International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 2018-2028
https://doi.org/10.70917/ijcisim-2026-5977
Article

Identity, Delegation, and Auditability for AI Agents with Production Access

Harsh Kaushikbhai Patel

Independent Researcher, USA
ORCID: 0009-0005-6840-1729

Abstract: Software agents based on large language models are also moving from programming proof-of-concept works to production work, reading telemetry, carrying out runbooks, mutating configuration, and calling infrastructure-level interfaces on behalf of engineering teams. The models used to govern these new agents were constructed for other actors. In contrast, while humans will be held accountable for their reviews, service accounts, assuming a known and fixed principal, can have their behaviors easily scoped ahead of time. The same cannot be said for agents. It is also broad in capability, nondeterministic, and the goals it learns are dependent on natural language inputs that an opponent might influence. Instead of inventing new solutions here, this work considers the exposure as a familiar problem at an infrastructure scale. In addition to permission scoping based on resources, this paper argues that such scoping is structurally insufficient, as all resources are involved in both trivial and catastrophic behavior. This paper also proposes bounded delegation, an agent credential that derives from the authority of the human actor, is scoped to the authority defined by the task, is valid for the duration of the task, and is gated based on the reversibility of the task rather than on which resource it touches. It explains how bounded delegation composes with production identity and authorization primitives that are in widespread use, defines autonomy retention, a metric for evaluating the compliance of other presence containment schemes, and defines an audit record of the actions of agents, allowing an agent's reasoning and authority to be reconstructed after the fact.

Keywords: Agentic Artificial Intelligence, Workload Identity, Delegated Authorization, Confused Deputy Problem, Least Privilege, Production Security, Audit Logging

1. Introduction

For two decades, production engineering teams have eliminated hand configurations from the deployment and remediation paths through the use of configuration management tools, and continuous delivery pipelines have replaced operators who manually typed deployment commands from a runbook. These auto-remediation systems absorbed a meaningful portion of on-call response by automatically restarting failed processes or rolling back bad releases before anyone even noticed. One thing that all of these systems had in common was that this behavior was largely hard-coded. The steps executed inside the pipeline are precisely the steps the programmer wrote it to execute; the remediation script will match some condition and execute one of a tiny set of actions. Because the space of possible behavior was enumerable, permissions could be restricted down to that exact space, and least privilege was a matter of engineering discipline applied to a known quantity [1].

Agentic automation lacks that property: a large language model-powered agent chooses its tool calls at inference time, choosing the order in which it strings its tool calls together in ways that no engineer designed ahead of time. It is not guided by a predetermined control flow but is instead expressed semantically in natural language. Relatedly, industry reporting throughout 2025 and into early 2026 has indicated that non-human identities, and agent identities within that category specifically, have grown far faster than the identity governance programs meant to govern them have [2]. In fact, the tokens given to these agents in most production deployments observed by security practitioners



Copyright: © 2026 by the authors

are the same class of standing service account credential that deterministic automation used for the prior decade, simply pointed at a less predictable consumer [3].

The conceptual problem is this: IAM systems authorize a request based on three aspects: the identity of the resource principal that requests to perform an operation, the resource on which the operation will be performed, and the operation itself. That assumption works when identity is a reasonably reliable signal of intent, which it has been for human operators of services and deterministic services, but it does not hold for an agent. An agent's operative intent exists in its context window. The context window is generally filled with content that the agent retrieved, rather than content that the human typed, such as logs, ticket bodies, monitoring payloads, documentation, and the results of other tool calls. Any part of the content the agent received could contain instructions not intended for the agent. This type of indirect prompt injection attack has been demonstrated against real deployed applications that integrate a language model with other tools and external data sources [4]. Prompt injection is the dominant risk pattern for such applications in Open Worldwide Application Security Project's agentic security guidance, while excessive tool permission and unconstrained action are classified as separate failure modes [5].

The consequence for infrastructure operations is less abstract: consider an agent that has production credentials and takes untrusted input as part of its normal operations. An attacker who can write text somewhere the agent will read it, e.g., a log line, support ticket, or wiki page, will then be able to get a privileged operation to execute under a legitimate identity. Normal authorization checks will not see this in action because there is nothing superficially abnormal about the request: an authorized principal invoked an authorized interface against a resource it was already permitted to touch. The evidence of what happened, recorded in the audit trail and read later, tells the same reassuring story. The model of a privileged deputy subverted by an unprivileged caller has been studied in an operating systems context for decades and was noted by Hardy as the confused deputy problem [6]. Recent work in multi-agent delegation security has framed agentic misuse as an over-trusted agent, a deputy that can be manipulated to work against the interest of the party it nominally serves [7]. Current infrastructure security practice lacks a mechanism for this pattern, as it grants permissions scoped to individual resources and uses standing credentials.

Accordingly, two primary questions structure the discussion ahead: first, why the two existing models of identity that govern human accounts (review accounts) and service accounts (static scope accounts) are both incapable of containing an agentic actor; and second, what property should replace resource as the primary unit along which agent permissions are scoped? Third, what an audit record needs to contain for an agent's action to be reconstructible after the fact, given that the reasoning behind the action is not something conventional request logging captures. This discussion is about infrastructure and production systems, not a consumer-facing assistant, so every idea described here should be buildable using primitives already deployed in production systems, rather than a new control plane.

First, the scale of the exposure is worth stating in concrete terms before proceeding further into this argument. It was noted in a 2026 non-human identity governance assessment that, in the surveyed environments, the ratio of non-human identities to human identities is between 144 to 1 and 45 to 1, and the year-over-year population trajectory of non-human identities is at a growth rate of 44% [2]. The same study reports that only 49% of organizations say there is clear ownership of AI-related identities, meaning that in half of the environments queried there may be no team that can say exactly what an agent identity is allowed to do [2]. Importantly, agentic identities have been consistently found to be the fastest-growing, least governed, segment within that broader non-human population, which is the population this article is concerned with.

2. Materials and Methods

Three categories of literature directly concern securing agents with production access. None of these, on their own, solve the problem this article addresses. Each is an essentially required component, with the gaps described above interspersed among them.

There is no vagueness about how infrastructural services operate; for example, an agent with production credentials consumes untrusted input during its operation. This situation means that anyone who can write to anything

that the agent can read, such as a log line, support ticket, or wiki page, could potentially request a privileged operation be performed under an identity that is not their own. Conventional authorization systems may not catch this scenario since there is nothing wrong with an authorized principal calling an authorized interface against a resource that they are entitled to touch. Symmetrically, reading the same audit trail in reverse encounters a problem with an operating systems research heritage, where a more privileged program is tricked into using that privilege to the detriment of its caller. Hardy called this phenomenon the confused deputy problem [7]. Applied multi-agent delegation security research casts agentic misuse as happening through a confused deputy: delegating authority to an over-trusted agent and having it act against the interest of the party it's nominally serving [8]. Existing practice in infrastructure security (mandating resource-scoped permissions and standing credentials) does not have an analog for this type of agentic misuse. Related empirical research observes agents using the ambiguity of the declared scope of authority to invoke other tools outside the task's scope, which is aggravated by the fact that the protocol must not have been designed with non-human callers in mind. [6]

These findings led to the main proposal that the scoping dimension should be changed from the resource that is being acted on to the reversibility of the action.

Workload identity standards solve the authentication half of the problem. The Secure Production Identity Framework for Everyone specification and its reference implementation issue short-lived, cryptographically attested identity documents to workloads at runtime. This eliminates long-lived static secrets and shared API keys with that same purpose [9]. This is a genuine improvement and a necessary precondition for anything proposed later in this essay. An agent cannot be governed by its task unless it can be identified as itself. Yet the current definition of workload identity answers the question of who is calling, but does not specify what actions that caller can take once authenticated. It establishes identity; it does not constrain action.

Delegated authorization standards go further, but only partway: the OAuth 2.0 Token Exchange specification allows for the exchange of one token for another that is constrained to the resource of a different audience or is granted a lesser set of permissions. It also provides a way to represent a chain of delegating actors inside the resulting token [10], related to what bounded delegation requires. The specification is very clear that a token consumer is not expected to inspect more than the top-level claims and the identity of the acting party at the time. In other words, in the whole delegation chain, the early parties are merely metadata that's useful for logging and auditing purposes rather than for an access control decision [10]. In other words, the delegation chain is an audit trail, not a permissions policy. The Internet Engineering Task Force (IETF) is currently standardizing workload identity schemes for multi-system architectures to address this issue for agentic call chains, with proposed applicability guidance for AI agent identity in that architecture [11], [12]. As of this writing, this work has not yet been settled and widely deployed.

In agentic threat modeling, the agent's attack surface is modeled instead of the control. A threat model for generative AI agents lists tool misuse, excessive agency, and cascading errors in control chains as failure modes. It further recommends fine-grained permissioning and real-time behavioral monitoring to reduce these failure modes [13]. For example, a threat-modeling project within industry distinguishes memory poisoning, tool misuse, and unconstrained autonomy as distinct threats and recommends confining privileges and monitoring ML behavior [5]. The MITRE ATLAS knowledge base of adversarial techniques against ML systems more generally discusses how an opponent may combine reconnaissance and manipulation techniques to attack a deployed ML model [14]. A broader discussion identifies the most common architectural failure mode of authority amplification, the unintended increase of an agent's effective privilege, across various language model agents [15]. These surveys are sound, and this paper will take them as the prima facie inventory of what can go wrong. What they leave underspecified is the dimension along which permissions should actually be narrowed. Recommending granular permissions without specifying granular along what axis leaves the practitioner in roughly the same place. Resource-based scoping already leaves them: an individual must enumerate individual API calls and configuration keys by hand as part of an access control policy that does not scale beyond a few services.

The second level of control includes reversible operations that can be applied automatically and undone in seconds, leaving behind no side effects: scaling a deployment up or down, changing a log verbosity level, draining a node before maintenance, and restarting a stateless process. All of these are equivalent, in the sense that they restore the system to some equivalent state, with no interesting lasting side effect, and I'm assuming for the rest of this article that they all proceed without a human at the wheel. The idea is that the blast radius of a mistake here is limited by the automated rollback, not by whoever happens to be watching when the agent does the action.

These operations are still considered to be recoverable but require time, multistage coordination between systems, or some other recovery mechanism to do so. Configuration changes that take some time to propagate through a fleet, deployments that must be rolled back to a previous release, and schema migrations with a defined rollback path are all examples of a recoverable operation. Both of these leave us with a window in which the system is in a state that has to be actively walked back to be able to return to its last safe state. In the model proposed in this article, both of these operations execute behind a human approval gate. The gate exists because the cost of rolling back the action is nontrivial, and a human rather than an automated process must enforce the tradeoff.

Other operations are by nature non-reversible: deleting data, rotating credentials in a way such that all sessions still known to an agent are invalidated, sending messages to outside entities, or triggering a financial transaction. According to the model in this article, these operations are structurally unavailable to an agent. The derived credential this agent holds lacks the required permission, and there is no gate to go through, because the agent has no path to that action. This is a stronger guarantee than a policy rule that could, in principle, be misconfigured or reasoned around.

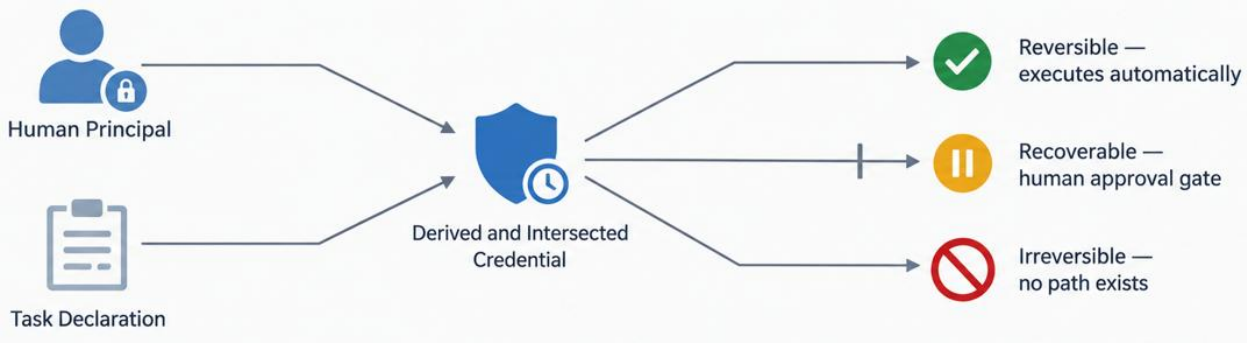


Figure 1. Credential derivation flow under bounded delegation, from human principal and task declaration through a derived and intersected credential to reversibility-gated execution.

Resource-based scoping does not work for agents because the operation consequences and the resource granularity are not related. For example, an agent can be given write access to update a shared configuration store. This matches the zero trust principle of per-request access control rather than network-based [16]. The zero trust argument has been generalized to non-human identities in other domains such as Internet of Things device fleets, but the agent's behaviors are less general than the arbitrary tasking vision for agents in IoT [17]. An example of a single write permission here would be changing a log verbosity level that can be trivially reverted and disabling an authentication requirement that cannot. Both are writes to the same resource, and a policy engine is applied that reasons about the resource and actions on it. While it could theoretically be restricted to specific configuration keys, at the scale of a real production fleet with thousands of keys across dozens of services, such an approach is not an operationally viable option. Nobody is maintaining that list by hand for long, and at some point the allowlist is either too permissive or entirely breaks automation, and neither is the goal.

These credentials are delegating, such that an agent's service credential is minted from the authority of the human who initiated the task (rather than handed directly to the agent as a separate identity). Resulting agents can never have more privileges than those of the principal on whose behalf they act, thus eliminating a general class of privilege escalation that a service account provisioned separately would be unable to eliminate by itself.

Credentials are intersected with the declared scope of the task or agent request. Therefore, the permissions granted to an agent are the intersection of what the principal performing the delegation is allowed to do and what the task permits. For example, an engineer with broad access to production who starts a task with a narrow scope should not give the agent broad access to production. Tasks don't simply inherit the principal's ceiling in this case.

The credential is scoped to the task, however, and is deleted when the task finishes or times out. In doing so, the credential is deleted, thus removing the standing, long-lived credential that makes an agent identity so attractive to an attacker in the first place. However, a credential that an attacker cannot reuse tomorrow is worth considerably less than one that never expires, which is precisely the failure pattern that standing service account credentials exhibit today [18].

Although the permission-intersection mechanism is not new, practitioner and research literature on zero trust identity for autonomous agents has only recently begun to emerge, including work on multi-agent delegation security [8]. One key difference, however, is that the reversibility gate can be layered on top of that pattern. Reversibility is the dual of resource identity, meaning that it serves as a complementary concept. The idea that security engineering cares about is that the only thing that matters when an agent misbehaves or exceeds its permission is its blast radius, i.e., the amount of damage that a single unauthorized action is able to cause before a human or other system has a chance to intervene. The theory upon which this compositional reasoning is based is capability-based access control. This upper bound on the effective power of a program corresponds to the amount of power that was explicitly passed to it and is similar to the confused deputy analysis above [7].

Such a containment scheme can be evaluated using a metric that punishes inappropriate over-restriction as well as inappropriate under-restriction. A model that gates all operations is safe but not practically useful. An important metric for this goal is defined here: the percentage of operations in a representative production workload that can be continued without human intervention after the model is deployed.

The evaluation process includes four steps: first, provide a corpus of operational tasks, based on the runbooks, remediation automation, and on-call procedures that already exist in operations, so the outcome reflects the real operational surface, not a hypothetical one. Then, for this corpus, classify each operation in the corpus into one of the three reversibility levels. This should be done by at least two independent engineers to enable inter-rater agreement. The main area of subjectivity in the model is where the recoverable tier and the irreversible tier are distinguished. Third, compute the retention of compute autonomy over the classified corpus, as the fraction of operations landing in the reversible tier plus the proportion of operations landing in the recoverable tier for that deployment. Fourth, for that deployment, compute the retention of compute autonomy over the incident record (that is, separately compute the fraction of incident-creating or incident-increasing operations that would have been gated or rendered unavailable per classification versus the historical incident record). These two numbers, read together, characterize the trade-off: the proportion of the history that the model would have stopped, and the amount of day-to-day freedom that was given up to get there. This procedure is designed to be constructable without any prior agent deployed in production, which is important in practice because most organizations analyzing these controls have not deployed agents at a scale that produces usable incident data [19].

Recent work on secret leaks provides an even stronger argument, regardless of the agentic angle. A survey of exposed credentials found that 64% of those exposed in 2022 were still exploitable four years later. This indicates that organizations are, in practice, quite slow to rotate or revoke credentials once they have leaked [18]. The same research suggests that exposed secrets in the context of AI-related services are exploding at 81% year over year, despite a lack of distinct identification of it as a line item in the last few years [18]. However, both these figures illustrate the standing-credential failure mode that bounded delegation is designed to prevent by construction, rather than by better hygiene practice.

3. Results

3.1 Reversibility as the Scoping Dimension

The analysis shows that reversibility, and not the identity of the resource, should govern the degree of autonomy granted to the agent for performing an operation. The point is clear when considering where operational risk really lies: two operations on the same resource can have wildly different results, while two operations on entirely different resources can have almost identical results. A policy language built around resources and actions cannot express this distinction without perhaps enumerating per-key exceptions, which does not scale either in size or in complexity to a fleet of nontrivial size. A policy language built around declared reversibility can express this, because reversibility is a property of the operation's effect, not of where the operation happens to be executed.

The three tiers and how they are managed are summarized in Table 1 (in the methods) and provide a framework for comparison in the remainder of the analysis.

Table 1. Three-tier reversibility taxonomy for scoping agent permissions.

Tier	Definition	Examples	Governance treatment
Reversible	Undone within seconds by an automated mechanism; no residual state.	Scaling a deployment; adjusting log level; draining a node; restarting a stateless process.	Executes autonomously.
Recoverable	Can be undone, but recovery requires time, coordination, or a separate system.	Configuration changes with propagation delay; deployments; schema migrations with rollback paths.	Executes with a human approval gate.
Irreversible	It cannot be meaningfully undone.	Data deletion, credential rotation that invalidates sessions, external communications, and financial transactions are all irreversible.	Structurally unavailable; the credential does not carry the permission.

3.2 The Bounded Delegation Model, Formalized

Bounded delegation combines the reversibility gate with the three credential-construction conditions described above in a single decision procedure. Bounded delegation permits a system to derive a credential that is scoped to the existing authority of a human operator performing a given task and constrained to the needs expressed by that task. The derived credential is usually for the expected lifetime of the task. All subsequent operations requested by the agent are first checked against the derived credential, then against its category of reversibility, and reversible operations within the credential's lifetime are run without being checked. While the scope does contain those recoverable operations, a human approval step must be completed to execute them, and the irreversible operations are rejected as the credential never had that permission in the first place.

This construction ensures that an agent with this credential cannot be socially engineered with adversarial content in its context window to perform any unauthorized operation. The authorization check is on a cryptographically scoped credential, not on the agent's impression of whether the instruction is legitimate. Prompt injection can influence what the agent actually attempts. However, it cannot expand what the agent is capable of successfully doing, which resource-scoped standing credentials do not provide. This is because, once given a standing credential, an attacker-influenced request has the same permissions as one authentically issued [4], [5]. But this formalization also highlights the model's limitations: it controls what an agent can do, but not what it can see or recommend. Preventing the dangerous action from being executed does not prevent the agent from recommending it. Reviewing agent-recommended actions, separate from the execution gate described here, is an important step that organizations still need to take.

3.3 Comparative Positioning

It is worthwhile to compare bounded delegation against the three approaches that organizations most commonly reach for today, since each is a real approach that organizations deploy.

Bounded delegation resolves the issue with active service account credentials that the persistent artifact is still useful to an attacker trying to exploit an agent because the credentials are still valid even after the agent's compromise. In practice such credentials are not useful to an opponent for very long after leak [18], and a derived, task-bound credential is due to expire in any case if the administrator fails to detect the leak, removing the incentive that motivates an opponent to expend resources to compromise the agents. It also avoids the kind of resource-level permission enumeration that makes traditional least privilege hard to achieve at fleet scale [24].

Bounded delegation thus provides a better preventive mitigation than most behavior-monitoring and anomaly detection approaches, such as the main mitigations recommended by some of the previous threat-modeling frameworks (e.g., [5], [13]), which are only able to detect the unusual behavior of agents after it has occurred. In the case of irreversible machines that operate at machine speed, the entire problem is detection latency, rather than just a parameter to tune. By the time the anomaly detector raises the alarm, the deletion has most likely already taken place. The monitoring system serves as a second line of defense and can be used for post-attack forensic reconstruction, but it should not be the first line of detection when the cost of a delayed response cannot be recovered.

Bounded delegation is the way to retain the efficiency gain that motivates agent use. Human review of all agent actions is often the first policy adopted by organizations that let agents operate in a production environment. However, this policy is generally abandoned quickly because engineers are asked to approve hundreds of such low-consequence prompts each day, and they tend to approve them reflexively, creating a false appearance of human oversight [19]. First, reserving the approval gate for the recoverable tier concentrates human attention on the decisions where that attention still carries meaning, rather than diluting it across a volume of trivial approvals that erode the reviewer's judgment over time [26].

Table 2 shows three of the most important dimensions for assessing production security: exposure to a persistent stolen credential, the axis on which permissions are scoped, and the burden to human reviewers.

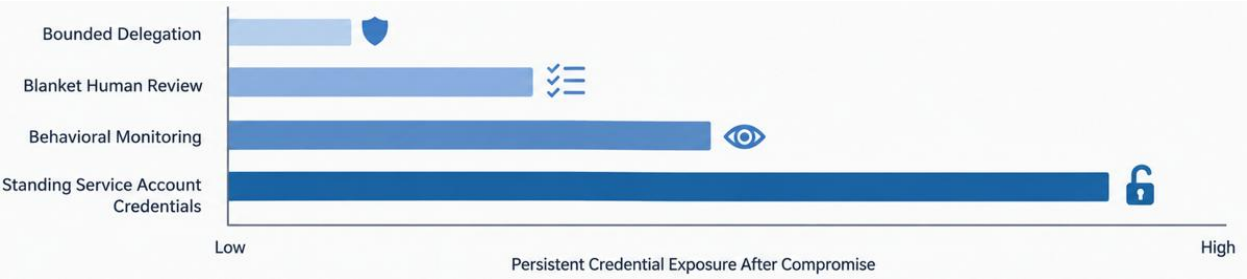


Figure 2. Relative persistent credential exposure across four containment approaches, illustrating why task-bound derivation reduces the standing attack surface compared with alternatives.

Table 2. Comparative positioning of bounded delegation against standing service accounts, behavioral monitoring, and blanket human review.

Model	Persistent credential exposure	Scoping dimension	Human review load
Standing service accounts	High; the credential remains valid indefinitely after compromise.	Resource and action, enumerated statically.	Low; approvals occur at provisioning time only.
Behavioral monitoring	High; the credential itself is unchanged, and detection is the only added control.	None directly; scoping stays resource-based, and monitoring is a separate layer.	Variable; concentrated on investigating flagged anomalies after the fact.

Blanket human review	Low to moderate; credential scope may still be broad, but every action is reviewed.	None; every action is gated regardless of resource or consequence.	Very high; tends to degrade into reflexive approval over time.
Bounded delegation	Low; the credential is derived, intersected, and expired with the task.	Reversibility of the operation.	Moderate; concentrated on the recoverable tier only.

4. Discussion

This approach is not uniformly applied over the domains where agents are now deployed against production systems, and the unevenness is itself informative as to where things should start.

The most immediate opportunity for incident response automation would be an agent triaging alerts and executing diagnostic runbooks. Because the operation mix heavily favors diagnostic reads and reversible mitigations, high autonomy retention is expected under this model. The recoverable-tier gate applies to the comparatively few active remediation steps beyond mere read-only diagnosis.

Continuous integration and delivery, where an agent proposes configuration or infrastructure-as-code changes, maps well to the recoverable tier and merge review and is already common in the modern software-engineering process of most engineering organizations. The model, then, does not represent an additional control so much as a natural consequence of boundaries imposed by code review. In effect, an agent that opens a pull request against an infrastructure repository is making a recoverable-tier request, and the review step involved in approving a pull request can serve as the human gate that the model envisions without requiring another separate approval workflow.

Capacity and cost management, where an agent rightsizes instances and reclaims unused resources, is a mixed case: resizing an existing resource is reversible. Offsetting resource acquisition involves two parts, which are cleanly separated. Deleting an idle resource outright is not an option, at least not without a separate backup or snapshot policy already in place. Since resource-based scoping mostly treats this permission as a monolithic resource management permission, the distinction is entirely lost.

The model is also needed most and has been least tested in multi-agent architectures, in which one agent calls upon another to help achieve its goals. Delegation depth makes it harder to determine whose authority ultimately guided the action. Specifically, in the case of a derived credential (one that passes through a second or third agent), the intersection and binding properties must be preserved at each hop, which is not something that the current delegation chain metadata is built for [10], [12].

More generally, the binding economic constraint on agentic automation in infrastructure operations is suspected to be less about the capabilities of the model and more about security organizations' reluctance to give production access to AI systems at all. A containment model that is demonstrably safer than a standing credential, while providing most of the automation benefits that an organization is hoping to capture, removes a meaningful part of that reluctance. The potential value released in each case will often relate to the operational toil currently absorbed by the on-call engineers doing work that an agent with supervision could achieve. The auditability requirement running through this model is socially important for reasons beyond this security concern. As operational tasks are automated, it becomes increasingly important to reconstruct both the reasons for an action and the action itself on failure [19]. In engineering cultures, this requirement is a factor in how incident reviews are conducted. In some domains, where an infrastructure change requires a decision-maker, this consideration is also a factor in regulatory compliance.

Regulation is beginning to catch up too. The European Union's Artificial Intelligence Act provides for record-keeping requirements to ensure that automated logging supports the meaningful traceability of a system's operation through the life cycle [20]. For risk management, NIST's guidance on generative AI recommends designing provenance and accountability mechanisms commensurate with the autonomy afforded to the system [21]. Neither framework follows the reversibility-based scoping proposed in this article. Both make explicit a traceability

requirement, which any viable containment model, including this one, must satisfy as part of its design from the ground up, rather than as an afterthought bolted on after the fact [23].

There are two future architectures that are, or likely to be, widely adopted in the next few years (though neither has fully arrived yet): Authorization systems will accept task context in policy decisions alongside the identity of the principal [25]. That large change from today, where request-evaluation architectures are built the other way around, is not something that happens with a configuration flag. It will likely be about reversibility or some closely related property being an intrinsic characteristic that infrastructure APIs declare about themselves in documented metadata rather than something every organization decides for itself after the fact, as this article's methodology would require [27]. Idempotency is a closely related property, which in about a decade evolved from an informal thing engineers reasoned about case by case into a documented contract that client libraries and retry logic depend on directly. Standardization efforts already underway around workload identity in multi-system environments look like the most plausible vehicle for the first of these two shifts [12]. The second currently has no clear standards owner, and it is not clear whether any organization will pick it up. More generally, the governance of agentic systems seems to be heading in the same direction of stressing trust, risk, and security across the agent lifecycle, which this reversibility and audit model complements [22].

5. Conclusions

The article has argued that resource and principal-based identity, which is essential for ensuring production security, does not hold when the agent is an actor requesting access. Instead, the agent's intent comes from content it retrieved, not from content a human authored. This breaks the assumption that an identity is the proxy for what a request is actually trying to achieve. It is the confused deputy problem at the infrastructure scale: a legitimate and authorized identity may be used to perform an action that nobody intended.

Bounded delegation answers this question by changing the scope of the permission it grants, not on a resource-by-resource basis (which collapses once applied to a real production fleet), but on the scope of the action's reversibility. A credential based on the initiating human's authority, scoped to the action being taken, and bound to the action's lifetime, removes the standing artifact that makes a compromised agent identity valuable to begin with. These reversible operations occur without friction, and each uses a human gate as a component. This means that irreversible operations are never reachable with the credential the agent holds, regardless of how the attacker would reassemble the request.

Retaining autonomy gives us a way to grade this model. The case that captures everything is not a serious proposal. And a metric that counts only the number of incidents avoided but not autonomy forfeited misses half the tradeoff. This model does not eliminate the need to audit and analyze incidents or to scope out tasks carefully. It gives those existing practices a credential boundary that holds even when the agent's judgment about a given instruction turns out to be mistaken.

Any organization contemplating this strategy can start by independently identifying what existing automation, whether agentic or otherwise, is actually permitted to do, regardless of any particular containment model. Reversibility provides a valuable way to classify that operational surface, working on the same reasoning, even before agents have executed a single action against it. As the share of production controlled by agents grows, an identity and authorization layer underlying them has to be built for an actor whose behavior cannot be fully predicted in advance. Reversibility provides a reasonable basis for that requirement.

6. Funding Information

This research received no external funding from any funding agency in the public, commercial, or not-for-profit sectors.

7. Author Contribution

As the sole author, Harsh Kaushikbhai Patel is responsible for all aspects of this work, including conceptualization, methodology, formal analysis, investigation, resources, writing of the original draft, review and editing, and supervision of the final manuscript.

8. Conflict of Interest

The author declares no conflict of interest.

9. Data Availability Statement

This article did not generate or analyze new datasets. The methodology described is a proposed evaluation procedure for future application; the discussion draws on previously published literature and publicly available industry reporting, all of which is cited in the references section.

10. Biographical Sketch

Harsh Kaushikbhai Patel is working in production engineering, with a practical focus on cloud infrastructure security, system reliability, and performance optimization across large-scale distributed environments. His work centers on the operational controls that keep automated systems, including emerging agentic tooling, accountable and safely scoped as they take on greater responsibility inside production infrastructure.

REFERENCES

1. J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308, 1975.
2. Cloud Security Alliance AI Safety Initiative, "The Non-Human Identity Governance Vacuum: AI Agents and the Fastest-Growing Unmanaged Attack Surface," CSA Lab Space Whitepaper, May 2026. Available: https://labs.cloudsecurityalliance.org/research/csa-whitepaper-nonhuman-identity-agentic-ai-governance-v1-cs/ (Accessed on: 15 September 2026).
3. J. Hughes, "Securing the AI Era: Privileged Access Management for Non-Human Identities," Identity Defined Security Alliance, 2026. Available: https://www.idsalliance.org/broadcast/securing-the-ai-era-privileged-access-management-for-non-human-identities/ (Accessed on: 15 September 2026).
4. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," in Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec 2023), pp. 79-90, 2023.
5. J. Sotiropoulos, "OWASP Top 10 for Agentic Applications: The Benchmark for Agentic Security in the Age of Autonomous AI," OWASP GenAI Security Project, OWASP Foundation, 2025. Available: https://genai.owasp.org/2025/12/09/owasp-top-10-for-agentic-applications-the-benchmark-for-agentic-security-in-the-age-of-autonomous-ai/ (Accessed on: 15 September 2026).
6. M. Leo, F. Tan, T. Miao, and G. Anand, "From Threat to Trust: Assessing Security Risks of Agentic AI Systems," International Journal of Information Security, vol. 25, no. 1, Article 23, 2026.
7. N. Hardy, "The Confused Deputy: (or why capabilities might have been invented)," ACM SIGOPS Operating Systems Review, vol. 22, no. 4, pp. 36-38, 1988.
8. X. Muruaga, "Bounded Agents: Delegation Security for Multi-Agent AI Systems," arXiv:2608.15888, 2026. Available: https://arxiv.org/abs/2608.15888 (Accessed on: 15 September 2026).
9. Cloud Native Computing Foundation, "SPIFFE: Secure Production Identity Framework for Everyone," Specification, 2025. Available: https://spiffe.io (Accessed on: 15 September 2026).
10. M. Jones, A. Nadalin, B. Campbell, J. Bradley, and C. Mortimore, "OAuth 2.0 Token Exchange," RFC 8693, Internet Engineering Task Force, January 2020.
11. K. Ni and P. C. Liu, "WIMSE Applicability for AI Agents," Internet-Draft draft-ni-wimse-ai-agent-identity-02, Internet Engineering Task Force, February 2026. Available: https://datatracker.ietf.org/doc/draft-ni-wimse-ai-agent-identity/ (Accessed on: 15 September 2026).
12. J. Salowey, Y. Rosomakho, and H. Tschofenig, "Workload Identity in a Multi System Environment (WIMSE) Architecture," Internet-Draft draft-ietf-wimse-arch-08, Internet Engineering Task Force, 2026. Available: https://datatracker.ietf.org/doc/draft-ietf-wimse-arch/ (Accessed on: 15 September 2026).
13. V. S. Narajala and O. Narayan, "Securing Agentic AI: A Comprehensive Threat Model and Mitigation Framework for Generative AI Agents," in Proceedings of the 8th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI 2025), pp. 1-6, 2025.
14. MITRE Corporation, "MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems," 2025. Available: https://atlas.mitre.org (Accessed on: 15 September 2026).

15. Y. Xu et al., "LLM Agents Security Duality: A Comprehensive Survey of Self-Security and Empowered Cybersecurity," Artificial Intelligence Review, vol. 59, no. 8, Article 174, 2026.
16. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST Special Publication 800-207, National Institute of Standards and Technology, August 2020.
17. S. Mthethwa, M. T. Dlamini, and E. Jembere, "Zero Trust for NHIs Based on Robust Identity and Access Management for a Resilient IoT Future," Sensors, vol. 26, no. 8, Article 2392, 2026.
18. GitGuardian, "The State of Secrets Sprawl 2026," Annual Report, 2026. Available: https://www.gitguardian.com/state-of-secrets-sprawl-report-2026 (Accessed on: 15 September 2026).
19. M. Mitchell, A. Ghosh, and S. Passi, "AI Agents Push Humans Out of the Loop," arXiv:2608.23642, August 2026. Available: https://arxiv.org/abs/2608.23642 (Accessed on: 15 September 2026).
20. European Union, "Regulation (EU) 2024/1689 (Artificial Intelligence Act), Article 12: Record-Keeping," Official Journal of the European Union, 2024. Available: https://artificialintelligenceact.eu/article/12/ (Accessed on: 15 September 2026).
21. National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile," NIST AI 600-1, July 2024.
22. S. Raza, R. Sapkota, M. Karkee, and C. Emmanouilidis, "TRiSM for Agentic AI: A Review of Trust, Risk, and Security Management in LLM-Based Agentic Multi-Agent Systems," AI Open, 2026. https://doi.org/10.22399/ijcesen.5498
23. H. Gaggar, "PromptDLP: Deterministic pre-inference egress control for preventing secret and PII leakage in LLM agents," Journal of Information Systems Engineering and Management, vol. 9, no. 1, 2024. Available: https://www.jisem-journal.com/download/59_JISEM_9_1.pdf
24. H. Gaggar, "Two Planes of Truth: Execution Fidelity and BehavioralCorrectness in Tool-Using LLM Agents," International Journal of Computational and Experimental Science and Engineering, vol. 10, no. 3, 2024. doi: 10.22399/ijcesen.5498.
25. A. Bhatnagar and V. R. K. Bonagiri, "Operational Excellence through Cloud-Native Distributed Systems," International Journal of Computational and Experimental Science and Engineering, vol. 12, no. 2, 2026. doi: 10.22399/ijcesen.5207.
26. K. Rekulapalli, S. R. Kagga, V. Ayyagari, A. Akula, and R. Akula, "AI in HRM: Enhancing Workforce Competency and Organizational Effectiveness," SSRN, 2026. Available: https://ssrn.com/abstract=6762998
27. K. Rekulapalli, H. Nilagiri, and M. Vakkanthula, "A comparative evaluation of national FHIR Implementation Guides: Strategic methodologies for enhancing interoperability, regulatory alignment, and clinical comprehensiveness in global health IT," Health Systems, pp. 1–13, 2026.