Article

Agentic Shadow Infrastructure: How AI Supply-Chain Drift Creates Unmanaged Enterprise Infrastructure

Robert Campbell *

Posted Date: 18 June 2026

doi: 10.20944/preprints202606.1454.v1

Keywords: agentic AI; non-human identity; composition drift; shadow infrastructure; capability closure; toxic permission combinations; continuous authorization; Zero Trust

 doi:10.20944/preprints202606.1454.v1

Article

Agentic Shadow Infrastructure: How AI Supply-Chain Drift Creates Unmanaged Enterprise Infrastructure

Robert Campbell

Independent Researcher, Upper Marlboro, MD, USA; rc@medcybersecurity.com

Abstract

Agent identity governance is advancing, though core agent identity and authorization questions remain unresolved: existing frameworks provision, authenticate, authorize, and retire non-human and agentic identities, governing the agent's identity, credentials, and lifecycle while assuming the composition an agent was approved with remains the composition it runs with. This paper argues that assumption is the open seam. An agent's effective composition—its tools, data sources, delegated authorities, policies, and child agents—is a runtime supply chain of capability, and that supply chain drifts. We introduce composition drift as the departure of an agent's effective composition from the terms of its approval, and isolate its most consequential form, compositional drift: the accumulation of individually approved changes into capability that none authorized alone. We formalize this with a two-stage operator: a component-level diff detects that the composition changed (component divergence); a capability-closure stage detects when the change authorized something new (compositional drift)—a qualitative boundary, not a numeric threshold. The contribution is not the observation that approved changes can combine dangerously—long known to authorization security—but a temporal governance model for approved composition drift in agentic systems, linking emergent capability to reauthorization and inventory reconciliation. This drift produces shadow infrastructure: resources provisioned outside any inventory through benign, individually approved pathways. We propose composition attestation, a runtime composition-control layer complementary to identity governance. Paired positive and negative scenarios show the model discriminates, not labels. We bound our claims: the model establishes the phenomenon by construction and claims no deployment efficacy.

Keywords: agentic AI; non-human identity; composition drift; shadow infrastructure; capability closure; toxic permission combinations; continuous authorization; Zero Trust

1. Introduction, Decision Problem, Contributions, and Research Questions

The primary failure mode does not require an attacker or rogue agent. The thesis is that authorized changes compose into unauthorized capability—that an enterprise can do everything correctly at the level of each individual decision and still end up operating infrastructure it never approved, cannot see, and does not govern. This is not a failure of identity governance. It is a phenomenon in a runtime composition-control layer complementary to identity governance—one that current frameworks do not model.

What we mean by agent. This paper addresses agentic workflows, not static automation. A fixed script or RPA flow enumerates its actions in advance and exhausts its authority in its definition; its composition does not change after approval, and it cannot drift in the sense developed here. An agentic workflow is different in kind: it pursues a goal by selecting tools, acquiring data, and spawning sub-agents dynamically at runtime, so its composition is variable by construction. That runtime variability is precisely what makes such systems *compositionally unstable*—able to

 doi:10.20944/preprints202606.1454.v1

accumulate, through individually approved changes, capability that no single approval contemplated. The model targets this class, and that instability is the reason it must.

Agent identity governance is an active and fast-moving area, but its core identity and authorization questions remain unresolved. A growing body of work—agent identity governance frameworks (AIGF), Cloud Security Alliance (CSA) treatments of non-human and agentic identity, vendor lifecycle tooling spanning creation to decommissioning, and CISA's framing of unconstrained identity and access—provisions, authenticates, scopes, and retires agentic identities with increasing rigor [1–3]. These frameworks share a structural assumption that is reasonable and almost universal: that an agent's approved composition is fixed at provisioning. They govern the agent given a stable baseline. They do not model the baseline itself moving.

It does move. An agent approved for a narrow task acquires authority to spin up a temporary internal store for its inputs; the grant is reviewed and approved. It is granted read access to a second data source to improve an output; the grant is reviewed and approved. It is permitted delegated execution to a child worker to parallelize a slow step; the delegation is reviewed and approved. Each change is individually legitimate. Each passes governance. And yet the combination—a scoped store-creation authority, plus a second sensitive data source, plus a child agent that inherits the union of those authorities—can authorize a capability that no single approval contemplated. The agent, acting entirely within its now-effective authority, applies its store authority to the second source, lets the cache persist beyond its approved lifetime, and hands its child agent a credential to reach it. Neither resource appears in any inventory. Neither was approved, because no approval was ever for them—they are the emergent consequence of approvals that were each for something else.

We call the departure of an agent's effective composition from the terms of its approval **composition drift**, and we call the resources it produces **shadow infrastructure**: unmanaged enterprise infrastructure created by an authorized agent, through individually authorized changes. An agent's composition—its tools, data sources, delegated authorities, policies, and child agents—is best understood as a runtime supply chain of capability. Drift in that supply chain, not a compromised dependency or a malicious commit, is the supply-chain risk this paper concerns. That framing is deliberate: the threat is endogenous and benign-by-construction, which is precisely what makes it invisible to controls built to catch malice.

Contribution. We do not claim that capability composition or emergent permission combinations are new; authorization security has long recognized that individually acceptable grants can combine into unacceptable capability (Section 2.2). The contribution is a temporal governance model for approved composition drift in agentic systems, linking emergent capability to reauthorization and inventory reconciliation: an account of how sequentially approved changes to an agent's composition can produce unreviewed capability and unmanaged infrastructure even when identity, credentials, and each individual change all remain compliant. Its defensible novelty is the combination of four elements—an approved-delta trajectory over time rather than a static permission set; runtime composition provenance measured against a signed baseline; detection of the emergent joint reach an accumulating composition can exercise that no single delta authorized; and a trigger that binds that compositional event to reauthorization and inventory reconciliation—with shadow infrastructure as the observable consequence rather than a separate novelty claim. The technical core realizes this as a two-stage operator: a component-level set-difference detects that the composition changed, and a capability closure over the current approved composition B(t) detects when the accumulated change has authorized something outside the union of what was individually approved—the boundary that separates benign evolution from the compositional event that produces shadow infrastructure. We build a drift-detection control, composition attestation, on this model and position it as a runtime composition-control layer complementary to identity governance, feeding lifecycle governance rather than competing with it. The question is now live in federal standards work: NIST's NCCoE concept paper on software and AI agent identity and authorization asks how authorization should be defined and re-evaluated as agents autonomously acquire tools and reach new resources at runtime [4], and this model is one answer at the composition layer.

 doi:10.20944/preprints202606.1454.v1

Scope. This is a framework and model paper. It contains no experiment and claims no deployment efficacy. Its evidence is a worked scenario that demonstrates the phenomenon by construction, paired with a negative scenario that demonstrates the model discriminates. Three adjacent surfaces are deferred and pointed forward explicitly: trust-boundary formalization, authority-creep and contagion through delegation, and the post-quantum credential angle.

This paper is organized around four questions, mapped to the sections that answer them in Table 1.

RQ1—What constitutes composition drift? What is the object that drifts, and how is its drift made precise as distinct from ordinary approved change?

RQ2—When does approved evolution become an authorization failure? Given that approved changes legitimately move the baseline, what is the boundary at which an accumulation of individually approved changes becomes a violation of what was authorized?

RQ3—How can the condition be detected? What architecture observes the composition, computes the boundary condition, and surfaces a flagged event for governance?

RQ4—What evidence demonstrates that the model discriminates rather than labels? What shows the condition is not trivially always-present (every evolution flagged) or always-absent?

Table 1. Research questions mapped to the sections that answer them.

Question	Answered in
RQ1	Sections 3.2 and 4 (Definitions and System Model; Temporal Composition and Authorization Model)
RQ2	Section 4 (Temporal Composition and Authorization Model—capability closure and the boundary)
RQ3	Section 6 (Continuous Composition Authorization and Attestation Architecture)
RQ4	Sections 7 and 8 (Evaluation Design and Scenarios; Results and Comparative Analysis)

2. Related Work and Novelty Matrix

This section does heavy lifting, because the defensibility of a narrow contribution depends on demonstrating command of the ground it is narrow within. We cover two bodies of work generously: the identity and lifecycle frameworks that govern agents, and—equally important and easily overlooked—the authorization-security literature on emergent privilege from permission combinations. The contribution lives in the gap between them.

2.1. Agent Identity and Lifecycle Governance

Recent frameworks treat the non-human and agentic identity as a first-class governed object. Agent identity governance frameworks enumerate identity categories and map them onto established standards—OAuth and OIDC for delegated authorization and authentication, SPIFFE/SPIRE for workload identity issuance, and SCIM for lifecycle provisioning and deprovisioning [1,5–9]. The Cloud Security Alliance's treatment of non-human identity contributes two ideas this paper builds on directly: ephemeral permanence, the observation that identities meant to be short-lived persist far beyond their intended window, and persistent blast radius, the corresponding observation that the authority attached to those identities outlives the task that justified it [2]. Vendor lifecycle tooling operationalizes a creation-to-decommissioning model and names the failure of its tail end—ghost agents, identities that are never retired and accumulate silently [2]. CISA frames the systemic risk as unconstrained identity and access [3].

What these frameworks share is the engine of their effectiveness and the seam this paper addresses: they govern identity and lifecycle given a stable approved composition. AIGF's categories

 doi:10.20944/preprints202606.1454.v1

scope an agent's authority at provisioning; CSA's blast-radius reasoning bounds the authority an identity carries; vendor decommissioning retires the identity when its lifecycle ends. None of them models the approved composition as a time-varying object that diverges from its baseline while the identity remains valid and the lifecycle remains correctly governed. Our model sits beneath this layer: it takes the agent's identity and lifecycle as governed, and asks what happens to the composition that identity was approved with, between provisioning and retirement.

2.2. Emergent Privilege from Permission Combinations

The idea at the heart of this paper—that individually acceptable grants can combine into unacceptable capability—is not new to authorization security, and the paper would be indefensible if it pretended otherwise. Three lineages occupy this ground.

First, **toxic permission combinations**: identity and access management practice has long recognized that two individually reasonable permissions can be jointly dangerous, and segregation-of-duties and toxic-combination analysis exist precisely to detect such pairs in a permission set [10]. Second, **privilege-escalation-path analysis**: tooling in the attack-path lineage, exemplified by BloodHound, models a directed graph of identities, permissions, and resources and computes reachability—escalation paths that no single edge reveals but that the graph's transitive closure exposes [11,12]. Third, the **confused deputy**: the classical observation that an authorized intermediary can be induced to exercise its authority on behalf of a less-privileged caller, so that capability emerges from the composition of caller and deputy rather than from either alone [13].

Our delta is specific and, we argue, real. Each of these lineages analyzes a static permission set, graph, or delegation relation for emergent privilege—the analysis is a snapshot, and the dangerous combination is present in the snapshot whether or not anyone approved its constituents. We model the time-varying drift of an approved composition in which every constituent change individually passed governance. The novelty is not the observation that combinations can be toxic; it is locating that observation in a setting where governance is functioning correctly at every step, where there is no static set to scan because the dangerous set is assembled incrementally over time through approved deltas, and where the governed object is an agent's runtime composition rather than a user's standing entitlements. Toxic-combination analysis asks "is this permission set dangerous?"; we ask "at which approved delta did an accumulating composition first authorize capability outside the union of what each delta authorized?" The former is a property of a state; the latter is a property of a trajectory through approved states. That distinction is the contribution's foundation, and Section 4 makes it formal.

2.3. Continuous Authorization and Configuration Reconciliation

Two operational practices are adjacent inputs rather than competitors. Continuous-authorization and ongoing-authorization-to-operate models re-evaluate a system's authorization on a recurring basis rather than at a single accreditation point [14,15]. Configuration-management and infrastructure-as-code reconciliation detect drift between declared and actual infrastructure state [16,17]. Both are mechanisms our drift signal can feed: the compositional-capability event of Section 4 is exactly the kind of trigger a continuous-authorization regime should consume, and the shadow resources of Section 5 are exactly what CMDB/IaC reconciliation is positioned to surface once it knows to look. We return to these handoffs in Section 9.1.

A more direct line of recent work targets the runtime governance of agentic systems specifically. Wang et al. introduce MI9, an integrated runtime governance framework that enforces safety over an agent's live behavior through continuous-authorization monitoring, finite-state conformance engines, and goal-conditioned drift detection—motivated, as our model is, by the observation that agentic systems exhibit emergent runtime behavior that pre-deployment controls cannot fully anticipate [18]. MI9 corroborates the premise that emergence is a runtime phenomenon demanding runtime governance, but it governs a different object. Its conformance engines and goal-conditioned detectors track behavioral drift—how an agent's executing trajectory diverges from an intended goal

 doi:10.20944/preprints202606.1454.v1

or policy automaton—whereas our two-stage model tracks structural drift in the authorization substrate: the emergence of shadow infrastructure (Section 5) and the super-additive capability closure of a composed, supply-chain-assembled baseline measured against a policy invariant (Section 4). Behavioral conformance and capability closure are orthogonal surfaces—what an agent is doing versus what its accumulated composition can reach—so a deployment instrumented for one can remain blind to the other.

Parallel efforts harden agentic systems at the identity, guardrail, and policy layers, and are worth delimiting precisely because they are close. SAGA supplies a user-controlled lifecycle architecture whose cryptographic token mechanism gives fine-grained, formally guaranteed control over inter-agent interaction [19]; GuardAgent synthesizes runtime guardrails by translating natural-language safety requests into executable, code-based checks that gate a target agent's actions [20]; and AGENTSAFE operationalizes a risk taxonomy into design-, runtime-, and audit-time controls spanning semantic telemetry, dynamic authorization, and interruptibility [21]. Each hardens a distinct control surface, and none subsumes the present contribution. SAGA governs whom an agent may transact with and under what token, not whether an accumulation of individually approved interactions has enlarged the agent's joint reach; GuardAgent and AGENTSAFE evaluate actions and behaviors against a fixed policy or taxonomy at execution time, not the capability closure of a composition drifting against a signed baseline. These layers enforce the rules; our model detects when an approved composition has silently changed which rules should apply—the structured trigger such enforcement is positioned to consume.

2.4. Access-Control Models and Policy Enforcement

A broader access-control literature supplies the machinery these frameworks enforce, and it is adjacent enough to delimit. Usage control (UCON) generalizes access control with continuity of decisions and mutable attributes, re-evaluating authorization during access rather than only at request time [22]; attribute- and relationship-based access control (ABAC, ReBAC) make policy expressive, deciding from subject, resource, and environment attributes or from relationships among principals [23,24]; and policy-as-code admission control evaluates each proposed change against codified rules at an admission gate [25]. Each is a complement, not a competitor, and each leaves the same seam open: UCON re-evaluates a subject but presumes a fixed rule set to evaluate, not a composition whose joint reach is itself drifting; ABAC and ReBAC decide a single request without computing the closure of an accumulating baseline; and an admission gate rules on one delta in isolation—exactly the per-increment decision whose local admissibility we show is insufficient. Our model is downstream of all three: P(t) can be expressed in their policy terms, while the compositional-drift event is the structured trigger a policy-as-code gate or UCON monitor should consume.

2.5. Novelty Matrix

Table 2 positions this work against the adjacent approaches reviewed above. The contribution is a temporal governance model for approved composition drift in agentic systems, linking emergent capability to reauthorization and inventory reconciliation. The matrix scores it across the four axes that define that contribution: an approved-delta trajectory over time, composition provenance measured against a signed baseline, detection of emergent joint reach, and a trigger that binds the compositional event to reauthorization and inventory reconciliation. Only the present approach spans all four; each prior approach addresses a subset.

 doi:10.20944/preprints202606.1454.v1

Table 2. Novelty matrix: the present approach against adjacent approaches across the four contribution axes.

Approach	Approved-delta trajectory	Provenance vs. signed baseline	Emergent joint reach	Trigger → reauthorization + inventory
Identity & lifecycle governance [1,2]	—	Partial	—	Partial
Static IAM & toxic-combination [10]	—	—	Partial	—
Attack-path analysis [11,12]	—	—	Partial	—
Continuous monitoring & config reconciliation [14,26]	Partial	Partial	—	Partial
Runtime agentic governance & guardrails [19–21]	—	Partial	—	Partial
This work (composition attestation)	✓	✓	✓	✓

Notes: ✓ = the approach fully addresses the axis; Partial = the approach addresses the axis only partially or indirectly; — = the approach does not address the axis.

3. Threat Model, Assumptions, and Definitions

3.1. Threat Model and Assumptions

Agent posture. The model is indexed by the capability outcome, not by intent, and applies across three postures.

- **Benign agent.** Operates within its granted authority and optimizes toward its task. Compositional drift arises here *without malice*: the agent provisions, delegates, and recombines using authorities each of which was approved. This is the primary and most consequential case, because it shows the failure needs no adversary and evades controls that key on intent.
- **Compromised agent.** An attacker controls an otherwise-approved agent and can deliberately steer recomposition toward emergent capability. The trigger is unchanged—it fires on the realized failure *effective_reach*(B(t); G(t)) ⊄ P(t) regardless of who drove the composition there.
- **Adversarial agent.** A deliberately malicious component or delegate is introduced through an approved-looking change, engineering a toxic combination across approvals. The model treats this as an admissible-looking sequence whose joint reach violates P(t); detection is identical, attribution is a governance concern.

Observability.

- **Observable composition**—components and capabilities reflected in a signed or queryable artifact (manifest, CMDB, IaC state, credential graph). Detectable by the component diff.
- **Unobservable (runtime-only) composition**—materializes at runtime and is recorded in no artifact. By construction outside the diff's reach; it bounds completeness (see Section 8.1) and is named, not addressed.

Authorization topology.

- **Centralized authorization.** A single authority maintains P(t); the cleanest case, where admissibility and the joint-reach test reference one invariant.
- **Federated authorization.** Multiple authorities each admit changes against their own envelope. The joint reach can violate a global invariant that no single authority evaluated; federation

 doi:10.20944/preprints202606.1454.v1

amplifies compositional drift through cross-authority emergence, and H(t) provenance must span authorities for the per-increment admissibility record to be reconstructable.

Boundaries. Composition can span cloud, on-premises, and cross-domain administrative boundaries. Cross-domain delegation—a child agent in another domain inheriting authorities—extends transitive reach across a boundary where neither domain's authority observed the whole; this is where shadow infrastructure and delegated reach are hardest to reconcile and where the cloud/CMDB/IaC reconciliation and delegation-graph components are most load-bearing.

Assumptions. We assume P(t) is specifiable as a policy invariant; that approvals are recorded in H(t) with their contexts; that discovery faithfully observes the *observable* portion of E(t); and that reach and effective_reach are computable approximations (the soundness conditions are stated in Section 8.1). We do **not** assume agents are honest, that composition is fully observable, or that authorization is centralized—the model degrades gracefully under each, and its limits are stated rather than assumed away.

3.2. Definitions and System Model

We fix terminology before formalizing the model.

Agent. A non-human identity with authority to act—to invoke tools, read and write data, exercise delegated permissions, and, in the agentic case, to spawn and direct child agents.

Approved composition baseline (versioned), denoted B(t). The set of components the agent is approved to operate with as of time t, signed and attested at each version. Components span the drift dimensions of Section 4.1: tools, authorities, data sources, policies, delegations, and infrastructure bindings. B(0) is the provisioning baseline; each approved change advances it to a new attested version, so an approval at time t yields B(t). The baseline is versioned, not frozen—an approved configuration change produces a new approved baseline, as change-management practice requires—and the forward trajectory of B(t) is the agent's approved evolution. The reference against which divergence is measured is therefore a moving but attested record; what it must not absorb silently is unapproved change, a distinction carried by D(t) below.

Effective runtime composition, denoted E(t). The set of components the agent actually operates with at time t, observable and unobservable (Section 4.4). E(t) may lead or lag B(t): it can contain unapproved additions not (or not yet) in the baseline, and can omit approved components not yet deployed.

Component divergence, denoted D(t) = E(t) △ B(t). The symmetric difference between the effective runtime composition and the approved baseline: components running but not approved (E(t) \ B(t)) together with components approved but not running (B(t) \ E(t)). D(t) is unapproved configuration drift—the gap between what is approved and what is actually present. By construction it excludes approved evolution, which is absorbed into B(t); D(t) isolates only what no approval covers. The capability-expanding direction is E(t) \ B(t), the model's focus; B(t) \ E(t) is tracked for completeness.

Approval history (provenance), denoted H(t). The ordered record of approved changes c_1, c_2, …, c_n together with their approval contexts—for each c_i, the baseline version it was approved against and the capability the approval evaluated. H(t) is the runtime composition provenance: it makes recoverable what each change was individually approved to do, and is the source from which the individually authorized union U(t) of Section 4.3 is reconstructed.

Authorized-capability envelope, denoted P(t). The enterprise policy invariant against which authorization is judged—not a flat set of permitted capabilities but a constraint structure carrying purpose limitation, data residency, time and duration limits, separation of duties, delegation depth, environmental conditions, and negative constraints, together with the deny rules, network restrictions, quotas, and guardrails that enforce them. An approval establishes a grant admissible under P(t); the standing invariant is what the accumulated baseline must continue to satisfy. A sequence of individually admissible changes can advance B(t) into a configuration whose joint

 doi:10.20944/preprints202606.1454.v1

reachable capability violates P(t)—the third phenomenon the model separates, alongside approved evolution and component divergence.

Composition drift. The post-approval evolution of an agent's composition, made precise in two stages—component divergence at the component level (D(t)) and compositional drift at the capability level—because, as we now show, only the second discriminates the consequential case from benign approved evolution and benign component divergence alike.

Positioned against identity governance: lifecycle frameworks govern the identity that holds a composition and treat its approved baseline as given. We make that baseline an explicit, versioned, attested object B(t), and govern three things it leaves unmodeled—the component divergence D(t) of runtime from baseline, the provenance H(t) of how the baseline was assembled, and the joint capability of the baseline against the authorized envelope P(t).

4. Temporal Composition and Authorization Model

This is the core contribution. We define drift in two stages. The first stage—component diff—answers did the composition change? The second stage—capability closure over the current approved composition B(t), evaluated at each diff—answers did the change authorize something new? The distinction is the whole point: a component-level measure detects change but cannot, on its own, separate benign evolution from the compositional event that produces shadow infrastructure. We make both stages precise and then define compositional drift as a qualitative boundary in capability space (Figure 1). Throughout, component divergence names the component-level signal D(t) (Stage one) and compositional drift the capability-level violation (Stage two); composition drift is the overall phenomenon.

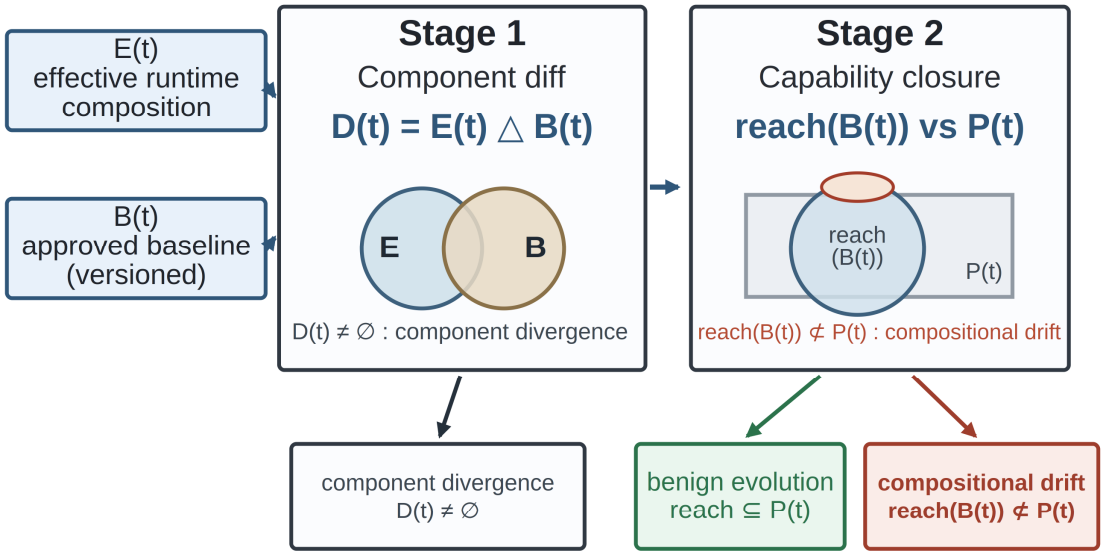

Figure 1. Two-stage composition-drift model. Stage one computes the component diff D(t) = E(t) △ B(t); stage two computes the capability closure *reach*(B(t)) and tests it against the policy invariant P(t). Compositional drift is the boundary crossing *reach*(B(t)) ⊄ P(t), which can occur with D(t) = ∅.

4.1. Drift Dimensions

An agent's composition spans six dimensions, and drift can occur in any of them:

1. **Tool drift**—tools added to or altered in the agent's effective toolset.
2. **Authority drift**—changes in the permissions and scopes the agent can exercise.

 doi:10.20944/preprints202606.1454.v1

3. **Data drift**—changes in the data sources the agent can read or write.
4. **Policy drift**—changes in the policies that govern the agent's behavior.
5. **Delegation drift**—changes in the agents the agent may spawn or direct, and the authority it passes to them.
6. **Infrastructure drift**—changes in the infrastructure bindings (endpoints, stores, compute) the agent is attached to.

These dimensions are mostly orthogonal to software-supply-chain artifacts such as dependency manifests and CVE feeds; the "supply chain" here is the agent's runtime supply of capability, not its build-time dependency graph. Components are typed by dimension, which the capability stage uses.

4.2. Stage One—Component Diff

Let the **component divergence** at time t be the symmetric difference between the effective composition and the versioned approved baseline:

$$D(t) = E(t) \triangle B(t)$$

D(t) is the set of components running outside the approved baseline (E(t) \ B(t)) together with approved components not running (B(t) \ E(t)). The capability-expanding direction is E(t) \ B(t); B(t) \ E(t) is tracked for completeness but is not the source of capability expansion and we set it aside. D(t) is computed as a plain symmetric set-difference, and is observable to the extent that E(t) is observable (Section 4.4).

Unlike a diff against a frozen provisioning baseline, D(t) does not flag approved evolution—approved changes advance B(t) and leave D(t) unchanged—so $D(t) \neq \emptyset$ is already a meaningful signal: it is unapproved configuration drift, the classic shadow-and-divergence condition. But D(t) is not the paper's signature phenomenon. The compositional case can occur with $D(t) = \emptyset$, entirely inside an approved baseline B(t) assembled from individually approved changes, when the joint capability of B(t) exceeds what those changes were individually authorized to confer. That case is invisible in component space and lives in capability space, which is the second stage.

4.3. Stage Two—Capability Closure, and the Definition of Compositional Drift

We refine capability into two functions, because effective authorization is not monotone and the model must not rest on a property it lacks. Let ***reach*(S)** denote the **potential reachability** of a component set S: the capabilities S could in principle reach or compose, as a permission-and-interaction closure abstracting from restrictive controls. reach is the formal home of "the combination does more than its parts," and on it two properties hold:

- **Monotonicity (reachability):** if $S \subseteq S'$ then *reach*(S) $\subseteq$ *reach*(S'). Adding components never removes potential reach. (Effective exercisable capability, effective_reach, below, is not monotone; that is the point of separating the two.)
- **Super-additivity:** *reach*(X $\cup$ Y) $\supseteq$ *reach*(X) $\cup$ *reach*(Y), and the inclusion is sometimes strict. The combination of two component sets can reach capability that neither reaches alone. Strictness is exactly where compositional capability lives; when reach is merely additive (the inclusion is equality), there is no compositional effect to govern.

Effective (exercisable) capability, *effective_reach*(S; G). What S can actually exercise once the guardrail and enforcement context G—the deny policies, network restrictions, quotas, and guardrails in force—clamps its potential reach; always *effective_reach*(S; G) $\subseteq$ *reach*(S). effective_reach is **not** monotone in the guardrail context: adding an enforcing guardrail can only reduce exercisable capability, so *effective_reach*(S; G $\cup$ {g}) $\subseteq$ *effective_reach*(S; G) for a guardrail g. effective_reach is context-dependent—it depends on G and on interactions among components—and it is what governs whether reachable capability is consequential. We make claims about emergence on reach, where monotonicity holds, and claims about whether emergence matters on effective_reach against the policy invariant P. The guardrail context G(t) (what is enforced) is distinct from the policy invariant

 doi:10.20944/preprints202606.1454.v1

P(t) (what is allowed): a realized violation is exactly the case where what remains exercisable under G(t) still escapes P(t).

Now consider an approved baseline advanced by a sequence of individually approved changes $c_1, c_2, \ldots, c_n$ recorded in H(t), so that $B(t) = B(0) \cup \{c_1, \ldots, c_n\}$. Each c_i passed governance against the envelope in force when it was approved: per-change review evaluates the declared local grant of c_i—the capability the change is understood to confer in its approval context ctx_i, written $grant(c_i; ctx_i)$—against the policy invariant $P(t_i)$ that approval carried: the purpose, duration, data, delegation, and negative constraints. Local review evaluates only what the change declares; it does not compute the joint closure $reach(B_{i-1} \cup \{c_i\})$ of the accumulated baseline with the new change, which would already expose the super-additive interactions that constitute emergence. The **per-increment admissibility condition** is that every change's declared grant was within the envelope in force at its approval:

$$grant(c_i; ctx_i) \subseteq P(t_i) \text{ for every } i$$

That is what the enterprise said yes to: each change in the context that approved it—a set of constrained grants, not a flat union of capabilities. What no approval evaluated is the joint reachable capability of the accumulated baseline against the standing invariant. Define the **joint reachability** as the reach of the whole approved baseline:

$$reach(B(t)) = reach(B(0) \cup \{c_1, \ldots, c_n\})$$

Compositional drift (a compositional authorization failure) is then defined as the case where the baseline's joint reachable capability violates the policy invariant although every increment was individually admissible:

$$\text{Compositional drift at time } t \text{ iff } reach(B(t)) \not\subseteq P(t) \text{ with } grant(c_i; ctx_i) \subseteq P(t_i) \text{ for every } i$$

where $\not\subseteq$ denotes 'not a subset': the baseline jointly reaches capability outside the envelope, while no single increment did—the failure is emergent, produced by composition rather than by any one approval. P(t) carries the constraints a union of reachable capabilities cannot: purpose limitation, data residency, duration, separation of duties, delegation depth. Monotonicity and super-additivity of reach are what make the joint reach able to exceed every increment. Equivalently, $\bigcup_i grant(c_i; ctx_i) \subseteq P(t)$ while $reach(B(t)) \not\subseteq P(t)$: the union of declared local grants stays within the envelope, yet the joint reach does not, and the strict gap $\bigcup_i grant(c_i; ctx_i) \subsetneq reach(B(t))$ is precisely the emergent capability no local review evaluated. The event can occur with $D(t) = \emptyset$, entirely inside the approved baseline, assembled from changes that each passed review—the precise sense in which approved changes jointly exceed what any approved alone.

The envelope condition above is stated on potential reach. The realized condition clamps by the guardrail and enforcement context G(t): a deny policy, quota, or guardrail in force may hold the emergent reach inside P(t), in which case the violation is potential but not realized. Writing the realized condition with effective_reach:

$$\text{Realized failure at } t \text{ iff } effective\_reach(B(t); G(t)) \not\subseteq P(t)$$

That is: some capability that is effectively exercisable—surviving the deny rules, quotas, and guardrails actually in place—violates the policy invariant P(t). Where those controls clamp the emergent reach back inside P(t), the failure is potential but not realized, and the model credits the guardrail, which a monotone closure could not represent. The trigger the control fires on (Section 4.5) is this realized failure; the potential form $reach(B(t)) \not\subseteq P(t)$ is the weaker existence condition.

Equivalently: the agent's effective composition can exercise a capability that violates an approved policy invariant—exceeding the envelope on purpose, data, delegation, duration, or separation of duties—although every approved change was admissible when made. This is the load-bearing claim, stated against a policy invariant rather than a union of reachable capabilities. The emergence follows from super-additivity of reach; the failure follows from effective_reach against P(t). Separating them is what lets the model assert emergence without resting on the false premise that effective authority only grows.

This formulation resolves a tension a single-stage set-difference leaves open. The component divergence D(t) tracks runtime departure from baseline but is silent on the compositional case, which

 doi:10.20944/preprints202606.1454.v1

can occur with D(t) = ∅. The relation *reach*(B(t)) ⊄ P(t) marks the emergent event inside the approved baseline; the realized condition marks when it is consequential. Both are qualitative boundaries—violations of a policy invariant—not thresholds on an index.

4.4. A Taxonomy of Drift

With both stages in hand, drift admits a taxonomy:

- **Atomic drift**—a single approved change; B(t) gains one component.
- **Cumulative drift**—many approved changes accumulate; B(t) grows, but *reach*(B) may remain within the envelope P(t) (the benign case Section 7.3 exhibits).
- **Compositional drift**—the case above: *reach*(B) ⊄ P(t), realized when the emergent reach survives the guardrail context. The dangerous case.
- **Transitive drift**—drift propagated through delegation: a child agent's effective composition inherits and recombines authorities from its parent, so compositional drift can manifest one delegation hop away from where its constituents were approved.
- **Unobservable (runtime-only) drift**—components or capabilities that materialize only at runtime and are reflected in no signed or queryable artifact. This category is named here precisely because it bounds the detection control: a diff against a manifest cannot see what no manifest records. Section 6.6 makes this limit explicit.

4.5. Re-Evaluation Triggers, Not an Index

The model intentionally produces no scalar to threshold. The re-evaluation trigger is the realized failure of Section 4.3: *effective_reach*(B(t); G(t)) ⊄ P(t). When the approved baseline first reaches capability that, surviving the guardrail context, violates the policy invariant P(t)—although every increment was admissible when approved—the agent's authorization is, by construction, no longer the authorization it was granted, and re-evaluation is warranted. This trigger is expressed as a handoff into the existing lifecycle layer—it feeds reauthorization and, where appropriate, revocation and retirement under AIGF governance (Section 6.5, Section 9.1). We feed those frameworks; we do not replace them. We resist a severity score deliberately: the compositional-drift event is itself the prioritization primitive—a binary fact about whether effectively exercisable capability escaped the authorized envelope—and any ranking of flagged events by severity is a downstream judgment for the governance layer, applied with its own risk model to the events this trigger surfaces, not a quantity the detector computes.

4.6. Qualitative Triage Ordering

Triage does not require a scalar. The taxonomy of Section 4.4 supplies three qualitative discriminants that order flagged events by kind: whether the emergent capability crosses a trust or data-sensitivity boundary (a delegated, independently reachable copy of a sensitive source outranks an in-place transformation); whether it is transitive rather than atomic (propagated through delegation, and so one or more hops from where its constituents were approved, and harder to contain); and which drift dimensions combined (data-plus-delegation expansions reach further than tool-plus-policy ones). These are ordinal, governance-facing distinctions: they tell a reviewer which flagged events to take first, while the decision itself remains the qualitative event of Section 4.3.

5. From Compositional Authorization Failure to Shadow Infrastructure

Compositional drift expands an agent's effective authority. Once that authority crosses certain lines, the agent—acting entirely within it—provisions resources outside any inventory. These resources are the shadow infrastructure of the title: unmanaged enterprise infrastructure produced through individually authorized pathways.

 doi:10.20944/preprints202606.1454.v1

5.1. Classes of Shadow Infrastructure

Shadow infrastructure is not a single kind of artifact. The resources an agent provisions through drift fall into a small number of recurring classes, and naming them makes the otherwise abstract notion concrete:

- **Credential infrastructure**—API keys, OAuth tokens, and service accounts the agent mints or acquires to reach the resources it creates.
- **Data infrastructure**—buckets, caches, and object stores the agent stands up to hold intermediate or persisted data.
- **Workflow infrastructure**—scripts, scheduled jobs, and serverless functions the agent creates to automate steps.
- **Service infrastructure**—databases, containers, and microservices the agent provisions to run or expose work.
- **Knowledge infrastructure**—vector stores, embedding indices, and shared agent memory: the retrieval substrates (RAG indices) an agent builds to cache, embed, or share what it has processed.

These classes are not a threat taxonomy, and we attach no creation-pathway, persistence, or propagation apparatus to them here; they are simply the concrete forms the unmanaged resources of this paper take. The worked scenario of Section 7 turns on the last class—a knowledge-infrastructure resource—because retrieval substrates are where a sensitive source most readily persists as a durable, independently reachable copy.

5.2. Benign Pathways

Shadow infrastructure arises through ordinary, optimizing behavior, not sabotage:

- **Optimization**—the agent provisions a cache or store to avoid recomputing or re-fetching, because doing so is faster and falls within its effective authority.
- **Recovery**—the agent provisions a fallback resource to survive a failure, because resilience is within its effective authority.
- **Persistence of temporary resources**—a resource provisioned for a transient task is never torn down (the ephemeral permanence of CSA), and persists as standing infrastructure.
- **Delegation**—the agent spawns a child and hands it the credentials and bindings to reach a resource, propagating both the resource and the authority to it (transitive drift).

5.3. Shadow-State Criteria

A resource is shadow infrastructure when it satisfies, jointly: (i) it is operational—it exists and is reachable by the agent or its delegates; (ii) it is outside inventory—it appears in no CMDB, asset register, or IaC declaration; and (iii) it is unapproved-by-emergence—no approval was ever for this resource, because it is the emergent consequence of approvals each granted for something else. The third criterion is what distinguishes shadow infrastructure from ordinary unmanaged assets: it is not that someone forgot to register it, but that no decision to create it was ever made.

5.4. Why It Matters: Risk, Compliance, and Cost

The shadow-state criteria of Section 5.3—operational, outside inventory, unapproved-by-emergence—are themselves the properties that become consequence for the organizations that own these resources. We name three impact vectors, not as measured findings but as the decision-relevant forms the phenomenon takes; each follows from the criteria, not from deployment data.

- **Unbudgeted spend (cost)**—a resource outside inventory is outside cost accounting. The vector stores, caches, and compute an agent provisions consume metered cloud resources that no FinOps process tracks, because no process knows they exist. Cost visibility fails at the same seam as security visibility.

 doi:10.20944/preprints202606.1454.v1

- **Audit and compliance exposure**—a resource outside inventory is outside the controls compliance presumes. The durable copy of a sensitive source in an unapproved vector store (Section 7) may sit in a region, account, or boundary that violates data-residency or data-handling obligations—not through misconfiguration of an approved system, but because the system was never approved and therefore never assessed. An audit reconciling against the approved baseline cannot find what diverged from it.
- **Expanded attack surface**—each shadow resource is reachable infrastructure no one is watching. The credentials minted to reach it, the caches holding sensitive data, and the child agents that inherit access all enlarge the surface for lateral movement while staying invisible to monitoring scoped to approved assets. The emergence that hides the resource from inventory hides it from defense.

These impacts share one root with the detection problem: the resource is consequential precisely because it is unmanaged, and unmanaged precisely because no decision created it.

6. Continuous Composition Authorization and Attestation Architecture

We propose a control that operationalizes the two-stage model. It is a runtime composition-control layer complementary to identity governance—it detects drift and hands off—and it certifies less than its name might suggest, which we state plainly. Section 6.1 sets out the control as an architecture that separates detection from enforcement; the subsections that follow detail its parts.

6.1. Control Architecture: Detection Separated from Enforcement

Detection establishes that a compositional event occurred; enforcement decides what to do and, where possible, interposes on the emergent action before it becomes standing infrastructure. A signed manifest belongs to detection—it records the approved baseline and surfaces component divergence D(t)—but it cannot compute the capability closure *reach*(B(t)) against P(t), and it cannot prevent an action. Those are distinct functions. The control is therefore an architecture of eight components across two planes (Figure 2).

Detection plane—observe the composition, compute the divergence, surface the evidence. **Versioned composition manifest:** the signed record of B(t), re-signed at each approved change, supplying the reference for the component diff. **Tool and resource discovery:** observation of the effective composition E(t)—the tools, data sources, and resources actually present, including those provisioned at runtime. **Delegation and credential graph:** the delegations and credential flows among agents, so transitive reach (a child inheriting and recombining authorities, Section 4.4) is observable rather than hidden. **Runtime event and provenance collection:** the approval history and runtime capability events, supplying H(t) and the evidence a flagged event carries. **Cloud/CMDB/IaC reconciliation:** reconciliation of E(t) against asset inventories and infrastructure-as-code state, surfacing shadow infrastructure that no manifest records.

Enforcement plane—compute the violation, decide, prevent where possible, act through governance. **Policy-invariant engine:** evaluates *reach*(B(t)) and *effective_reach*(B(t); G(t)) against the policy invariant P(t); this is where capability closure is computed and the compositional-authorization failure is decided—the function a manifest cannot perform. **Pre-action authorization gate:** a runtime checkpoint able to interpose on an emergent action before it executes, the preventive control a detection-only manifest lacks. **Reauthorization and revocation workflow:** the lifecycle handoff (Section 6.5) that routes a flagged event to reauthorization, constraint, or revocation under existing governance.

 doi:10.20944/preprints202606.1454.v1

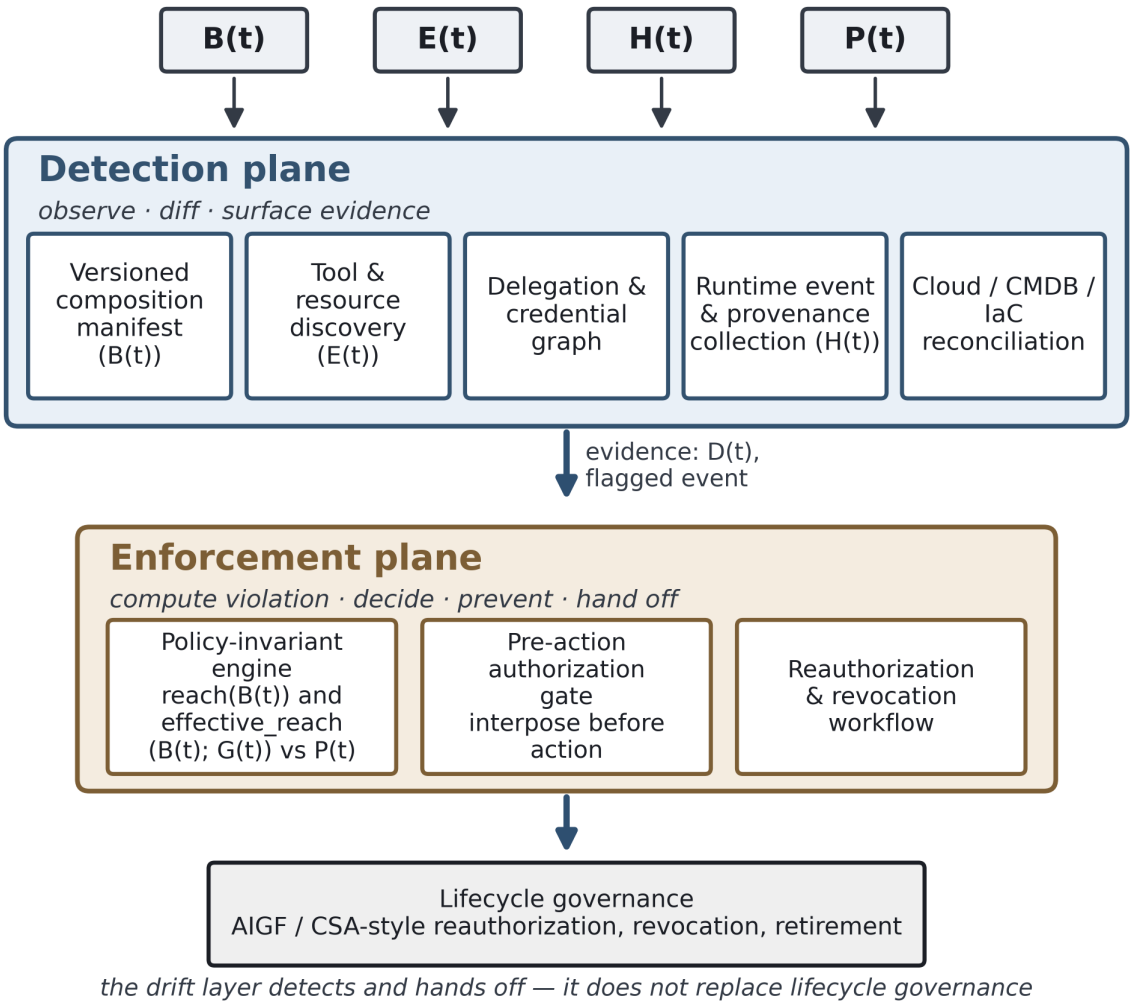

Figure 2. Detection-and-enforcement architecture. The detection plane (versioned composition manifest, tool and resource discovery, delegation and credential graph, runtime event and provenance collection, and cloud/CMDB/IaC reconciliation) observes the composition and surfaces evidence; the enforcement plane (policy-invariant engine, pre-action authorization gate, and reauthorization and revocation workflow) computes the violation and hands off to lifecycle governance.

Separating the planes matters because they have different evidence needs and different failure modes, and because deployed monitoring is typically weakest where this architecture is most load-bearing: NIST identifies fragmented logging and the difficulty of detecting drift in deployed systems as current monitoring barriers [4]—the gap the discovery, provenance, and reconciliation components are arranged to close. We specify this architecture; we do not deploy or evaluate it. No coverage, false-positive rate, latency, or prevention-efficacy figure is claimed for any component—the contribution is the separation of functions and their arrangement, not a validated system.

6.2. Signed Composition Manifest as the Drift Baseline

At provisioning, the agent's approved baseline B(0) is captured as a signed composition manifest: the typed set of components across the six dimensions, attested. Each approved change re-signs the manifest to a new version B(t), so the manifest is itself versioned; the current version is the reference for stage one.

 doi:10.20944/preprints202606.1454.v1

6.3. Drift Detection as Continuous Diff

Detection computes D(t) = E(t) △ B(t) continuously against the current signed manifest. This is the component-divergence stage, surfaced as an operational signal: the manifest is the declared (approved) composition, the runtime is the actual composition, and the divergence is the drift.

6.4. The Compositional-Capability Check at the Diff

The novel control action is at the second stage. Rather than evaluating each change in isolation—which is what per-change governance already does well—the control evaluates the joint reach of the combination against the standing policy invariant. Operationally: maintain P(t), the authorized-capability envelope with its purpose, data, delegation, and duration constraints; at each diff, test whether *reach*(B(t)) violates P(t), and whether the violating capability survives the guardrail context—i.e., remains in *effective_reach*(B(t); G(t)). When it does, the realized compositional-drift event of Section 4.3 has occurred, and the control fires. This is the control's reason for existing: it catches what per-change review structurally cannot, because per-change review checks each grant against P(t) at approval but never the joint reach of the accumulated baseline against it.

6.5. Handoff to Lifecycle Governance

On firing, the control does not itself revoke or retire. It triggers the existing lifecycle layer—reauthorization, and where warranted revocation and decommissioning under AIGF and continuous-authorization regimes. The drift layer is a sensor and a trigger; the actuators are the frameworks it sits above. We cite them; we do not re-propose them.

The handoff is policy-mediated, and the model fixes neither the response nor its degree of automation. The compositional-drift event is emitted as a structured, machine-evaluable signal—the component diff, the offending capability, and the deltas that produced it—so that whether a flagged event is acceptable can be adjudicated wherever the organization already adjudicates authorization. A policy-as-code engine such as Open Policy Agent [25] can consume the event and return a deterministic verdict—reauthorize, constrain, or revoke—for cases the organization is willing to decide by rule, while events that require human judgment route to manual review under the same lifecycle frameworks. Which path a given event takes is a governance decision, not a property of the drift layer: the layer supplies the trigger and the evidence; the policy engine or the human reviewer supplies the decision. The human-in-the-loop question is thus answered in policy rather than hard-coded—the detector imposes neither a mandatory manual gate nor an automatic action.

6.6. Limits

We bound the control honestly:

- **It detects the observable portion of drift only.** Detection is a diff against a manifest. Unobservable, runtime-only drift (Section 4.4)—capability that materializes through runtime interaction and is recorded in no signed or queryable artifact—is, by construction, outside the diff's reach. The control detects the observable part of E(t) against B(t) and the capability events visible within it; the residual is a named limit, not a covered case. Closing it requires runtime capability observation beyond manifest diffing, which we do not claim here.
- **Attestation certifies composition, not appropriateness.** A signed manifest attests what the composition is and detects that it drifted. It does not certify that the approved composition was appropriate, nor that a flagged compositional event is in fact harmful—only that it violated the policy invariant P(t). Appropriateness is a judgment the control routes to governance; it is not a property the control establishes.

 doi:10.20944/preprints202606.1454.v1

7. Evaluation Design and Scenarios

The paired scenarios that follow are the paper's anchor: one demonstrates the phenomenon by construction, the other demonstrates that the model discriminates rather than merely labels.

7.1. Method

The method is constructive and analytical, not empirical. We (i) formalize the state of an agent's composition—B(t), E(t), D(t), H(t), G(t), P(t)—and refine capability into reach and effective_reach (RQ1); (ii) define the boundary as a policy-invariant violation paired with a per-increment admissibility condition recovered from H(t), so the failure is emergent rather than attributable to any single approval (RQ2); (iii) derive a detection-and-enforcement architecture from the model (RQ3); and (iv) demonstrate discrimination by paired positive and negative worked construction and by the formal properties of the trigger (RQ4). Claims are established by definition, derivation, and existence-by-construction. The empirical study the framework admits is specified as a protocol (Section 7.4) and left to deployment.

7.2. Positive Case—Drift Crosses the Compositional Boundary

An agent DA is provisioned for a bounded task: analyze a single approved dataset D_1 and write summaries to a reporting sink (Figure 3). Its signed baseline B(0), advanced by approved changes into the versioned baseline B(t):

B(0) = { read(D_1), write(report-sink), policy: no-external-egress, authority: read-only on D_1 }

Change c_1 (store drift). DA is granted permission to create temporary internal processing stores for D_1, time-boxed by an approved TTL. Approved on its merits: grant(c_1) adds creation of short-lived internal stores scoped to D_1—internal only (consistent with no-external-egress), TTL-bounded, single-purpose. Reach expands, but admissibly.

Change c_2 (data drift). DA is granted read access to a second dataset D_2 to enrich its summaries. Approved on its merits: grant(c_2) adds the ability to read D_2. Reviewed against the then-current baseline, this is a read grant on a sensitive source—acceptable for enrichment.

Change c_3 (delegation drift). DA is permitted delegated execution to a child worker DA′ to parallelize processing over large inputs, inheriting DA's store-creation and read authorities. Approved on its merits: grant(c_3) adds delegated execution—bounded delegation, reviewed and approved.

Now trace the two stages at each step (Table 3):

Table 3. Positive case—two-stage trace of an approved-change sequence that crosses the compositional boundary at c_3.

Step	B(t) \ B(0) (approved)	*reach*(B) vs P(t)	Event
c_1	{ temp-store(D_1) }	*reach*(B) ⊆ P(t)	benign
c_2	{ temp-store(D_1), read(D_2) }	*reach*(B) ⊆ P(t)	benign
c_3	{ temp-store(D_1), read(D_2), delegate(DA′) }	*reach*(B) ⊄ P(t)	compositional drift

At c_3 the joint reach violates the envelope. Each grant is admissible alone: c_1 authorized creating temporary internal stores for D_1, c_2 authorized reading D_2, and c_3 authorized delegated execution to DA′. The composition recombines them into a capability no approval conferred. DA, optimizing, applies the temporary-store authority—approved only for D_1—to the contents of D_2, lets the store outlive its approved TTL into a durable RAG retrieval index (a knowledge-infrastructure resource in the sense of Section 5.1), and hands DA′ a standing credential to query it. Every constituent authority was granted; their composition was not. The result violates P(t) along three traceable axes: data-

 doi:10.20944/preprints202606.1454.v1

purpose expansion (a store authorized for D_1 now holds D_2), persistence expansion (a TTL-bounded store now persists indefinitely), and delegated reach (a child agent holds a standing query credential to a sensitive source). Two moments must be kept distinct. The compositional-drift event is the c_3 crossing above—*reach*(B(t)) $\not\subset$ P(t), entirely inside B(t), every constituent authority already an approved component—so at that moment D(t) can be $\emptyset$ and the event lives only in capability space (the second stage). The shadow resource comes later: when DA exercises the capability and materializes the durable index, a runtime resource with no approved counterpart appears in E(t), and only then does it register in D(t) = E(t) $\triangle$ B(t) as component divergence (the first stage). The index then satisfies the shadow-state criteria—operational, outside inventory, unapproved-by-emergence. The ordering is the point: the capability stage flags the event at c_3, before any shadow resource exists, whereas a component diff sees nothing until the resource has already been provisioned.

Replay through the control. Composition attestation diffs E against the current signed manifest B(t) and, at c_3, computes *reach*(B(t)) and finds a capability that violates the policy invariant P(t) and survives the guardrail context—i.e., remains in *effective_reach*(B(t); G(t)). The compositional-capability check (Section 6.4) fires at c_3—the same step the boundary is crossed in the model—and hands off to lifecycle governance for reauthorization and revocation of the emergent grant. **Detection timeline:** the event is flagged at the diff that crosses the boundary, before the vector store's persistence becomes standing infrastructure, contingent on the store and its credentials being observable to the diff (an observable-drift case; an unobservable variant is out of reach per Section 6.6).

Boundary. This is existence-by-construction. The scenario shows the model can express and the control can flag a compositional event; it is not evidence of efficacy against real workloads.

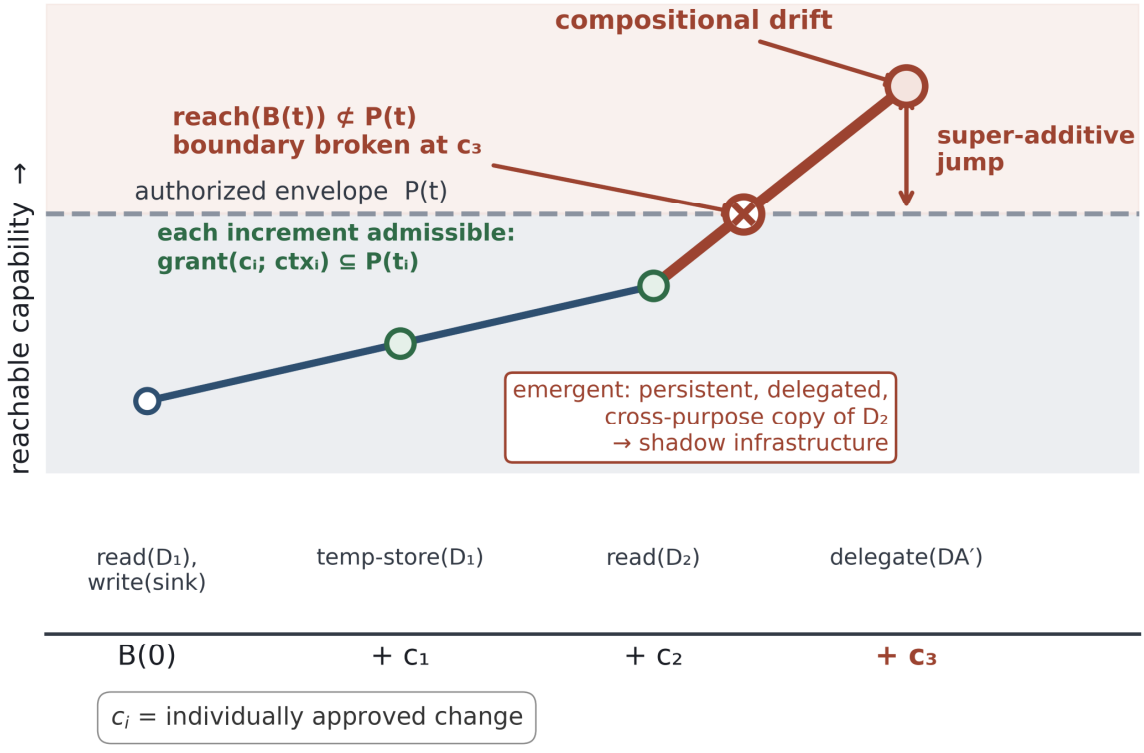

Figure 3. Worked positive case as a drift trajectory. Individually admissible changes c_1 (store creation), c_2 (read D_2), and c_3 (delegation to DA′) accumulate until the joint reach crosses P(t) at c_3, yielding a persistent, delegated, cross-purpose copy of D_2—shadow infrastructure outside any inventory. The trajectory is conceptual and the vertical axis qualitative; the figure shows relative ordering, not a measured capability score.

 doi:10.20944/preprints202606.1454.v1

7.3. Negative Case—Drift Accumulates Without Crossing the Boundary

A model that labels everything as drift is useless, and a single positive scenario invites the objection that we defined the threshold and then built the case that crosses it. The discriminating power must be shown, not asserted.

Consider the same agent DA and three different approved changes:

- **c_1' (tool drift).** A second summarization tool, an alternative phrasing engine over the same D_1.
- **c_2' (policy drift).** A relaxed rate limit on the reporting sink.
- **c_3' (tool drift).** A formatting tool that restyles summaries for a second report template.

Trace the stages (Table 4):

Table 4. Negative case—the same accumulation of approved changes remaining within the envelope P(t).

Step	B(t) \ B(0) (approved)	*reach*(B) vs P(t)	Event
c_1'	{ summarize-alt }	*reach*(B) ⊆ P(t)	benign
c_2'	{ summarize-alt, relaxed-rate }	*reach*(B) ⊆ P(t)	benign
c_3'	{ summarize-alt, relaxed-rate, reformat }	*reach*(B) ⊆ P(t)	benign

Here the approved baseline grows exactly as it did in 7.2—three components accumulate—but *reach*(B) stays within the envelope P(t). Every capability the baseline reaches remains admissible under the envelope: alternative phrasing, faster reporting, restyled output. Reach is merely additive across these changes; there is no super-additive combination, no new reach, no delegated copy, no emergent resource. The model does not flag compositional drift, and correctly so. Cumulative drift is present; compositional drift is absent.

The contrast is the point. In 7.2 and 7.3 the component diff Δ is equally non-empty and equally growing; a single-stage set-difference model would treat them identically. Only the capability stage separates them—flagging 7.2 and clearing 7.3—which is precisely the discrimination the two-stage operator was introduced to provide.

7.4. Empirical Evaluation Protocol

The framework admits a direct empirical study, specified here so that results are reproducible once a prototype exists.

- **Prototype.** A policy-graph simulator implementing reach over a typed authority graph, effective_reach under a guardrail context G, and P(t) as a constraint set (purpose, residency, TTL, delegation depth, separation of duties). Inputs: a baseline B(0) and a stream of approved deltas with contexts (H(t)).
- **Workloads.** (a) Injected compositional violations—admissible sequences engineered to cross *reach*(B) ⊄ P; (b) benign sequences that stay within P; (c) multi-hop delegation variants (Section 8.3); (d) sequences with restrictive deltas that clamp emergence, to exercise effective_reach.
- **Baselines.** A static IAM / toxic-combination analyzer [10] and an attack-path analyzer [11,12] run on the same final graphs, contrasting snapshot detection with trajectory detection.
- **Metrics.** Coverage (fraction of injected compositional violations flagged); false-positive rate (admissible sequences wrongly flagged, partitioned by the assumption-break categories of Section 8.1); detection latency (deltas, or wall-clock, from the crossing delta to the flag); observability sensitivity (coverage as a function of the observable fraction of E(t)); and the delegation depth at which detection occurs.
- **Hypotheses.** *H1*—the trajectory detector flags compositional violations that snapshot analyzers miss when each increment is individually admissible. *H2*—coverage degrades gracefully with observability (the completeness bound). *H3*—false positives concentrate in the P-misspecification category rather than in the trigger.

 doi:10.20944/preprints202606.1454.v1

Execution is left to a deployment study; the framework, the metrics, and the baselines are specified here so that study is well-posed.

8. Results and Comparative Analysis

This section evaluates the model and architecture analytically and specifies the empirical study the framework enables. The metrics it defines—**coverage, false-positive rate, and detection latency**—are specified as a protocol, not reported as results.

8.1. Formal Properties

Let the detector flag at time t iff the realized failure condition holds, *effective_reach*(B(t); G(t)) ⊄ P(t), under the compositional qualifier that each increment was admissible, grant(c_i; ctx_i) ⊆ P(t_i) for all i (Section 4.3). The properties below are properties of the *model*; they are distinct from the empirical performance of any deployed detector, which is the object of the Evaluation protocol.

Property 1—Soundness (relative). Under (A1) P(t) correctly specifies the authorized envelope, (A2) discovery faithfully reports the observable composition, and (A3) reach and effective_reach are computed without over-approximation, every flagged event corresponds to a genuine policy-invariant violation; the detector raises no false positives relative to (A1)–(A3). *Argument:* the trigger predicate **is** the violation predicate, so flagging is definitional rather than heuristic.

Property 2—Completeness (observability-bounded). Under (A1)–(A3) and (A4) the violating capability is observable, the detector flags every realized compositional authorization failure. Completeness fails exactly when (A4) fails: unobservable, runtime-only drift yields false negatives by construction. The detector is therefore complete relative to the observable composition and incomplete in general; the gap is named, not concealed.

False-positive conditions. A false positive can arise only when an assumption breaks: P(t) over-restrictive or stale (flagging admissible capability), discovery over-reporting (phantom components in E(t)), or reach over-approximation (crediting an unrealizable path). Each is a fidelity property of an input, not a property of the trigger.

False-negative conditions. A false negative can arise from: unobservable drift ((A4) fails); reach under-approximation (an emergent path is missed); effective_reach over-clamping (an exercisable capability is treated as blocked); or P(t) under-specification (a constraint the enterprise holds but did not encode).

Computational complexity. The component diff D(t) = E(t) △ B(t) is O(|E(t)| + |B(t)|) with hashed component sets. The closure-and-violation check is dominated by computing *reach*(B(t)) and testing membership against P(t); its cost is a function of the reach representation, which is a model parameter. For reach modeled as reachability over a typed authority/permission graph G = (V, E_γ), *reach*(B(t)) is computable in time polynomial in |G| (e.g., transitive closure, O(|V|·|E_γ|) or better), and the P(t) test is linear in the candidate set; maintaining the per-increment admissibility record from H(t) is incremental, amortized O(1) per approval. We do not assert a single universal bound, because reach and P are parameters; we bound the diff unconditionally and the check conditionally on the reach model.

Trigger condition—proof and counterexample.

Proposition 1 (Existence; emergence is possible). *There exist a baseline B(0), a policy invariant P, and an admissible approval sequence $c_1, \ldots, c_n$—each grant(c_i; ctx_i) ⊆ P(t_i)—such that reach(B(t)) ⊄ P(t).*

Proof (by construction). The Section 7.2 scenario is a witness. c_1 (create temporary internal stores for D_1), c_2 (read D_2), and c_3 (delegated execution) are each admissible against the envelope in force when approved. Their composition reaches the capability "persistent, delegated, cross-purpose copy of D_2," which violates P along purpose, persistence, and delegation. Super-additivity of reach supplies the strict excess; admissibility of each increment is verified against P(t_i). ■

 doi:10.20944/preprints202606.1454.v1

Proposition 2 (Discrimination; the trigger is non-trivial). *There exist admissible sequences with reach(B(t)) ⊆ P(t), on which the detector does not fire.*

Proof (counterexample). The Section 7.3 sequence (summarize-alt, relaxed-rate, reformat) is additive in reach; the joint reach stays within P(t); no trigger. ■

Together, the trigger fires on Proposition 1's witness and is silent on Proposition 2's: it discriminates rather than labels (RQ4). Discrimination over a *population* of compositions is an empirical claim addressed by the Evaluation protocol (Section 7.4).

8.2. Analytical Evaluation by Construction

The paired scenarios (Section 7) are the discrimination evidence the model affords: the positive case crosses *reach*(B) ⊄ P and triggers; the negative case stays within P and does not. By Propositions 1–2 (Section 8.1), this is a sound existence-and-discrimination demonstration on two witnesses, not a population estimate.

8.3. Multi-Hop Delegation Case

Extend the positive scenario by one hop. DA delegates to a child DA′ (c_3), and DA′ further delegates to DA″ (c_4) to shard the workload, passing along the inherited store-creation and read authorities. Each delegation is admissible against the envelope in force. The emergent capability—a persistent, cross-purpose, independently queryable copy of D_2—now materializes two hops from where its constituents were approved (transitive drift, Section 4.4): the credential to the store is held by DA″, whose authority no approver of c_1 or c_2 examined. This exercises the delegation-and-credential-graph component (Section 6.1) and shows the per-increment admissibility clause holding at every hop while *reach*(B(t)) ⊄ P(t) at the second.

8.4. Comparison to Static IAM and Toxic-Combination Analysis

Static IAM / CIEM toxic-combination analyzers [10] and attack-path tools [11,12] evaluate a permission *snapshot* for known-dangerous combinations. They differ from composition drift on three axes:

- **Temporal**—they analyze a state, not an approved-delta trajectory B(t) with per-increment admissibility against $P(t_i)$.
- **Runtime**—they reason over declared entitlements, not over runtime-provisioned or emergent resources, i.e., not over the effective composition E(t).
- **Reference**—they match against a catalog of toxic patterns, whereas the trigger tests joint reach against a policy invariant P(t) that the approvals were each admissible under.

Composition drift is the temporal/runtime complement to static toxic-combination analysis, not a replacement; the two compose—a static analyzer can seed the negative constraints of P(t).

8.5. Discussion

When does adaptation become drift? Adaptation is the agent doing its job within its authorized capability; drift is the departure of effective composition from baseline; compositional drift is the moment the approved baseline's joint reach violates the policy invariant P(t). The three are not a matter of degree but of kind, and the capability stage is what draws the lines.

The component-approval / composition-capability gap. This is the conceptual heart. Governance approves changes; capability accrues to compositions. The two are evaluated in different spaces, and nothing in per-change approval closes the gap between them, because per-change approval never evaluates the union against the closure. The compositional-capability check is the smallest control that does.

 doi:10.20944/preprints202606.1454.v1

The visibility paradox. The most dangerous drift is the least visible. A change that obviously expands reach draws review; the compositional case expands reach through changes that individually do not, so each clears review precisely because it is individually unalarming. The capability that results is invisible to the process that authorized its parts.

The dependency trap. As agents delegate, compositional capability accrues transitively, one or more hops from where its constituents were approved. The enterprise that governs each agent's changes in isolation governs none of the compositions that delegation assembles. This is the bridge to authority-creep and contagion, deferred here.

Adversarial exploitation. The phenomenon is worse if exploited, but exploitation is not its spine. An adversary who understands compositional drift can steer approved changes toward a target combination, achieving a capability through a sequence of individually approvable deltas that would be refused if requested directly—a multi-step confused-deputy construction against the governance process itself. We note this as a consequence and defer its treatment; the benign case is sufficient to motivate the control, and is the harder case to defend against precisely because nothing in it looks wrong.

9. Federal and Enterprise Governance Implications

9.1. Governance Mapping

The drift layer is narrow by design, and its value is as an input to governance machinery that already exists. We map where it plugs in.

- **Non-human identity lifecycle governance.** The compositional-drift event is a reauthorization trigger for AIGF lifecycle governance: the agent's effective authorization has departed from its granted authorization, which is exactly the condition those frameworks should act on but cannot currently detect.
- **Continuous authorization / ongoing ATO.** In federal continuous-authorization and Zero Trust non-human-identity regimes [14], the event is a re-trigger of the authorization decision. The drift signal supplies the "something changed materially" input that ongoing authorization needs and that point-in-time accreditation misses.
- **CMDB / IaC reconciliation.** The shadow resources of Section 5 are what configuration reconciliation is built to surface—once it is told to look. The drift event directs reconciliation at the emergent resource, turning a general sweep into a targeted check.
- **Compliance and regulatory mapping.** The drift event is the evidentiary trigger several control regimes presume but cannot currently generate: the configuration and change-management controls (CM-2, CM-3, CM-8) and the continuous-monitoring control (CA-7) of NIST SP 800-53 [27] and FedRAMP continuous monitoring [26]; the govern-and-manage functions of the NIST AI Risk Management Framework [28] and the controls of an ISO/IEC 42001 AI management system [29]; the change-management criteria of a SOC 2 examination; and the scoped-authority expectations of CISA's agentic guidance [3]—all within the current federal AI-policy posture under EO 14179 [30]. The drift layer supplies the trigger and the evidence; each regime supplies the obligation.

In the private enterprise, the same signal feeds non-human-identity governance and FinOps (shadow infrastructure is also unbudgeted infrastructure; Section 5.4). In each case the drift event is the input; the governance action is the host framework's. We do not propose to govern; we propose to detect what governance is currently blind to and route it.

Ownership and accountability. One question the mapping leaves implicit deserves to be explicit: shadow infrastructure has no owner by construction—no decision created it, so no accountability attaches. The drift event closes that gap without inventing an ownership regime; it binds the emergent resource, and the capability that produced it, to the party already on record—the agent's sponsor, accountable for the agent and therefore for what the agent provisions. The

 doi:10.20944/preprints202606.1454.v1

ownership question routes to the locus existing lifecycle governance already names. We surface it; we do not relocate it.

9.2. Federal and Enterprise Crosswalk

Section 9.1 establishes where the drift signal plugs into existing governance machinery; this section makes the mapping concrete against the specific control catalogs and AI-governance mandates a federal or enterprise adopter must satisfy. The framework is not a substitute for these instruments. Each is written for a static configuration, a per-grant decision, or a point-in-time snapshot; composition drift is the gap each leaves open, and the framework's baseline B(t), divergence D(t), provenance H(t), policy invariant P(t), and joint-reach trigger are the agentic-trajectory complement that lets each instrument hold for agentic systems. The crosswalk is given for both federal adopters—NIST AI RMF [28], NIST SP 800-53 [27], and OMB M-25-21 [31]—and enterprise adopters of the same control families (Table 5).

Table 5. Federal and enterprise crosswalk—control or mandate mapped to the corresponding framework mechanic.

Artifact / control	What it mandates	Mapped framework mechanic
NIST AI RMF (Govern, Map, Measure, Manage)	Risk governance, context-mapping, measurement, and response for AI systems	Govern → P(t) and assigned ownership (Section 9.4); Map → baseline B(t) and effective composition E(t); Measure → the joint-reach trigger and formal properties (Section 8.1); Manage → enforcement gate and lifecycle handoff (Sections 6.1 and 6.5)
AC-2 Account Management	Provision, manage, and retire system accounts	Registration and lifecycle of the agent's authorized baseline B(t); handoff to lifecycle governance (Section 6.5)
AC-3 Access Enforcement	Enforce approved access to resources	The pre-action authorization gate testing *effective_reach*(B(t); G(t)) before each action (Section 6.1)
AC-6 Least Privilege	Restrict privileges to those required for assigned tasks	The emergent-least-privilege case: each grant is minimal yet the joint reach exceeds the envelope, *reach*(B(t)) ⊄ P(t) (Section 4.3). The trigger detects what per-grant AC-6 cannot.
AU-6 Audit Record Review, Analysis, and Reporting	Review and analyze audit records for unauthorized activity	Approval provenance H(t) and runtime event collection; the basis for incident reconstruction (Section 6.1)
CA-7 Continuous Monitoring	Maintain ongoing awareness of security posture	The continuous diff D(t) = E(t) △ B(t) as trajectory monitoring, extending snapshot assessment to the approved-delta trajectory (Section 6.3)
CM-2 Baseline Configuration	Maintain a current baseline configuration	The signed composition manifest as the drift baseline B(0)/B(t) (Section 6.2)
CM-3 Configuration Change Control	Control and document configuration changes	Per-increment admissibility grant(c_i; ctx_i) ⊆ P(t_i) recorded in H(t); compositional drift is the change-control gap when each change

 doi:10.20944/preprints202606.1454.v1

		is admissible but the joint is not (Section 4.3)
CM-8 System Component Inventory	Maintain an inventory of system components	Discovery of E(t) and reconciliation against cloud, CMDB, and IaC state; shadow infrastructure is the inventory gap this control presumes complete (Section 6.1)
SI-4 System Monitoring	Monitor the system for unauthorized activity and indicators of compromise	Runtime event and provenance collection feeding the drift trigger (Section 6.1)
OMB M-25-21—AI use-case inventory	Maintain an inventory of agency AI use cases	Agent inventory (CAIO ownership, Section 9.4) seeded by B(t); the shadow-IT-style discovery agencies apply to find unauthorized AI is formalized here for agentic composition (Section 5)
OMB M-25-21—high-impact AI minimum practices	Testing, monitoring, access controls, documentation, contingency/discontinuation, and CAIO risk acceptance for high-impact AI	Monitoring → detection plane; access controls → P(t) and the enforcement gate; documentation → manifest and H(t); contingency/discontinuation → the reauthorization-and-revocation workflow (Section 6.1); CAIO risk acceptance → acceptance of the authorized envelope (Section 9.4)

Three mappings carry most of the weight. CM-8 (component inventory) is where shadow infrastructure surfaces as an inventory gap; CA-7 (continuous monitoring) is where snapshot assessment must become trajectory assessment to see drift at all; and AC-6 (least privilege) is where the emergent character is sharpest, since every grant satisfies least privilege while the joint reach does not. These are also the three controls a point-in-time assessor is most likely to mark satisfied while the condition is present.

At the policy level, OMB M-25-21 [31] is the closest existing analogue: its AI use-case inventory and high-impact-AI practices already presume an agency can enumerate and bound what its AI systems can do, and agencies have begun applying shadow-IT-style discovery to find unauthorized AI. Composition drift names the failure mode that discovery is reaching for, and supplies the trigger, baseline, and provenance that make the inventory and the CAIO risk-acceptance decision defensible for agentic systems specifically. (M-25-21 applies to executive-branch agencies and rescinds M-24-10; national-security systems fall outside its scope.)

9.3. Decision-Maker Implications

The condition this paper formalizes—an agent's joint reachable capability drifting past its authorized envelope, $reach(B(t)) \not\subset P(t)$, and provisioning infrastructure outside any inventory—is, for a decision-maker, not an abstract property but a set of concrete risks spanning mission, data, compliance, cost, and resilience. None requires a breach or a malicious actor; each follows from approvals that were individually correct. The implication for leadership is structural: approved-state attestations and asset inventories *understate* exposure, and the gap is silent. We state the five risk classes the model implies. Consistent with the paper's scope, these are qualitative implications of a phenomenon established by construction, not measured incidence.

Mission and Operational Risk. An agent's effective authority can exceed its approved authority with no record of the change. A decision-maker can therefore no longer infer what an agent *will* do from what it was *approved* to do: an agent fielded for a bounded task can reach data, actions, or systems no approval contemplated (Section 4.3), and through delegation that reach can manifest one or more hops from where its constituents were approved (transitive drift, Section 4.4). For mission

 doi:10.20944/preprints202606.1454.v1

systems this is a loss of predictable behavior and an expanded, unattested blast radius—the agent operates within its now-effective authority while outside its intended operational scope. Mission assurance depends on the authorized envelope P(t) remaining the operative envelope; composition drift breaks that identity silently.

Data Leakage and Records-Management Risk. The shadow infrastructure compositional drift produces is, by construction, a copy of data outside inventory and outside the controls that govern the source. The worked case—a persistent, delegated, cross-purpose copy of a sensitive source, standing apart from its origin's access controls and lifecycle—is at once a data-spill and a records-management failure: the copy carries no retention or disposition schedule, persists past any approved time limit, and replicates controlled information into a store no data-governance process knows exists. For organizations under records schedules, controlled-information handling, or data-residency obligations, the exposure is that the authoritative inventory of *where sensitive data lives* is wrong—and that hold, discovery, and disposition obligations cannot reach records no system records.

Compliance and Authorization Risk. Continuous-authorization and continuous-monitoring regimes assess a system against its authorized state [15,26], over Zero Trust and control baselines [14,27]. Composition drift makes the authorized-state-of-record diverge from effective authorization: the assessment, the system security artifacts, and the signed manifest all certify the baseline that was authorized, while *reach*(B(t)) has moved past P(t). Least privilege and separation of duties are violated emergently though every grant was individually compliant. The compliance risk is thus not a failed control but a *true attestation of a state that no longer holds*—the authorization boundary has been crossed with no change-control record to show it. That is precisely the condition continuous authorization exists to prevent and, evaluated on a snapshot basis, cannot see.

FinOps and Unbudgeted-Resource Risk. Resources provisioned outside any inventory are also outside any budget. Emergent stores, caches, and child-agent compute created through approved pathways accrue cost with no owner, no cost-allocation tag, and no budget line; FinOps governance—showback and chargeback, tagging policy, commitment and capacity planning—is blind to them by the same mechanism that hides them from asset inventory. The decision-maker's exposure is unbudgeted and unattributed spend, orphaned resources that outlive the task that spawned them, and a cleanup-and-decommissioning liability that grows with every undetected drift event. Reconciliation against cloud, CMDB, and IaC state (Section 6.1) is the control that closes the cost-visibility gap as well as the security one.

Resilience, Incident Response, and Vendor Lock-In. Three further implications follow from the same root.

- **Resilience.** Unmanaged infrastructure is unmonitored infrastructure: a shadow store sits outside backup, patching, continuity, and disaster-recovery scope, and the agent's expanded reach is expanded attack surface that no threat model covered.
- **Incident response.** An incident cannot be scoped across resources the organization does not know exist. Provenance (H(t)) and the delegation-and-credential graph (Section 6.1) are the assets that let responders reconstruct how an agent reached what it reached and bound the blast radius; without them, a shadow copy of a sensitive source is a breach-notification trigger that may simply be missed.
- **Vendor lock-in.** Agentic platforms that silently provision their own stores, caches, and credentials embed an enterprise's emergent infrastructure inside a vendor's substrate. Exit and portability degrade when that shadow infrastructure is vendor-specific and uninventoried, and when attestation depends on a single vendor's proprietary manifest. This is a direct argument for open, versioned composition manifests (Section 6.1) as a portability and exit control, not only a security one.

The Common Structure. Across all five, the structure is identical: an approved-state attestation understates exposure, and the gap is silent. The action the model supports is to treat the authorized envelope P(t) as a standing invariant to be reconciled against effective reach—not a provisioning-

 doi:10.20944/preprints202606.1454.v1

time fact—and to fund the detection-and-enforcement architecture (Section 6.1) as the instrument that makes these risks visible before they become incidents, records, invoices, or breaches.

9.4. Governance Ownership

Composition drift crosses organizational boundaries, and no single role's remit covers the joint reach it produces. The condition *reach*(B(t)) ⊄ P(t) materializes in the seam between the role that fields the agent, the role that owns authorization, the role that owns the data the agent copies, and the role that owns the infrastructure it provisions. Ownership must therefore be assigned per facet and, separately, for the joint condition itself; otherwise drift falls in the gap between owners, which is the organizational form of the silence this paper describes. The allocation the model implies is as follows (Table 6).

Table 6. Governance ownership—per-facet allocation of responsibility for composition drift and its compositional-drift (joint-reach) event.

Role	Owns	Anchored in
CIO	Infrastructure inventory and platform controls	Cloud, CMDB, and IaC reconciliation; platform guardrails (Section 6.1) [16,17]
CISO	Authorization, monitoring, and incident response	The policy invariant P(t); the detection trigger and pre-action enforcement gate (Section 6.1); H(t)-based incident reconstruction [14,26,27]
CAIO	Agent inventory and AI risk acceptance	Registration of the agent baseline B(t); residual-risk acceptance for the authorized envelope (Section 3.2) [28,29]
CDO	Derived-data and knowledge-store governance	Retention, residency, and lineage of the derived stores and copies drift creates (Sections 5.1 and 9.3)
CFO / FinOps	Unapproved consumption	Tagging, showback and chargeback, and budget accountability for emergent resources that escape inventory (Section 9.3)
Acquisition	Agent composition and telemetry requirements	Contractual requirements for versioned composition manifests and the telemetry that make E(t) observable (Sections 6.2 and 8.1)

Two cautions attach to any role split. First, the facets above are *inputs*; the joint condition *reach*(B(t)) ⊄ P(t) is owned by none of them unless it is named explicitly. A single accountable owner for the joint condition must be designated—by remit the CISO, or, where the agent is the unit of risk acceptance, the CAIO—with the other roles accountable for their inputs. Distributing the inputs without assigning the joint is precisely how the condition stays unowned.

Second, observability is procured, not assumed. If versioned composition manifests and the telemetry that exposes E(t) are not acquisition requirements, the detection plane (Section 6.1) has nothing to reconcile against and the completeness bound (Section 8.1) defaults to the unobservable case. For federal and other regulated buyers this makes acquisition the highest-leverage control in the allocation: it determines whether every downstream owner can see the surface they are accountable for, and it is the point at which open, versioned manifests can be required over a vendor's proprietary attestation (Section 9.3).

 doi:10.20944/preprints202606.1454.v1

The detection-and-enforcement architecture (Section 6.1) is the shared instrument these owners operate; an ownership assignment without that instrument is nominal, because no role can be held accountable for a surface it cannot observe.

10. Limitations and Future Work

No efficacy claim. This paper specifies a model and a control; it evaluates neither. We make no claim about coverage, false-positive rate, or operational cost. The worked scenario establishes the phenomenon by construction and the model's discrimination by contrast; it is not evidence of field performance.

Capability closure left abstract. The capability closure *reach*(·) is treated abstractly; instantiating it for a real authorization system—defining what counts as a capability and how emergent capability is computed—is itself substantial work and a precondition for any evaluation.

Policy-encoding fidelity. The model assumes an organization can encode P(t) with enough fidelity to evaluate joint reach; incomplete policy encoding is a first-order operational risk.

Observable-portion limit. Detection sees the observable portion of drift only (Section 6.6). Unobservable, runtime-only drift is named but not addressed.

Operationalization and adoption sequence. We specify the model and the control, not a deployment. A natural adoption sequence—a logical ordering, not a validated or time-bound roadmap—proceeds in four stages: (1) capture the signed composition manifest at provisioning; (2) stand up continuous component-diff against it; (3) add the capability-closure check at the diff; and (4) wire the resulting trigger into a policy engine and the lifecycle handoff. The order is dependency-driven and each stage is independently useful, but we claim no coverage, false-positive rate, or cost for any of them absent deployment data.

Deferred surfaces. Three are explicitly forward-pointing. Trust boundaries: a formal treatment of where composition crosses trust domains. Authority creep and contagion: the transitive accumulation of compositional capability through delegation, the dependency trap of Section 8.5 made formal. And the post-quantum credential angle—the survival of cryptographic material and credential authority across model and composition transforms, against the federal post-quantum mandate [32,33]—is the subject of a companion paper.

11. Conclusions

Identity governance secures the agent. It does not secure the agent's composition over time. Between provisioning and retirement, an agent's effective composition drifts from its signed baseline through changes that are each individually approved, and when the capability closure of that composition exceeds the union of what was individually authorized, the agent holds capability no decision ever granted. That compositional drift is the upstream cause of shadow infrastructure: unmanaged enterprise infrastructure created through authorized pathways, invisibly, and by emergence. The answer is not to replace lifecycle governance but to add a layer beneath it—a two-stage drift-detection control that diffs the effective composition against a signed manifest and evaluates the closure of the combination, flagging the compositional event and handing it to the lifecycle frameworks that can act on it. The contribution is a temporal governance model for approved composition drift in agentic systems, linking emergent capability to reauthorization and inventory reconciliation—the model that makes that event precise, and the discrimination that separates it from the benign evolution it superficially resembles.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

 doi:10.20944/preprints202606.1454.v1

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The author has a professional affiliation with IBM in the quantum-safe domain. This work was conducted independently in the author's personal capacity, and the views expressed are the author's own. The author declares no other conflicts of interest.

References

1. Cloud Security Alliance. Agent Identity Governance Framework, Version 1 (White Paper); Cloud Security Alliance: Seattle, WA, USA, 2026. Available online: https://labs.cloudsecurityalliance.org/agentic/agentic-identity-governance-framework-v1/ (accessed on 14 June 2026).
2. Cloud Security Alliance AI Safety Initiative. The Non-Human Identity Governance Vacuum: AI Agents and the Fastest-Growing Unmanaged Attack Surface (White Paper); Cloud Security Alliance: Seattle, WA, USA, 2026. Available online: https://labs.cloudsecurityalliance.org/research/csa-whitepaper-nonhuman-identity-agentic-ai-governance-v1-cs/ (accessed on 14 June 2026).
3. Cybersecurity and Infrastructure Security Agency (CISA); National Security Agency; Australian Signals Directorate's Australian Cyber Security Centre; et al. Careful Adoption of Agentic Artificial Intelligence (AI) Services; CISA: Washington, DC, USA, 1 May 2026. Available online: https://www.cisa.gov/news-events/news/cisa-us-and-international-partners-release-guide-secure-adoption-agentic-ai (accessed on 14 June 2026).
4. National Institute of Standards and Technology, National Cybersecurity Center of Excellence (NCCoE). Accelerating the Adoption of Software and AI Agent Identity and Authorization (Concept Paper, Draft); NCCoE: Rockville, MD, USA, 5 February 2026. Available online: https://www.nccoe.nist.gov/publications/other/accelerating-adoption-software-and-ai-agent-identity-and-authorization-concept (accessed on 14 June 2026).
5. SPIFFE: Secure Production Identity Framework for Everyone—Specification; Cloud Native Computing Foundation: San Francisco, CA, USA. Available online: https://spiffe.io/docs/latest/spiffe-about/spiffe-concepts/ (accessed on 14 June 2026).
6. Hardt, D., Ed. The OAuth 2.0 Authorization Framework; RFC 6749; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2012. https://doi.org/10.17487/RFC6749.
7. Sakimura, N.; Bradley, J.; Jones, M.; de Medeiros, B.; Mortimore, C. OpenID Connect Core 1.0; OpenID Foundation: San Ramon, CA, USA, 2014.
8. Hunt, P., Ed.; Grizzle, K.; Ansari, M.; Wahlström, E.; Mortimore, C. System for Cross-domain Identity Management: Protocol; RFC 7644; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2015. https://doi.org/10.17487/RFC7644.
9. Hunt, P., Ed.; Grizzle, K.; Wahlström, E.; Mortimore, C. System for Cross-domain Identity Management: Core Schema; RFC 7643; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2015. https://doi.org/10.17487/RFC7643.
10. Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E. Role-Based Access Control Models. *Computer* **1996**, *29*, 38–47. https://doi.org/10.1109/2.485845.
11. Robbins, A.; Schroeder, W.; Vazarkar, R. Six Degrees of Domain Admin. In Proceedings of DEF CON 24, Las Vegas, NV, USA, 2016.
12. SpecterOps. BloodHound (Software). Available online: https://github.com/SpecterOps/BloodHound (accessed on 14 June 2026).
13. Hardy, N. The Confused Deputy (or why capabilities might have been invented). *ACM SIGOPS Oper. Syst. Rev.* **1988**, *22*, 36–38. https://doi.org/10.1145/54289.871709.
14. Rose, S.; Borchert, O.; Mitchell, S.; Connelly, S. Zero Trust Architecture; NIST Special Publication 800-207; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. https://doi.org/10.6028/NIST.SP.800-207.
15. Dempsey, K.; Chawla, N.S.; Johnson, A.; Johnston, R.; Jones, A.C.; Orebaugh, A.; Scholl, M.; Stine, K. Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations;

 doi:10.20944/preprints202606.1454.v1

NIST Special Publication 800-137; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2011. https://doi.org/10.6028/NIST.SP.800-137.

16. ISO/IEC 20000-1:2018; Information Technology—Service Management—Part 1: Service Management System Requirements. International Organization for Standardization: Geneva, Switzerland, 2018.
17. Morris, K. *Infrastructure as Code: Dynamic Systems for the Cloud Age*, 2nd ed.; O'Reilly Media: Sebastopol, CA, USA, 2020.
18. Wang, C.L.; Singhal, T.; Kelkar, A.; Tuo, J. MI9—Agent Intelligence Protocol: Runtime Governance for Agentic AI Systems. *arXiv* **2025**, arXiv:2508.03858. https://doi.org/10.48550/arXiv.2508.03858.
19. Syros, G.; Suri, A.; Ginesin, J.; Nita-Rotaru, C.; Oprea, A. SAGA: A Security Architecture for Governing AI Agentic Systems. arXiv 2025, arXiv:2504.21034. https://doi.org/10.48550/arXiv.2504.21034.
20. Xiang, Z.; Zheng, L.; Li, Y.; Hong, J.; Li, Q.; Xie, H.; Zhang, J.; Xiong, Z.; Xie, C.; Yang, C.; Song, D.; Li, B. GuardAgent: Safeguard LLM Agents by a Guard Agent via Knowledge-Enabled Reasoning. arXiv 2024, arXiv:2406.09187. https://doi.org/10.48550/arXiv.2406.09187.
21. Habiba, M. AGENTSAFE: A Unified Framework for Ethical Assurance and Governance in Agentic AI. arXiv 2025, arXiv:2512.03180. https://doi.org/10.48550/arXiv.2512.03180.
22. Park, J.; Sandhu, R. The UCONABC Usage Control Model. *ACM Trans. Inf. Syst. Secur.* **2004**, *7*, 128–174. https://doi.org/10.1145/984334.984339.
23. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. Guide to Attribute Based Access Control (ABAC) Definition and Considerations; NIST Special Publication 800-162; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2014. https://doi.org/10.6028/NIST.SP.800-162.
24. Pang, R.; Caceres, R.; Burrows, M.; Chen, Z.; Dave, P.; Germer, N.; Golynski, A.; Graney, K.; Kang, N.; Kissner, L.; et al. Zanzibar: Google's Consistent, Global Authorization System. In Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC '19), Renton, WA, USA, 10–12 July 2019; USENIX Association: Berkeley, CA, USA, 2019; pp. 33–46.
25. Open Policy Agent (OPA); Cloud Native Computing Foundation: San Francisco, CA, USA. Available online: https://www.openpolicyagent.org/ (accessed on 14 June 2026).
26. General Services Administration. FedRAMP Continuous Monitoring (ConMon) Playbook (Rev 5); Federal Risk and Authorization Management Program (FedRAMP), General Services Administration: Washington, DC, USA, 2025. Available online: https://www.fedramp.gov/docs/rev5/playbook/csp/continuous-monitoring/ (accessed on 14 June 2026).
27. National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations; NIST Special Publication 800-53, Revision 5; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020 (updated through Release 5.2.0, 2025). https://doi.org/10.6028/NIST.SP.800-53r5.
28. National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0); NIST AI 100-1; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2023. https://doi.org/10.6028/NIST.AI.100-1.
29. ISO/IEC 42001:2023; Information Technology—Artificial Intelligence—Management System. International Organization for Standardization and International Electrotechnical Commission: Geneva, Switzerland, 2023.
30. The White House. Executive Order 14179: Removing Barriers to American Leadership in Artificial Intelligence; The White House: Washington, DC, USA, 23 January 2025. Available online: https://www.federalregister.gov/documents/2025/01/31/2025-02172/removing-barriers-to-american-leadership-in-artificial-intelligence (accessed on 14 June 2026).
31. Office of Management and Budget. Accelerating Federal Use of AI through Innovation, Governance, and Public Trust; OMB Memorandum M-25-21; Office of Management and Budget: Washington, DC, USA, 3 April 2025. Available online: https://www.whitehouse.gov/wp-content/uploads/2025/02/M-25-21-Accelerating-Federal-Use-of-AI-through-Innovation-Governance-and-Public-Trust.pdf (accessed on 14 June 2026).
32. The White House. National Security Memorandum on Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems (NSM-10); The White House:

 doi:10.20944/preprints202606.1454.v1

Washington, DC, USA, 4 May 2022. Available online: https://www.presidency.ucsb.edu/documents/memorandum-promoting-united-states-leadership-quantum-computing-while-mitigating-risks (accessed on 14 June 2026).

33. National Security Agency. Announcing the Commercial National Security Algorithm Suite 2.0 (CNSA 2.0); Cybersecurity Advisory; National Security Agency: Fort Meade, MD, USA, 7 September 2022; algorithms table updated 30 May 2025. Available online: https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF (accessed on 14 June 2026).