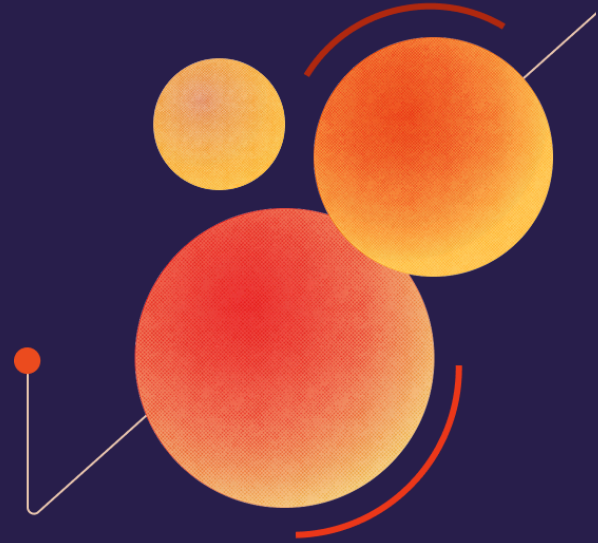




The Gateway Is Not the Identity: *Attested Trust for Agentic AI*

By Ken Huang

Disclosure: This article was written in collaboration with Aembit.
The framing, architectural opinions, and checklist are my own.



Why the Request *Should Be Trusted*

“I need a gateway” is the wrong first sentence in most agentic AI security conversations. An AI gateway is easy to draw, and it offers one place that can route, inspect, rate-limit, log, and block traffic. As applications add models, tools, MCPservers, and agent-to-agent connections, that choke point starts to look like the whole answer.

It is not the whole answer. The first security question is: **Why should anyone trust the request?**

Before a Policy Enforcement Point can allow or deny live traffic, an identity and access layer has to establish who is asking; whether the software presenting itself as an agent is the approved workload running in the expected environment; which human, process, or upstream agent delegated authority to it; what resource and action that authority covers; how long the authority remains valid; and whether current risk and business context still permit the action. Those are questions about identity, attestation, delegated authorization, credentials, and policy decisions. An AI gateway can enforce these policy decisions, but it cannot create the underlying trust evidence that justifies them.

This distinction is far sharper for autonomous agents than for traditional software. In a standard web application, platform engineers typically grant the service a high-privilege connection to the datastore, leaving the application logic to enforce fine-grained authorization for deterministic code.

An agent, however, is a dynamic planner. The underlying language model determines the execution path, selecting tools and parameters at runtime. Prompt guardrails offer behavioral guidance, not deterministic access control. Because an agent may require read access in one step, a tightly scoped write in the next, and zero permissions in an unanticipated third, issuing a static credential broad enough to cover every potential action turns dynamic execution into dangerous standing privilege.

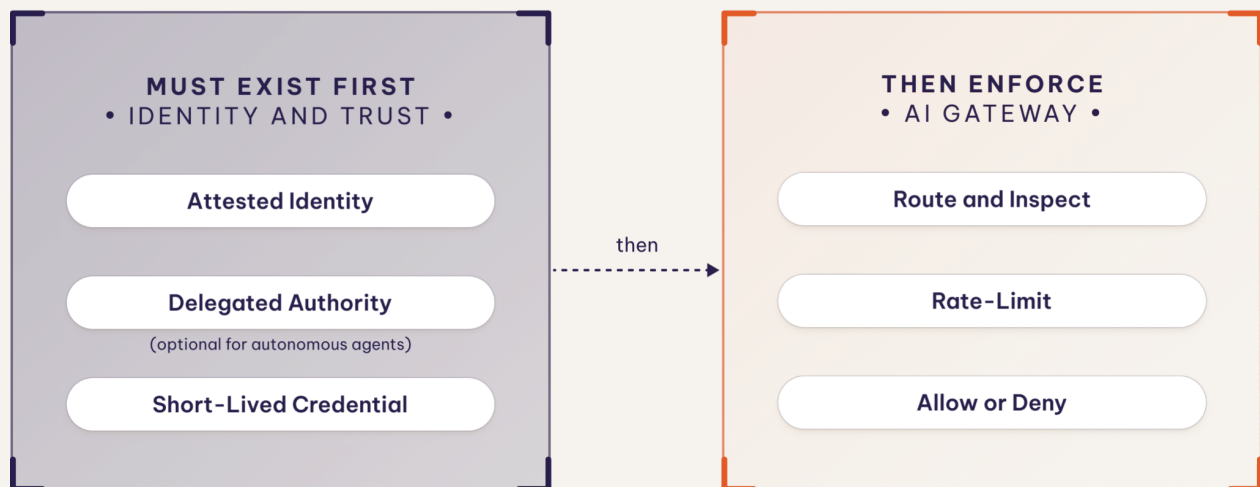


A safer model applies deterministic policy to a non-deterministic actor. The system attests the agent as a workload, binds that attested identity to the human, process, or organizational role on whose behalf the agent acts, and re-evaluates policy at the action boundary. It then issues a short-lived, audience-bound credential and records the chain so an auditor can reconstruct who acted, for whom, and with what authority.

Figure 1 illustrates the architectural relationship between trust establishment and policy enforcement: a gateway can route or block traffic, but defending an allow-or-deny decision requires pre-existing identity attestation, understanding whether there is delegated authority, and short-lived credentials. Delegated authority is the workforce case: a person or an upstream agent hands the agent a task. An unattended agent that runs on its own service account has no delegator, so that input is optional for it, while attestation and short-lived credentials still apply.

FIGURE 1

Enforcement is Downstream of Trust



A gateway can route and block. It does not create the trust evidence.



What an AI Gateway *Does Well*

The strongest case for identity-first security does not diminish the AI gateway. It gives the gateway a job it already performs well.

Agent execution generates erratic traffic patterns through streaming, retries, fan-out tasks, dynamic model selection, and tool discovery. AI gateways address this by standardizing provider APIs, optimizing cost and regional routing, handling model failover, converting AI API JSON responses into Model Context Protocol (MCP) tools, and mediating agent-to-agent transports. Modern solutions represent distinct approaches to this role: Kong layers AI gateway features onto an established enterprise API management core; the Linux Foundation's Agentgateway provides a lightweight, open-source Rust proxy built specifically for LLM, MCP, and A2A protocols; and Google Cloud Apigee integrates MCP endpoints, tool registries, and governance into mature enterprise infrastructure.

These functions matter to security even when they are not identity functions. Controlled routing keeps regulated data in approved regions. Failover can stop an outage from sending prompts to an unapproved model. A catalog can cut down on shadow integrations.

An agent can turn one user request into a burst of model and tool calls. Traditional request-per-second limits still help, and AI workloads add token budgets, model spend, concurrency, tool-call fan-out, and loop duration. An AI gateway can rate-limit by client, model, tool, or credential and cut off a runaway loop. Identity makes those limits attributable, so a limit keyed only to an IP address or a shared key does not punish unrelated users. A limit keyed to a verified agent-plus-user pair can stop the offending workflow and leave everyone else running.

Gateways are well positioned to inspect prompts, tool parameters, retrieved context, and model responses. They can invoke moderation APIs, enforce data loss prevention (DLP) rules, and restrict egress to unauthorized destinations. However, content inspection evaluates the payload, whereas identity and access management evaluates the actor and delegated authority—robust production architectures require both.



Inline components capture standard operational telemetry—traffic volume, latency, model selection, token consumption, tool invocations, execution status, and policy outcomes.

Observability stops at metrics and logs; true auditability requires provenance:

- Attested agent instance A (instantiated from blueprint B), operating on behalf of user C for task D, requested an action against resource E.
- Policy version F authorized the request.
- Credential G was minted specifically for audience H with an ephemeral time-to-live.

While the gateway captures the runtime event, the identity layer provides the principal context, policy evaluation, and joinable audit trail required for SIEM ingestion.

Architecturally, the gateway functions as a Policy Enforcement Point (PEP), applying allow-or-deny decisions directly to network traffic. The Policy Decision Point (PDP) remains distinct: a centralized control plane (such as Aembit Cloud) evaluates policy, while enforcement components—such as an MCP identity gateway, Kong, Apigee, Agentgateway, or a service mesh—enforce the decision on the wire. Gateways with embedded policy engines collapse these distinct roles into a single component. However, decoupling them remains critical because their failure modes are fundamentally different: a PEP failure disrupts traffic availability, whereas compromised identity evidence causes the PEP to execute unauthorized actions accurately.



However, decoupling them remains critical because their failure modes are fundamentally different: a PEP failure disrupts traffic availability, whereas compromised identity evidence causes the PEP to execute unauthorized actions accurately.

The Model Context Protocol (MCP) formalizes these enforcement boundaries. The MCP authorization specification defines OAuth-based flows over HTTPS, mandates protected-resource metadata, and restricts token scopes to designated resource servers. It prohibits clients from relaying inbound MCP tokens to downstream APIs. While an AI gateway can enforce these transport-level constraints, standard JWT signature validation only proves that a trusted issuer signed a payload. Defensible security still depends on the upstream identity layer registering verified agents, evaluating runtime attestation, maintaining delegated authorization chains, and managing credential lifecycles.





What Only Identity *Can Answer*

An agentic access decision is more than “client X has scope Y.” It needs a verifiable software actor; proof that this actor is the approved workload; the human or process behind it; a real grant of delegated authority; a specific resource and action; current context; a short-lived, audience-bound credential; and a durable record.

If you identify only the user, the agent disappears. Downstream logs attribute every action to the person even when the software chose the tool. If you identify only the agent, you lose the person or process that delegated the work. Shared service accounts collapse several automations into one principal.

Aembit calls the combination Blended Identity. One policy decision evaluates the agent’s attested workload identity and the human identity validated through an enterprise workforce identity provider such as Okta, Microsoft Entra ID, or Google. Policy can distinguish between Ken using an approved client, Ken using an unapproved client, and another employee using the same approved client. The same model works when no user is present. The platform attests a custom autonomous agent as a workload identity from the trust sources the environment already has: cloud metadata services, Kubernetes service-account tokens, OIDC, Kerberos, VMware, or a custom attestor. You get one identity with many sources of trust, for both blended and autonomous cases.

Static secrets do not prove identity. A string such as `agent_id=expense-bot` is an assertion any caller can copy. Strong workload identity begins with evidence the platform can verify. SPIFFE and SPIRE supply a useful primitive, a signed, short-lived workload identity, but they do not decide business authorization. Aembit maps attested evidence to ownership, purpose, delegated authority, and the credential the target already accepts.



A Cloud Security Alliance survey commissioned by Aembit reports that 68 percent of surveyed organizations cannot clearly distinguish activity performed by AI agents from activity performed by humans. That finding points to a practical gap: organizations are deploying actors faster than they can name, govern, and attribute them. It is a strong case for an identity layer, not a reason to wait [3].

**It is a strong case for an identity layer,
not a reason to wait [3].**

Delegation belongs in the policy model as a first-class object. If each hop forwards the original bearer token, every component receives more authority than it needs. OAuth 2.0 Token Exchange (RFC 8693) can express that one party is acting for another, but few systems implement the proposed standard widely today. A production system still needs a policy model for what can be delegated, how deeply, to which agent, and for how long. The secure default is attenuation: every grant narrows or preserves authority, and none of them silently expands it.

Lifecycle is still identity work. Teams create, copy, version, suspend, and retire agents. Microsoft Entra Agent ID creates specialized directory objects and blueprints and extends Conditional Access to agents. Okta Cross App Access focuses on signed identity assertions between requesting agents and resource applications. CyberArk approaches the problem from privileged access and zero standing privilege. These platforms are strong where the human, the enterprise directory, and SaaS applications sit at the center.

Aembit sits alongside them, not as a directory replacement and not as a generic AI gateway. It is the workload-attestation and credential-broker layer that works across clouds, SaaS, on-premises systems, and AI platforms. Like Microsoft Conditional Access, Aembit evaluates policy when credentials are issued or refreshed and can apply conditional access on risk posture, geo, and time. The difference is the left-side job Aembit was built to do: attest the running software, blend it with the human when a human is present, and give every enforcement point the same identity context.





Human approval for a payment, a source-code merge, or a destructive change is not new. Mature organizations already require a second person before money moves and a pull request before code lands. That control lives in the business process. What changes with agents is the path. A conventional application does at runtime what the developer coded. An agent does what the model selected, within whatever deterministic constraints exist at the time. Identity does not replace the approval workflow. It makes the agent a named principal in that workflow and binds the approval to the exact action, not to a vague conversational confirmation.

Even in an intent-based world, policies should stay deterministic. A prompt may say “draft, do not execute.” A manipulated agent can still produce a plausible explanation for an unsafe call. Policy should prefer verifiable facts: approved task type, tool, operation, resource, classification, amount, user role, runtime identity, and required approval. Intent can help choose which policy to evaluate. It should not create authority the principal does not already have.

Those answers sit on five planes. Governance and directory register the agent, its sponsor, its lifecycle, and the resources it may ever touch. The trust and policy plane attests the running workload, blends identities, and decides. The credential plane mints a short-lived, audience-bound token. Enforcement applies that decision on the wire. Resource authorization, row-level rules, transaction limits, and domain invariants stay with the target.



Figure 2 shows the five planes of a mature agent-access architecture. Aembit provides the identity and policy layer across the first three. Enforcement can sit in Aembit, in another AI gateway, or in both.

FIGURE 2

Five Planes of Agent Access



One Identity Layer. Multiple Enforcement Points

*Aembit is the identity and policy layer across planes 1-3.
Plane 4 can be Aembit, Kong, Apigee, agentgateway, or a mesh.*



The *Blended Identity* Decision

Identity for an AI agent is not a static identifier—it is an ephemeral authorization decision evaluated per action. Static labels like expense-bot or upstream user tokens forwarded into downstream tool calls fail to establish true access context. Defensible security requires continuous, request-time evaluation: is this specific principal or sponsor, operating through this attested agent instance, authorized to perform this exact action on this target resource, under an active grant, within current risk parameters, and against a specific policy version?

That is why the unit of authorization is a tuple, not a badge. The left side of the decision holds two subjects that must not collapse into each other. The human or sponsor is the user, customer, process owner, or upstream agent that delegated the work. The agent or workload is the blueprint, the deployment, the instance, and the runtime evidence that proves the software is the approved one. Aembit's glossary calls the fusion of those two planes a blended identity: the agent's cryptographically attested workload identity plus the human currently instructing it. When no human is present, a process owner, a service identity, or an upstream agent still fills the sponsor slot, so the decision never becomes "the software, acting as itself, with no one to name."

The center of the decision binds those subjects to a grant. Delegation has to be a first-class input: what was given, how far it may hop, and whether this hop attenuates. Resource and action pin the audience: this tool, this operation, this object, not "whatever the model selects next." Task and approval capture the business-process control the previous section described: a payment, a merge, or a destructive change, bound to the exact action rather than a chat confirmation. Time, risk, and posture play the same role Conditional Access plays for people: geography, maintenance window, runtime health, and incident identifier. Policy version is what lets you reconstruct the decision six months later.

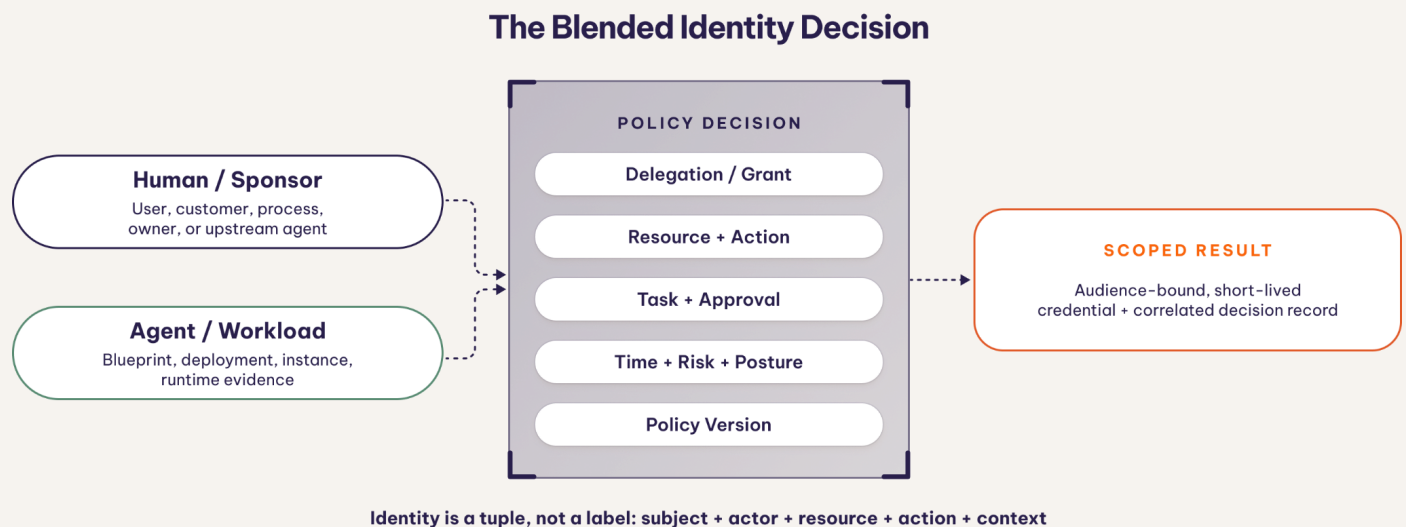


The only output that should leave that decision is an audience-bound, short-lived credential and a correlated record. The agent does not inherit the user's standing token, and it does not receive a shared service-account key. It receives a credential that is valid for that audience, for a short lifetime, and that you can join in the SIEM to the human, the workload instance, the grant, the policy version, and the outcome. Aembit's documentation is explicit that blended identity is not impersonation. The agent is not allowed to become the user. Policy evaluates both identities independently, so Ken on an approved client, Ken on an unapproved client, and another employee on the same approved client are three different decisions.

If any of those inputs is missing, the system should not issue a scoped result. A gateway that validates a JWT without them is only enforcing a label.

Figure 3 makes the unit of authorization explicit. A decision you can defend binds the human or sponsor, the attested agent, the grant, the resource and action, current risk, and policy version.

FIGURE 3



Aembit's *Control Plane*

Aembit provides identity and access management for software that calls other software, agents, and deterministic workloads alike. An Agent or Client Workload starts the access. A Server Workload is the target. A Trust Provider validates the workload's evidence. An Access Policy connects them, and Access Conditions add context. A Credential Provider obtains or mints the credential the server already accepts: a cloud federation token, OAuth credential, OIDC or JWT, SPIFFE SVID, vault token, API key, or another type the target requires. Aembit Edge intercepts or mediates the request. Aembit Cloud decides and coordinates credentials. The caller never stores a long-lived master secret.

For agents, three capabilities matter most.

Blended Identity evaluates the attested agent together with the human, or attests an autonomous agent as a single workload identity when no user is present. Aembit uses the strongest identifier the environment offers. Where a managed consumer agent such as Claude Desktop does not yet expose an OIDC endpoint or account token, a registered redirect URI can identify the approved client product. Where a custom agent runs on Kubernetes or a cloud VM, Aembit uses the service-account token or instance metadata document. The common weak design is the opposite: accepting one kind of signed client identifier everywhere. Real environments need many identifiers and should lean toward the most secure one available.

The MCP authorization service issues Aembit access tokens according to policy after the user authenticates to the enterprise identity provider. Customers using the managed MCP Identity Gateway do not deploy a standalone authorization server; Aembit Cloud wires authorization into the flow. For downstream user-specific SaaS access, Aembit supports per-user OAuth credential storage and refresh. For shared-service scenarios, it can use an administrator-authorized credential.



The MCP Identity Gateway is an optional data-plane component. Agents connect to it as if it were an MCP server. It validates the Aembit-issued token, requests a policy decision, routes to the permitted server, filters available MCP tools, obtains the downstream credential, and proxies the request. The gateway returns 401 for unauthenticated requests and fails closed on a deny. Agents never receive the downstream credentials. The gateway retrieves credentials for the server on each request, caches them briefly, and does not write them to disk. Events capture agent identity, user identity, target, and policy outcome.

Aembit is designed to complement AI gateways, not to replace them. Organizations may run more than one enforcement point. Aembit contributes attested identity, policy context, and time-limited credentials, and a gateway such as Kong, Apigee, or agentgateway can apply traffic controls, routing, and analytics. The architectural value is a consistent agent-trust layer across those enforcement points.

The architectural value is a consistent agent-trust layer across those enforcement points.



Aembit does not replace an MCP server's or application's record-level authorization, and that boundary is deliberate. The identity layer decides whether an attested agent, for a particular user or process, may call a server or tool under current conditions. The destination still enforces row-level access, transaction limits, workflow state, and domain invariants. Aembit policies can evaluate identity and contextual signals such as the user, the workload, risk posture, geography, and time. Those controls should fit the use case. They should not imply that every possible tool argument is always a universal policy key.

Aembit's identity-first focus complements specialized content-security controls. Organizations can combine identity and access policy with third-party prompt and content inspection, including security signals from tools such as CrowdStrike, so that identity, content controls, and gateway enforcement reinforce one another rather than compete.



How Identity and Enforcement *Work Together*

Centralized identity does not require centralized traffic. The control plane governs principals, delegated authority, policies, and credentials. Enforcement stays distributed across AI gateways, sidecars, service meshes, cloud token services, and applications. That is how mature enterprises already run human and API access: they keep authority in one place and enforce it at several Policy Enforcement Points, not all from the same vendor.

Gartner's public research on IAM for AI agents treats identity issuance, workload identity, credential management, authorization, governance, and gateway enforcement as separable capabilities that must be composed. NIST's 2026 concept paper similarly frames agent identity, authorization, audit, and non-repudiation as infrastructure concerns. The MCP specification separates the resource server from the authorization server and prohibits token passthrough. That prohibition is a confused-deputy control. A token minted for one audience, replayed by the server that received it, spends authority the caller never granted for that hop, and the downstream log records the relaying server instead of the original requester. Each hop should present its own audience-bound credential. The standards stack is still assembling, but the architectural split is already clear [4-6].

Three short patterns show how the split works in practice.

An employee research agent runs in Kubernetes and reaches tools through Kong. Aembit attests the workload, maps it to the approved agent, and binds it to the employee's current identity-provider session. Policy permits read-only document tools for that business unit and write-only access to a drafts folder. Aembit issues a short-lived token. Kong verifies it with the identity layer, rejects untrusted identity headers, rate-limits, scans content, and routes. If prompt injection tries to exfiltrate the contracts, no credential exists for that audience.



A scheduled operations agent can restart a narrow set of non-production services without a live user. Aembit attests the instance from cloud or Kubernetes evidence. Policy checks environment, version, incident identifier, ownership, and maintenance window, then issues a credential scoped to one restart tool and one namespace. Agentgateway handles MCP routing, CEL tool policy, rate limiting, and OpenTelemetry. A restart outside the namespace requires a separate approval-bound grant.

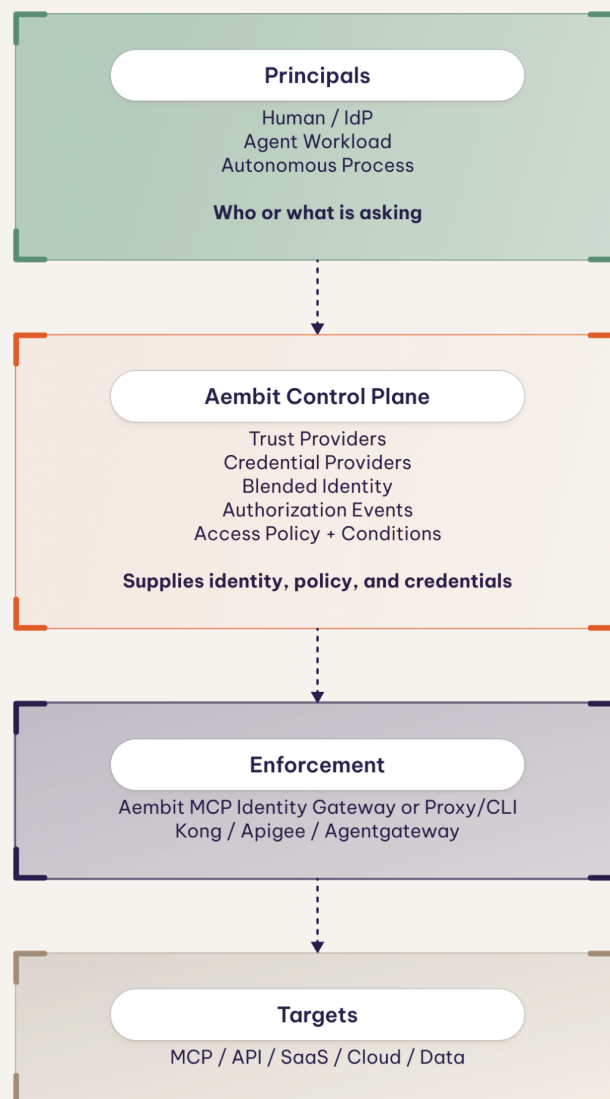
A customer-service agent reads accounts and may issue small credits. Apigee already fronts the APIs. Aembit preserves the representative, the agent workload, and the case identifier, then issues an audience-bound token. Apigee validates it, enforces rate limits, and logs operation-level analytics. The backend still applies row-level security. A larger credit triggers human approval bound to the exact amount and account. Aembit is not asked to provide the API catalog or monetization Apigee already owns.



Figure 4 shows that composition. A human from the identity provider, an attested agent workload, or an autonomous process enters the Aembit control plane as a principal. Trust providers, blended identity, access policy and conditions, credential providers, and authorization events produce the decision. Aembit supplies that principal evidence, blended identity, policy conditions, and credentials. You can enforce the decision with Aembit's MCP Identity Gateway, with another AI gateway such as Kong, Apigee, or agentgateway, or with a proxy or CLI. The targets remain MCP servers, APIs, SaaS applications, cloud services, and data.

FIGURE 4

Aembit in a Composed Architecture



Aembit supplies identity, policy, and credentials; the chosen data plane enforces and observes.



What To Ask of the *Architecture*

When you evaluate an architecture, start with what “agent identity” means and what attests the running workload. The system should preserve the human, process, or upstream-agent grant. A user token alone fails that test, and so does a shared service account.

Ask where the Policy Decision Point sits and how often it runs, at login, connection, token refresh, MCP request, or tool invocation. Aembit evaluates at issuance, refresh, and request. If a vendor says the decision is “continuous,” ask for the cache lifetime and whether the system fails closed.

Ask how many Policy Enforcement Points the design ends up with, because the answer is rarely one. An MCP identity gateway, an AI gateway in front of model traffic, a service mesh, and a proxy ahead of a legacy API can all sit on the same path. Ask whether every one of them resolves the same decision from the same control plane, what each does when that control plane is unreachable, whether any route bypasses them, and whether their logs join on the same principal and policy version. Enforcement in several places is normal. Policy invented in several places is not.

Ask whether credentials are short-lived, scoped, audience-bound, and hidden from the agent. Confirm that inbound MCP tokens are not passed through, that downstream credentials do not enter model context, and that a replay outside the intended target fails.

Ask whether the organization can reconstruct the action chain: the sponsor, the current agent, the workload instance, the delegated authority, the task, the resource, the tool, the policy version, the credential, the decision, and the outcome.

Ask which controls remain next to the identity layer: prompt defense, model routing, privileged approvals, and record-level business rules. A clear boundary is a sign of a mature design.



Conclusion

An AI gateway provides the point of control for live agent traffic, but an allow-or-deny decision is defensible only if the broader architecture can attest the software workload, preserve delegator provenance, enforce fine-grained authority constraints, evaluate real-time risk context, and mint purpose-scoped, ephemeral credentials.

AI agents make this trust chain non-negotiable. Their operational value lies in dynamic runtime planning; their primary security risk stems from holding static privileges broad enough to cover execution paths no human explicitly reviewed. Consequently, security architectures must abandon static session trust in favor of continuous, step-level verification: establishing runtime attestation and delegated authority, evaluating each consequential action, and injecting just-in-time credentials for immediate execution.

Aembit serves as the identity control plane for this paradigm—delivering workload attestation, blended agent identity, deterministic policy evaluation, and short-lived credential minting that integrates natively with existing enterprise infrastructure. Platforms like Kong, Agentgateway, and Apigee remain the designated traffic and routing planes; Microsoft, Okta, and CyberArk remain the core directory and privilege management planes. The identity control plane bridges these layers, establishing a unified source of truth before traffic hits the wire.

Architectural planning should not start with “We need an AI gateway.” The foundational requirement is: “We must establish who is acting, verify their delegated authority, bound their execution context, and enforce policy through an un-bypassable point of control.”

We must establish who is acting, verify their delegated authority, bound their execution context, and enforce policy through an un-bypassable point of control.



References

[1] Aembit. IAM for Agentic AI. 2026.

<https://aembit.io/resources/iam-for-agentic-ai-aembits-approach-to-closing-the-ai-identity-gap/>

[2] Aembit Documentation. What is Aembit? (including linked documentation for Blended Identity, Access Policies, Credential Providers, and the MCP Identity Gateway). Accessed 13 August 2026. <https://docs.aembit.io/get-started/>

[3] Aembit and Cloud Security Alliance. The Identity and Access Gaps in the Age of Autonomous AI. 2026.

<https://aembit.io/resources/the-identity-and-access-gaps-in-the-age-of-autonomous-ai>

[4] Gartner. IAM for AI Agents: A One-Size-Fits-All Strategy Is Guaranteed to Fail. 16 June 2026.

<https://www.gartner.com/en/documents/8009869>

[5] NIST National Cybersecurity Center of Excellence. Accelerating the Adoption of Software and AI Agent Identity and Authorization Concept Paper. 5 February 2026.

<https://www.nccoe.nist.gov/publications/other/accelerating-adoption-software-and-ai-agent-identity-and-authorization-concept>

[6] Model Context Protocol. Authorization. Specification version 2026-07-28.

<https://modelcontextprotocol.io/specification/2026-07-28/basic/authorization>

[7] IETF. RFC 8693, OAuth 2.0 Token Exchange. January 2020. <https://www.rfc-editor.org/rfc/rfc8693>

[8] SPIFFE. SPIFFE Identity and SVIDs. <https://github.com/spiffe/spiffe/blob/main/standards/SPIFFE-ID.md>

[9] Kong. Kong AI Gateway. <https://docs.konghq.com/gateway/latest/ai-gateway/>

[10] agentgateway. Agent Connectivity Solved. <https://agentgateway.dev/>

[11] Google Cloud. Model Context Protocol (MCP) in Apigee overview.

<https://cloud.google.com/apigee/docs/api-platform/apigee-mcp/apigee-mcp-overview>

[12] Microsoft. Microsoft Entra Agent ID documentation. <https://learn.microsoft.com/en-us/entra/agent-id/>

[13] Okta. Understanding Cross App Access in Okta.

<https://support.okta.com/help/s/article/understanding-cross-app-access-in-okta>

[14] CyberArk. Secure AI Agents. <https://www.cyberark.com/solutions/secure-agentic-ai>

