jones, and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)," RFC 9449, Sep. 2023. DOI: 10.17487/RFC9449.