NON-HUMAN IDENTITY SERIES · PAPER #2

Authorization for AI Agents: Beyond RBAC

Capability Tokens, User-Context-Aware Policy, and the Limits of Roles

Mohamad Amin Hasbini

Independent researcher · Paris, France

Published April 28, 2026

ABSTRACT

RBAC was built for humans in stable roles. Agents require authorization evaluated per-task, in real time, against the intersection of agent capability, user delegation, and task scope.

KEYWORDS — AI agent security, non-human identity, post-quantum cryptography, agent authorization, capability tokens, NIS2, DORA, EU AI Act

CITE AS

Hasbini, M. A. (2026). *Authorization for AI Agents: Beyond RBAC*: Capability Tokens, User-Context-Aware Policy, and the Limits of Roles. Non-Human Identity Series, Paper #2.
Available at https://mahasbini.org/papers/02-authorization-beyond-rbac/
PDF https://mahasbini.org/publications/papers/02-authorization-beyond-rbac.pdf

TL;DR

- **The problem.** RBAC was built for humans with stable roles. AI agents have variable scope, multiple callers per agent, and non-deterministic tool selection. Three properties RBAC has no native model for.
- **The remedy.** Authorization evaluated per-task, in real time, against the intersection of agent capability × user entitlement × task scope. The primitive that carries it is the capability token, not the role.
- **The compliance angle.** NIS2, DORA, and EU AI Act audit cycles arriving 2026-2027 expect per-request, per-task, composite-identity decisions. Shared roles cannot produce that evidence.

Between April 19 and April 21, 2026, a Vercel employee's Google Workspace was used by an attacker who never touched the employee's password. The attacker did not phish. No CVE was involved. The attacker held a valid OAuth token that the employee had handed to an AI agent named Context.ai three months earlier.

Context.ai's own cloud infrastructure was compromised. The OAuth tokens that Context.ai held on behalf of its customers were exfiltrated. One of those tokens carried the Vercel employee's full Google Workspace scope. The attacker replayed the token, walked into the employee's Workspace with the agent's privileges, and pivoted from there into Vercel internal systems. Mandiant and CrowdStrike were engaged.

The public analysis that followed kept returning to a single structural observation. The breach was not a SaaS compromise. The breach was an agent credential compromise whose blast radius happened to include SaaS.

This paper is about the authorization layer that should have prevented that blast radius from scaling the way it did.

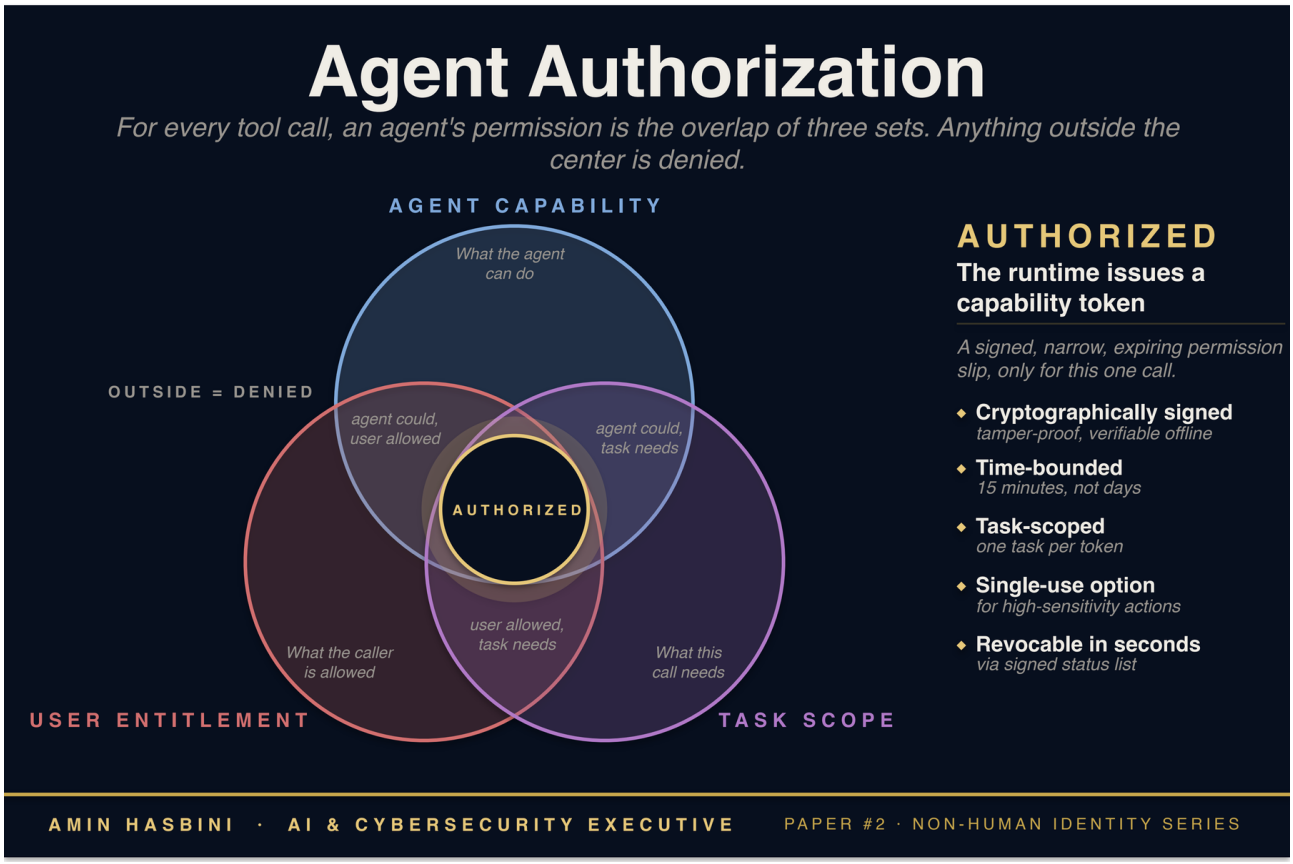
Paper #1 of this series argued that AI agents need per-agent identity. Cryptographic identity is the floor. It answers "who is the agent." It does not answer "what is this agent permitted to do, on whose behalf, and for how long."

That is the question this paper takes on. It addresses **ASI02 (Tool Misuse)** from the OWASP Top 10 for Agentic Applications 2026. **ASI03 (Identity & Privilege Abuse)** was Paper #1. **ASI07 (Inter-Agent Communication)** is Paper #3. The authorization layer sits between them: identity is authenticated at the bottom, a channel is secured at the top, and in the middle every tool call by every agent has to pass a policy decision that existing role-based models were never designed to produce.

The thesis is specific.

> *Role-Based Access Control (RBAC), built for humans in stable roles, is the wrong primitive for agent authorization. Agents require authorization evaluated per-task, in real time, against the intersection of what the agent is permitted, what the delegating user is permitted, and what the specific task requires.*

The primitive that emerges is the capability token, not the role. The policy engine evaluates per-request, not per-session. The stack this produces is different enough from the one most enterprises deploy today that pretending otherwise is the expensive mistake.



Authorized = where all three overlap. Anything outside is denied.

Why RBAC existed, and why it worked for humans

When a human joins an enterprise, a stack of paper and process anchors what they are allowed to do. The employment contract names a role. The job description enumerates the duties of that role. The employee countersigns an NDA, a code of conduct, a data-classification acknowledgement, and often an IP assignment. Access-request forms route through the line manager, then through compliance, then through the identity team. The role is attached in the directory. Training is completed and logged. A workstation is provisioned. Months later, in a promotion or a transfer, the role is mutated through another set of forms. When the employee leaves, offboarding closes accounts within days.

RBAC was the technical primitive that carried this entire apparatus into the directory. A role in Active Directory or an IAM role in AWS was the digital shadow of the role the HR department had already validated on paper. A treasury analyst had a role called “Treasury Analyst” with permissions to read balances, post journal entries within a ceiling, and run end-of-day reports. The permissions stayed consistent because the role stayed consistent. The role stayed consistent because HR was maintaining the paper underneath it on a weeks-to-quarters clock.

RBAC was formalized in a 1992 NIST paper and standardized in the NIST RBAC Reference Model in the early 2000s. It scaled because role membership was a slow-moving property. Human resources changed it when someone was promoted, transferred, or left. Identity governance tools reconciled the directory against the payroll system quarterly. Auditors walked the role matrix and signed it. The whole apparatus assumed that the answer to

"what is this principal allowed to do right now" could be looked up in a table maintained by a human process operating on a weeks-to-quarters clock. That assumption held because the paper under the table held.

NIST SP 800-162 began the formal migration toward Attribute-Based Access Control (ABAC) for precisely the cases where that assumption broke. ABAC evaluates attributes of the subject, the resource, the action, and the environment at decision time, instead of pre-binding a principal to a fixed role. Agents make that evolution urgent, not optional. An agent has no HR file, no countersigned NDA, no quarterly reconciliation. The paper underneath the directory is missing. Whatever the directory asserts about an agent has to be asserted at the moment of the action, not months earlier by a clerical process that no longer applies.

Four structural reasons RBAC fails for agents

Consider a concrete scenario. An enterprise deploys one procurement agent. Three employees call it the same morning. Alice, in Treasury, asks for a supplier risk summary. Bob, in Operations, asks it to raise a purchase order for industrial lubricant. Carol, in Legal, asks it to pull the active contract terms with Supplier X. Three callers, three tasks, three different data domains. One agent, one RBAC role.

The role attached to the agent has to cover the union of what all three callers could ever legitimately ask for. Anything less and the agent fails to serve one of them. Anything more and the agent has latent authority it should never exercise in a given call. RBAC has no native way to scope the action down to the caller plus the task. The four failure reasons that follow are what this single structural gap produces.

Agents are not single-principal. A human in a role acts on behalf of themselves. An agent acts on behalf of whoever invoked it. A single LangChain agent exposed to five business lines is called by employees with different scopes. The agent's effective authority on any given invocation is a property of the caller, not of the agent. RBAC has no native way to express "this call should intersect the agent's own permissions with the caller's permissions." The agent ends up with a role that is the union of every caller's capabilities, because anything less breaks the default invocation.

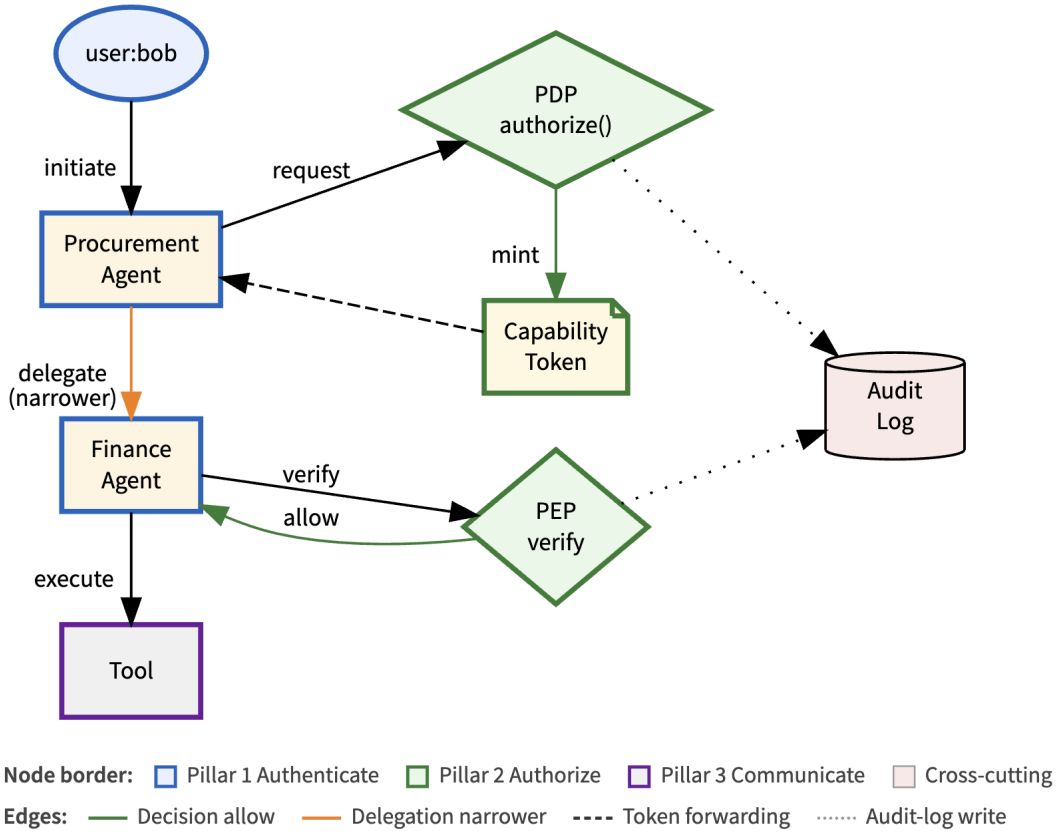
Agent scope is task-variable, not role-stable. A treasury analyst always does treasury. A procurement agent asked to "find me the three best-rated suppliers for industrial lubricant in Northern Italy with a delivery commitment under two weeks" and the same agent asked to "renegotiate the MSA with Supplier X for the Q3 volumes" should not operate with the same effective scope. The first task needs read access to the supplier database. The second needs write access to contract records, probably with a spend-ceiling guardrail, and an approval routing. RBAC binds the agent to one role. The task boundary that should scope the action down does not exist in the model.

Role explosion is combinatorial. The obvious fix is to define a role per agent per task per caller. Ten agents. Twenty tasks per agent. A thousand callers. Two hundred thousand roles. No directory service survives this at scale. No auditor reviews this matrix. The explosion is not an implementation detail. It is the boundary at which the RBAC primitive loses analytic usefulness.

The actor is non-deterministic within a session. A human in a role executes an approximately knowable workflow. An agent driven by an LLM chooses its tool invocations at runtime, against a surface of a dozen or a hundred functions, with reasoning not reproducible across runs. The session-level policy decision that worked for humans (check the role once, trust the session) breaks for agents: the agent may invoke a tool in minute five that

was not reachable from the reasoning trace of minute three. Policies must be evaluated per tool call, not per session. The multi-university *Agents of Chaos* red-teaming study, published in 2026 and co-authored across ten institutions, demonstrated empirically that session-level policy enforcement allows adversarial prompt sequences to walk an agent into tool combinations that the original role binding would have refused. The paper is not prescriptive on the remedy. The remedy is obvious once the failure mode is named.

The three dimensions authorization must evaluate



The runtime flow. A human launching principal initiates a task through an agent. The agent requests capabilities from the Policy Decision Point, which evaluates the three-dimensional intersection (agent envelope, user entitlement, task scope) and mints a signed capability token on allow. The token is carried forward; narrower child tokens are derived for sub-agents. Each downstream tool call is gated by a Policy Enforcement Point that verifies signature, expiry, and scope without round-tripping to the PDP. Every decision writes to a cross-cutting audit log. Click the figure to run the flow interactively on platform.mahasbini.org.

For every tool call by every agent, the authorization engine must intersect three things.

The agent capability envelope. What the agent is fundamentally permitted. Issued at provisioning. Encoded in the agent's registered identity (Paper #1) plus a capability declaration that the registry signs. Answers "could this type of agent, in principle, do this."

The delegating user's entitlement. What the human caller is permitted at this moment, evaluated fresh against the directory, not cached. Answers "does the person on whose behalf the agent is acting have the right to authorize this."

The task-specific scope. What this specific invocation actually needs, derived from the request context, time-bounded. Answers "does this particular operation fit within the task the human initiated."

Authorization is the **intersection** of the three. Any one scopes the others down.

Dimension	Classic RBAC	Capability-token
Agent capability	role attached (union of scopes)	capability envelope, signed at provision
User entitlement	not consulted at tool-call	fresh entitlement check, every call
Task scope	not modeled	`task_id` in token
Evaluation	once per session	once per tool call

The failure mode of today's agent deployments is that the system evaluates at most one of the three. The agent's IAM role carries the agent's capability. The delegating user's entitlement exists in the directory but is not consulted at tool-call time. The task-specific scope is captured in a conversational turn that never enters the policy decision. Three dimensions reduce to one. Authorization becomes "what does the agent's role allow," which is the 2023-2025 pattern that the Vercel breach and a dozen less-public incidents have now demonstrated to fail under OAuth token theft.

Capability tokens as the primitive

The primitive that carries all three dimensions is the capability token, not the role.

A capability token is a cryptographically signed assertion that the bearer is authorized to perform a specific operation, on a specific resource, within a specific time window. It encodes a **capability** ("can read supplier records for orders under 50k euros, until 15h04 UTC") rather than an **identity** ("is a member of the Procurement Agent role"). The relying party can verify the token offline by checking the signature. The relying party does not need to consult a directory for every decision.

Four properties make capability tokens the right primitive for agents.

One task per token. The token says what the task is, not who the agent is. Back to the procurement agent scenario above: when Bob asks for a purchase order, his agent receives one token that says "read suppliers, initiate a PO on Bob's behalf, within Bob's spend ceiling, for the next 15 minutes." When Carol asks for contract terms two minutes later, her agent receives a different token that says "read contract records for Supplier X, on Carol's behalf, for the next 15 minutes." Same agent, same day, different tokens. Neither token can be used to do the other's job.

Short expiry. A capability token lives for minutes, not days. If the token is stolen, the window of misuse is small. Compare this to the Vercel incident: the OAuth token held by Context.ai had been valid for months. A capability

token would have expired long before the attacker pulled it out of storage. Short expiry is not a burden for the agent; the token is re-issued as naturally as a fresh HTTP request.

Self-explanatory in the log. The relying party can read the token's claims directly. The audit log ends up with a line that already contains the answer to "which task, which user, which data, which moment." No stitching a session log to a role matrix to a conversation transcript after the fact. An auditor asked "what did agent X do on Tuesday at 14h03" reads one line instead of joining three databases.

Forwardable, only narrower. Agents often call other agents. When they do, the first agent hands the second agent a token derived from its own, but scoped down. If agent A holds a token good for three actions, agent A can give agent B a token good for one of them. Agent B cannot widen the token. Agent B cannot forward it without narrowing further. This pattern is what **OAuth 2.0 Token Exchange (RFC 8693)**, **Rich Authorization Requests (RFC 9396)**, and **GNAP (RFC 9635)** collectively standardize. The rule is the same in all three: forward the authority, never grow it.

None of these standards are new. What is new in 2026 is that the AI-agent community has inherited OAuth 2.0 casually, without adopting the extensions that make it safe for delegated task-scoped authorization.

Two IETF drafts to engage now

Agent authorization is not an isolated invention. Two active working drafts at the Internet Engineering Task Force already address pieces of the problem, and both shape what enterprise deployments will standardize on by 2027. Naming them matters for three reasons: readers (CISOs and architects) can track a concrete document rather than a concept, the enterprise-deployment and regulatory-mapping layers this paper adds are complementary to the spec work, and the direction of travel is already visible in the drafts. Paper #2 takes positions on both.

draft-klrc-aiagent-auth-00 (March 2026). Co-authored by Brian Campbell (Ping Identity), AWS engineers, Defakto Security, and Zscaler. The draft composes OAuth 2.0 Token Exchange (RFC 8693) and Rich Authorization Requests (RFC 9396) for AI agent authentication and authorization. It names the problem of delegated agent identity and the composability argument for assembling authorization from existing OAuth primitives. The draft is the right direction and incomplete. It specifies the token mechanics but underspecifies three things that matter for enterprise deployment. Task-scope decomposition semantics: how a generic "read supplier records" claim maps into the task-bounded scope an agent actually needs for one invocation. Delegation-chain depth limits: whether a forwarded token can be forwarded further, and if so with what additional scope reduction. EU regulatory mapping: how the token produces the audit artefacts that DORA Chapter II, NIS2 Article 21, and EU AI Act Annex IV will demand from the 2027 supervisory cycle onward.

This paper contributes the enterprise deployment patterns and the EU regulatory mapping layer. The goal is complementary, not competitive. The IETF draft does the spec-level mechanics. Paper #2 does the deployment and regulatory semantics that the spec by design does not.

draft-ietf-oauth-transaction-tokens-08 (March 2026). Contributor: George Fletcher (Practical Identity LLC). Transaction Tokens propagate user identity, workload identity, and authorization context through multi-service call chains. The draft is the call-chain-propagation primitive that the agent-delegation literature has needed without always naming it. Transaction Tokens are close to the "composite identity carrying launcher plus actor" pattern that engineering practitioners have been assembling by hand. The draft makes the pattern standard.

Paper #2 cites both drafts as the state of the art and extends them with what the enterprise-deployment and regulatory layers actually need. Reviewers with standing in either working group are invited to engage before the May publication window.

Monotonic scope decay in the delegation chain

The rule for delegation in multi-agent chains has a specific form. Every delegation hop must shrink the scope carried forward. It must never widen it.

An agent A holds a capability token with scope S. Agent A invokes agent B to perform a sub-operation. Agent A derives a new token with scope S' for agent B, where S' is strictly narrower than S, shaped to exactly what B's sub-operation needs. Agent B cannot expand the token's scope by combining it with another token. Agent B cannot hand the token to agent C without further narrowing.

This is the **monotonic scope decay** property. It is the delegation-chain analogue of the per-agent credential revocation principle from Paper #1. In production, it is enforced by a Policy Enforcement Point that sits in front of each downstream tool and calls a Policy Decision Point (Open Policy Agent, Cedar, or equivalent) with a composite identity payload: the launching principal's claim, the originating agent's claim, and the immediate caller's claim for this hop. The policy engine enforces the intersection of all three against the requested operation.

The composite identity is carried through the chain so that the audit log answers a specific question at any moment: which human launched this chain, which agent is currently acting, and under which task scope. The answer is not reconstructed after the fact from correlated logs. It is attached to every call by construction.

What this rules out is the ambient-scope pattern. An agent cannot act on an authority it inherited once and then carried around. Every downstream call needs a fresh token derived from the current task's authority. Caching is bounded to the token's own expiry. Session-level authority caching is the wrong answer.

Revocation and replay defense

Short expiry is a first line of defense against a stolen capability token. It is not sufficient. Two additional primitives bound the damage.

Revocation via a status list. Every capability token carries a unique identifier. The issuer publishes a signed status list naming revoked token identifiers. The Policy Enforcement Point consults the status list at verify time and refuses any token whose identifier appears in it. Revocation takes effect within the status-list polling interval, which in practice is seconds to low tens of seconds. This is the capability-token analogue of the certificate revocation list from the PKI world, tuned for short-lived tokens instead of long-lived certificates.

Single-use tokens for high-sensitivity operations. A capability token minted with a single-use flag passes verification exactly once. A second verification attempt for the same token identifier is refused with a replay-defense reason code, independent of signature, expiry, and scope. This eliminates the replay window entirely for operations where the capability should not be reusable within its TTL. The cost is operational: the PEP has to

maintain a consumption cache (or query the status list at each call). The benefit matters for financial transactions, credential issuance, and irreversible actions where one valid use is correct and a second is an attack.

The Vercel / Context.ai incident that opens this paper illustrates the importance of both. A compromised Context.ai infrastructure held long-lived OAuth tokens for its customers. With short-lived capability tokens plus an active status list, the exposure window compresses from months to minutes. With single-use tokens on the subset of operations that warrant it, the replay opportunity reduces to zero.

Neither primitive eliminates the root compromise. They compress the attack surface into a narrow time window where detection and response can catch up. That compression is the operational value.

Policy-as-code as the evaluation layer

Three-dimensional intersection cannot be evaluated in a spreadsheet. It has to run in code, at the moment of every tool call. Four stacks are worth knowing by name in 2026.

Cedar. A policy language that Amazon open-sourced in 2023. Policies read close to plain language ("permit User to perform Action on Resource if…"). The engine runs as a library inside the application. Easy to reason about. Good first choice when building a new agent platform from the ground up.

Open Policy Agent (OPA), with the Rego policy language. The most widely deployed policy engine in the Kubernetes and service-mesh world. Runs as a sidecar or a library. More expressive than Cedar, at the cost of being harder to read. Good choice when the organization already runs OPA for other workloads and wants to extend into agent authorization without adding a second engine.

OpenID AuthZEN. Not a policy language. A wire protocol, still being standardized in 2026, that lets an application ask any compliant policy engine "is this action allowed." Think of it as the HTTP of authorization decisions. Adopting AuthZEN means the organization can swap Cedar for OPA later without rewriting the application side.

Google Zanzibar. The relationship-based model behind Google's internal authorization. Open-source rebuilds exist (SpiceDB, Keto). Worth mentioning because it is cited often. Worth qualifying because relationships are a weaker fit than capabilities for the task-scoped, per-invocation decisions agents require.

The operational implication is simple. Policy evaluation is a per-request event. It sits at a Policy Enforcement Point in front of every tool the agent can call. Which engine to pick follows the organization's existing footprint. The directory does not need rebuilding. The Paper #1 identity layer does not need replacing. An authorization decision layer is added on top and consulted at every tool call.

The mandate pattern, made cryptographically enforceable

There is a pre-digital analogue that makes the agent authorization problem legible to non-technical stakeholders. A human gives a lawyer or a real-estate agent a mandate. The mandate authorizes the agent to perform a specific

task, on specific terms, for a specific duration. The mandate is explicit. The mandate does not travel. The agent cannot use a mandate written for a house sale in Paris to conduct an unrelated transaction in Geneva.

Agent authorization in 2026 is the digital form of the mandate, made cryptographically enforceable. The capability token is the mandate. The Policy Enforcement Point is the notary who checks the mandate before letting the action proceed. The monotonic-scope-decay rule is the principle that a sub-delegation cannot widen what the principal authorized.

This framing matters for two reasons. The first is that boards and legal counsel understand mandates. The vocabulary connects agent authorization to an accountability model that predates the digital era by centuries. Liability accumulates invisibly only when the mandate disappears from the audit trail. The second is that the mandate framing disciplines technical choices. A capability token is the digital mandate. A long-lived API key is not a mandate. A shared role is not a mandate. Anything that cannot be pointed to as "the mandate for this specific action by this specific agent on this specific principal's behalf" is the weak link.

What NIS2, DORA, and the EU AI Act will demand

Three regulatory regimes are converging on the same operational expectation for agent authorization, through different articles.

NIS2 Article 21 requires access controls commensurate with risk. For AI agents operating on critical services, "commensurate with risk" in the ACPR and ANSSI inspection language of late 2025 and early 2026 has begun to mean per-request evaluation against the delegating user's current authority. A shared role that cannot produce that evidence is increasingly treated as a material control gap during ICT on-site missions. ANSSI's December 2025 guidance on cybersecurity qualification for AI products enumerates "traceability of authorization decisions per task" among the evaluation criteria.

DORA Chapter II requires auditable ICT access trails. The trail must reconstruct who did what on whose behalf at which moment. For agent workflows, this translates to per-agent, per-user, per-task logging at the authorization decision layer. Capability tokens with introspectable claims produce this artifact natively. Role-based session logs do not.

EU AI Act Annex IV requires high-risk AI systems to be documented, logged, overseen, and robust. The logging granularity is specified at "sufficient detail to investigate and reconstruct system behaviour." For agent systems, the only log format that satisfies this is the per-task authorization decision log. The AI Act supervisory cycle begins to apply for high-risk systems in 2026-2027. The ACPR and Banque de France have signaled that AI-agent access control will be among the controls evaluated during their supervisory dialogues with regulated entities from 2027 onward.

The convergent pattern is that per-request, per-task, composite-identity authorization decisions are the granularity at which the three regimes, reading each other's work, collectively arrive. The regulatory expectation is not forward of the technical best practice. It is arriving at the same point through a different route.

An observation from Paper #1 launch-day discussions is worth elevating. Per-agent authorization is increasingly framed by experienced CIOs and Heads of AI not as an audit finding but as a prerequisite to AI budget approval.

The executive-level reframing is strategically useful. It shifts the investment decision from cost-center (compliance) to enabling-infrastructure (without this, the AI program cannot ship to regulated production).

Five things your CISO can do next Monday

1. **Inventory your agents by role attachment.** For every agent in production, list the tools it has access to and the IAM role or OAuth client that carries the permission. Any agent with more than ten tools attached, or with a role that grants access to data belonging to more than one business function, is flagged.
2. **Instrument the task context.** Every agent invocation should produce a task identifier that propagates through the tool calls. Without a task identifier, the three-dimensional intersection cannot be evaluated, and the audit log collapses to session granularity.
3. **Deploy a Policy Decision Point alongside the existing IAM.** Cedar or Open Policy Agent, whichever matches the organization's existing footprint. Start with one high-value agent. Evaluate authorization per tool call. Measure latency overhead; it is typically in the single-digit milliseconds when the policy fits in memory, which it should.
4. **Wire OAuth Token Exchange (RFC 8693) into the agent platform** so that agent-to-agent and agent-to-tool calls carry explicit delegation. Align with draft-klrc-aiagent-auth-00 as the draft stabilizes. Do not wait for the draft to ship as an RFC. The deployment pattern is clear enough to adopt incrementally.
5. **Log at task granularity, not session granularity.** Each authorization decision produces a log line with the composite identity, the task identifier, the capability scope evaluated, and the decision. Route the log to the SIEM. Surface it in the audit interface. This one change is what turns the system into something a regulator can read. The Agent Identity Platform companion tool demonstrates this end-to-end.

These five are operational, not strategic. They do not require a platform replacement. They do require the organization to treat the authorization decision as a first-class event in the agent runtime. Most 2026 deployments do not, which is why the adoption curve will be measured in single-quarter sprints at sophisticated shops and in two-year programs at the median enterprise.

Closing

RBAC was a good answer to “who is a treasury analyst, and what can treasury analysts do.” It is the wrong answer to “what can this agent, acting for this user, on this task, in this moment, do to the supplier database.” The question shape is different. The primitive must match.

Capability tokens express the capability, not the identity. Policy engines evaluate per-request, not per-session. Delegation chains propagate composite identity and shrink scope monotonically. Three regulations converge on per-task authorization as the audit granularity they expect. None of these pieces is new as a standard. What is new is treating them as the operating baseline for AI agents in regulated production, rather than a research topic.

The authorization layer is also where the second failure mode hides. Paper #1 described the identity failure. This paper has described the authorization failure. The third failure is at the channel: what agents send to each other, on

what cryptographic substrate, with what guarantees of confidentiality and integrity against a quantum adversary. Paper #3, *Agent-to-Agent Communication: MCP, A2A, and the PQC Question*, takes that on.

There is a further layer beyond the channel. Paper #3 also introduces the zero-knowledge proof primitive that lets an agent prove it has authorization for an operation without revealing the credential that carries the authorization. The primitive matters because it bounds the blast radius that this paper's capability-token model cannot eliminate by itself. The harder question is how an agent proves to a peer that it holds a valid capability without disclosing the capability itself. The answer is constructible today with the NIST-finalized ML-DSA and ML-KEM primitives. The work of the next two years is making the construction operationally viable at enterprise scale.

Companion tools

- **Agent Identity Platform**: reference implementation demonstrating per-agent identity (Paper #1), capability-token issuance, 3-dimensional authorize() decisions, monotonic-scope-decay at delegation, PEP verification per call, revocation via status registry, and single-use replay defense. Unified audit log with composite identity, task_id, and capability scope columns for regulator-ready review. Dedicated pages on the platform:
- **Regulatory Mapping**: each NIS2, DORA, and EU AI Act article cited in this paper, mapped to the exact demo feature that satisfies it and the evidence artifact it produces.
- **Threat Model**: ten-threat attack-surface analysis naming what the capability-token layer defends against (T1, T2, T9, T10) and the four production-hardening residuals the demo knowingly leaves open (T4, T6, T7, T8).
- **AI Agent Security Maturity Assessment**: three-pillar diagnostic grounded in Paper #1. Paper #2 authorization-layer scoring is forthcoming; for the interim, use the Regulatory Mapping and Threat Model pages on the Agent Identity Platform to assess the Paper #2 controls.
- **AgentTrustLab**: simulator for agent-to-agent authentication and authorization under PQC and ZKP constraints. Paper #3 territory, available now for early experimentation.

Feedback welcome. Field observations shape what comes next.

Appendix: Concrete construction pointers for peer reviewers

The body of this paper is written for CISO and board audiences. For cryptographic and protocol peer reviewers, the following pointers name the specific constructions behind the capability-token and delegation sections.

Capability token. A candidate instantiation: an SD-JWT-VC carrying custom claims for `capability`, `task_id`, `composite_actor` (launching principal plus agent actor), and a short `exp`. Signed with ML-DSA-65 (NIST FIPS 204) by the issuing registry (see Paper #1 appendix for the registry construction). Bound to the agent's SPIFFE SVID at issuance so that the transport channel and the authorization claim are anchored to the same cryptographic root.

Delegation. RFC 8693 OAuth Token Exchange with a typed `actor_token_type` that identifies the caller as an agent rather than a human. The `scope` claim is derived from the invoking capability token with a strict narrowing rule enforced at the Token Exchange endpoint. The Token Exchange service itself is stateless: it verifies the input token, narrows the scope, and signs a new short-lived token without requiring a session lookup.

Policy evaluation. Policy Enforcement Point integrated as a sidecar or library at each tool. Policy Decision Point evaluates Cedar or Rego policies that take the composite identity, the task context, and the requested operation as inputs. OpenID AuthZEN wire protocol between PEP and PDP for portability.

Forward-secrecy and revocation. Capability tokens carry no refresh. When a capability expires or a delegation session is revoked, any outstanding forwarded tokens carrying that capability also lose authority at next verification. Revocation is effectively a capability status list checked at policy evaluation time, not a synchronous callback.

PQC posture. ML-DSA-65 signatures on capability tokens and actor claims. Hybrid X25519+ML-KEM-768 key encapsulation on the channels that carry tokens (MCP and A2A, Paper #3 territory). No capability token should be protected by a pure classical signature scheme for deployments whose confidentiality horizon extends past 2030.

A full threat model and security argument is deferred to a companion technical note. Peer review is welcome before publication. Specific review interest: the composite-identity semantics in multi-hop delegation, the monotonic-scope-decay rule at the Token Exchange boundary, and the mapping from capability-token claims to NIS2 / DORA / EU AI Act evidence artifacts.

About the author

Amin Hasbini is an AI and cybersecurity executive based in Paris. Former director of Kaspersky's Global Research & Analysis Team (GReAT) for the Middle East, Turkey, and Africa. Twelve years, seventy countries of threat coverage. Invited contributor to the French Senate's OPECST report on AI risks (2024). Former subject-matter expert on ICANN's second DNS Security, Stability, and Resiliency Review Team (SSR2, 2017-2019). Current focus: post-quantum cryptography maturity and AI agent security inside regulated enterprises. mahasbini.org.