AEMBIT WHITEPAPER

IAM for Agentic AI: Aembit's Approach to *Closing the AI Identity Gap*

aembit

Gartner COOL VENDOR 2022

CSO Cybersecurity startups to watch for in 2023

2024 WINNER CYBER SECURITY EXCELLENCE AWARDS

RSAC Innovation Sandbox 2024 FINALIST

2024 SINET16 INNOVATOR AWARD

Summary

The Problem

Every agent your teams deploy today is running under someone else's identity — a shared API key, a human's full session, or nothing distinct at all. That's not a gap that closes itself: 97% of security leaders expect a material agent-driven incident in the next year, and most can't yet tell you, after the fact, whether it was a person or an agent that took action.

Aembit's Approach

Aembit is the identity control plane that sits between every agent or workload and everything it needs to reach — deciding, at the moment of each request, whether its identity is valid, what access level it gets, and which credential the destination system requires.

That decision draws on blended identity (user + agent) plus real-time posture signals from the device or environment, evaluated together for fine-grained authorization rather than the agent inheriting access wholesale.

Aembit exchanges that decision for whatever credential the destination system actually accepts – the agent or workload never needs to store a static secret. It's the same model enforced consistently across cloud, SaaS, and on premises, and it deploys without displacing the human identity provider or gateway already in place.

Benefits

Every agent action becomes attributable, revocable, and least-privileged by default — not because of new processes, but because the architecture makes the old failure modes (leaked static credentials, over-broad authorization, no accountability) structurally harder to hit. And it deploys in weeks against infrastructure you already run, not a multi-quarter build.

Proof

A $300 billion investment firm rolled this out across their enterprise Claude deployment in two weeks, with under six hours of security team time — replacing stored long-lived secrets and full-identity impersonation with scoped, attributable access, on top of the Okta instance they already had.

Table of Contents

1. Introduction

Every morning, somewhere inside a Fortune 500 company, an AI agent is already logging into Salesforce, querying a financial database, or reaching into a codebase to open a pull request — on behalf of an employee who never typed a password to get there.

This isn't a future state. It's Tuesday.

Agentic AI has moved from demo to production faster than almost any enterprise technology in recent memory, and the identity and access infrastructure meant to govern it hasn't caught up.

97%

of enterprise security leaders expect a material AI agent-driven security incident within the next twelve months.

Enterprise security survey, 2026

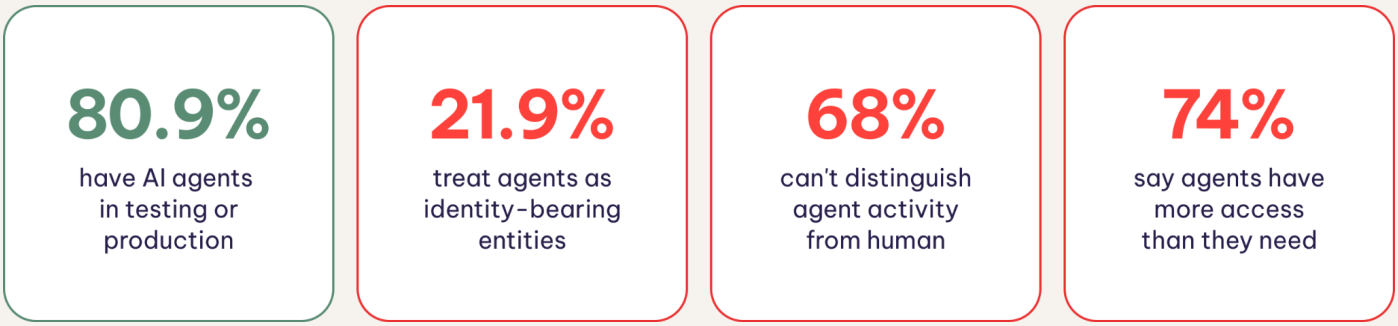
6%

of security budgets are allocated to address it.

Same population

That gap is the subject of this paper, and the scale of it is no longer speculative:

- Gartner's February 2026 cybersecurity trends report named **"IAM Adapts to AI Agents"** one of six trends with broad enterprise security impact this year, citing gaps in identity registration, credential automation, and policy-driven authorization for machine actors.
- A 2026 survey of 900+ security practitioners **found** 80.9% of organizations already have AI agents in active testing or production — but only 21.9% treat those agents as independent, identity-bearing entities. The rest rely on shared API keys or inherited service accounts.
- Aembit's own research with the Cloud Security Alliance found that 68% of organizations can't clearly distinguish agent activity from human activity, and 74% say their agents end up with more access than they need.
- Most tellingly: 97% of enterprise security leaders expect a material AI agent-driven security incident within the next 12 months, while only 6% of security budgets are allocated to address it.



We go into this in more depth in **why traditional IAM is no match for agentic AI**: workforce identity providers are built to authenticate a person logging into a session, and workload identity systems are built to authenticate a known, stable service. Agentic AI needs elements of both at once, not a replacement for either — the workload-side proof of what the software actually is, and the workforce-side proof of who it's acting for, evaluated together for software that spins up on demand, acts on behalf of a different user with each request, and can delegate part of its task to another agent entirely. Building on both is what the rest of this paper works toward.

A small set of recurring failure patterns sits underneath these numbers:

- Agents granted far more permission than any single task requires.
- Weak or nonexistent verification of the human behind a request.
- Credentials that live far longer than they should.
- Monitoring that shows what happened after the fact instead of stopping it in the moment.
- Prompt-based instructions that attackers can talk an agent out of following.

We cover each of these in depth later in this paper; the unabridged version, with real incidents attached to each, is **cataloged here**.

Not every agent carries the same risk. A customer-facing chatbot, an internal agent your finance team runs against Snowflake from inside your own Kubernetes cluster, and an agent embedded in a SaaS product you didn't build all need different things from an identity layer. Before we get to architecture, we introduce a simple framework for telling these apart — organized around who an agent acts for and where it actually runs — because that's what determines the right identity model, especially once you're operating across the mix of clouds, platforms, and SaaS products most enterprises actually run.

Aembit's answer is to act as a Workload Access Management control plane — a single access broker for both AI agents and traditional workloads that:

1

Establishes a Unique Blended Identity for Every AI Agent

Binding its workload identity to the human it's acting for, and a unique workload identity for every autonomous, non-agentic workload.

2

Grants Access Just-in-Time

Credentials scoped according to least-privilege policies, with tokens that are short-lived by default.

3

Tells the Difference Between a Human and an Agent

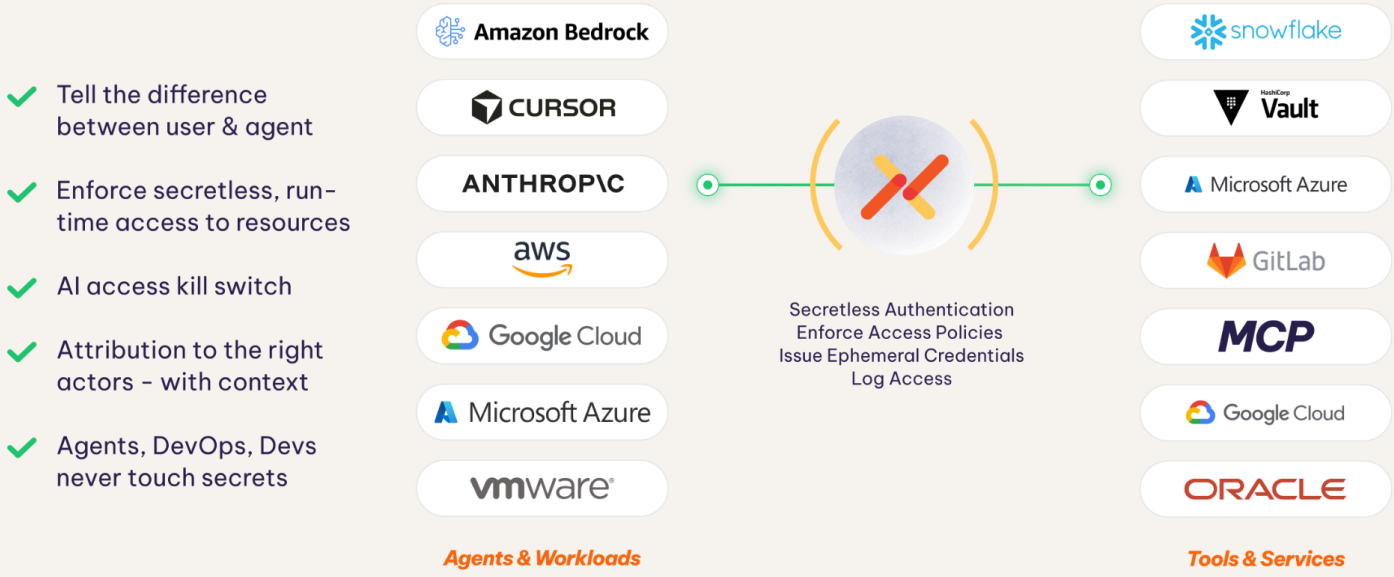
Both at the moment of access and after the fact in the audit trail.

4

Works Consistently

Across cloud, SaaS, and on-premises trust boundaries, instead of one environment at a time.

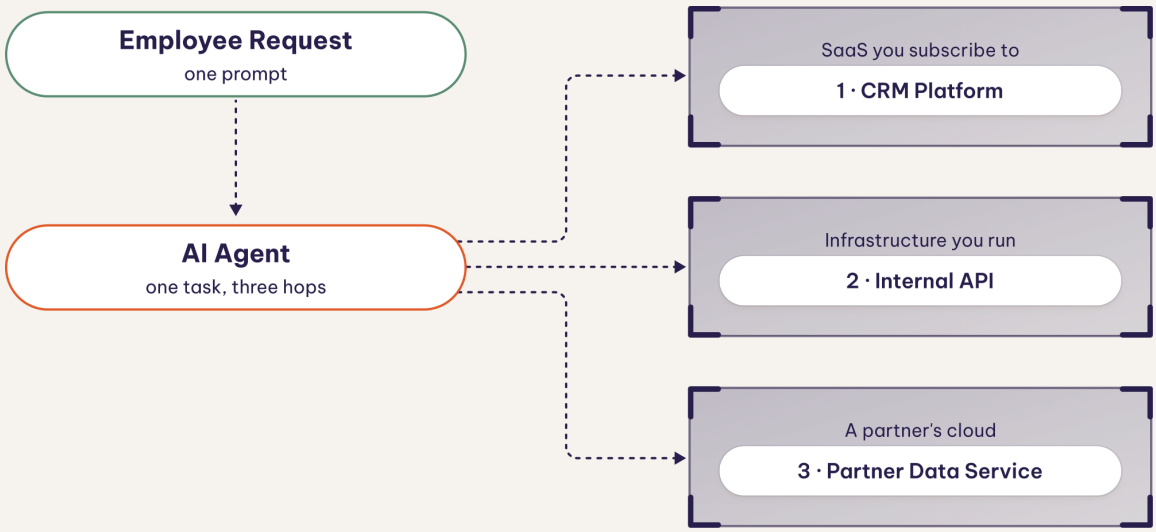
From there, we show how Aembit compares to the tools already sitting in your stack, why building this in-house rarely survives contact with a second environment, and how it holds up in production today — including inside one of the most heavily regulated corners of the economy, at a $300 billion investment firm that deployed Aembit at scale in two weeks, with under six hours of security team time required. The full CSA data behind the gap cited above is **available here** if you want to bring it into your own board conversations.



2. The Heterogeneous Enterprise

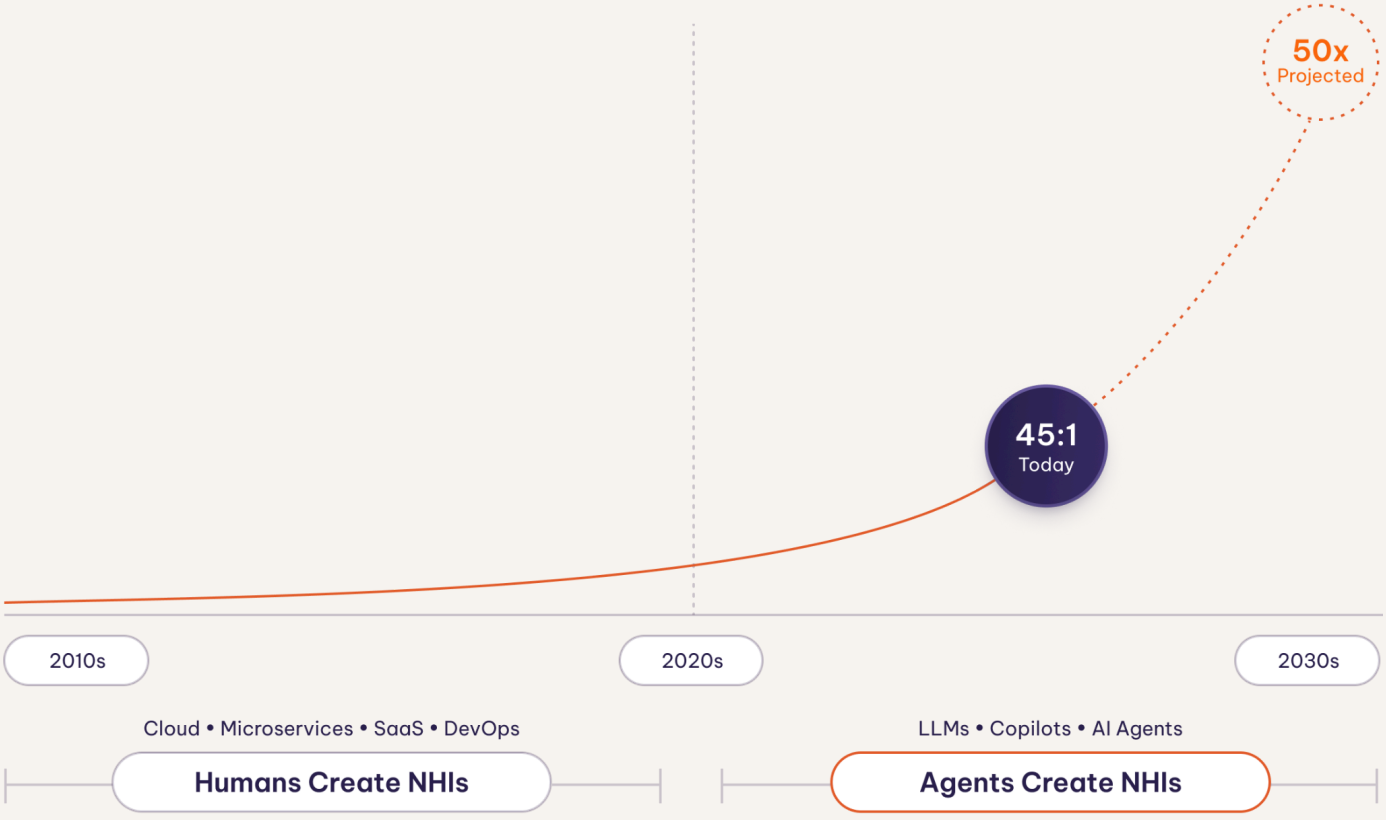
An enterprise adopting agentic AI doesn't get to deploy into one clean environment. Agents show up simultaneously as features inside AI platforms, as custom software your own teams build and run, and as capabilities embedded inside SaaS products you already pay for — and each of those agents reaches into a different mix of clouds, MCP servers, APIs, and data stores to do its job.

This heterogeneity isn't something agentic AI inherited from an older workload problem that happened to still be unsolved. It's a direct consequence of how agents work: a single agent task can hop from a SaaS platform, to an internal API, to a partner's cloud, all within one user request — something a traditional, single-purpose workload rarely had to do in one execution.



Agents don't arrive into a blank slate, either. They inherit an enterprise environment that was already fragmented before agentic AI showed up:

- Multiple clouds, on-premises infrastructure, SaaS applications, and partner systems, often serving the same business process.
- A **non-human identity** population that already outnumbers human users by as much as 45 to 1 at many enterprises.
- Access management handled differently by team, by environment, and by vendor, with no consistent policy layer connecting any of it.



None of the tools already sitting in most environments were built to handle this, agents or not:

- **Cloud provider IAM** works well inside a single cloud but breaks down the moment access needs to cross into another cloud, a SaaS product, or an on-premises system.
- **Vaults** are built to dispense a secret to whatever asks for it in the correct format — a model that gets especially dangerous once the thing asking is an AI agent. An agent holding a credential directly can leak it in a response, pass it along to another agent it delegates a task to, or be manipulated through prompt injection into handing it over outright. A vault can confirm a request was well-formed; it can't tell you whether the agent making it should be trusted at all.
- **Gateways** are designed to front applications and APIs, controlling throughput and routing requests. They **assume identity has been provided and verified.** Essentially, they outsource the IAM challenge.
- **DIY approaches** — manual configuration and allow-lists — **don't survive contact with scale**: they're built for one application at a time and tend to fall apart during an actual incident. For agents specifically, the DIY pattern is usually a custom-built OAuth integration wired up one agent at a time. That means no central control over how any of it is configured, and every implementation drifting slightly from the next, which turns routine maintenance, troubleshooting, and incident response into an outsized burden for developers and security teams alike.

Agentic AI doesn't wait for these gaps to close. It multiplies them, because a single agent, operating on behalf of a workforce user or an external customer, needs to be trusted across more of this fragmented landscape than any one traditional workload did. That's the reason identity for agents can't be treated as a special case bolted onto whatever access model already exists — it has to assume crossing boundaries is the default, not the exception. The next section introduces the framework we use to organize that reality.

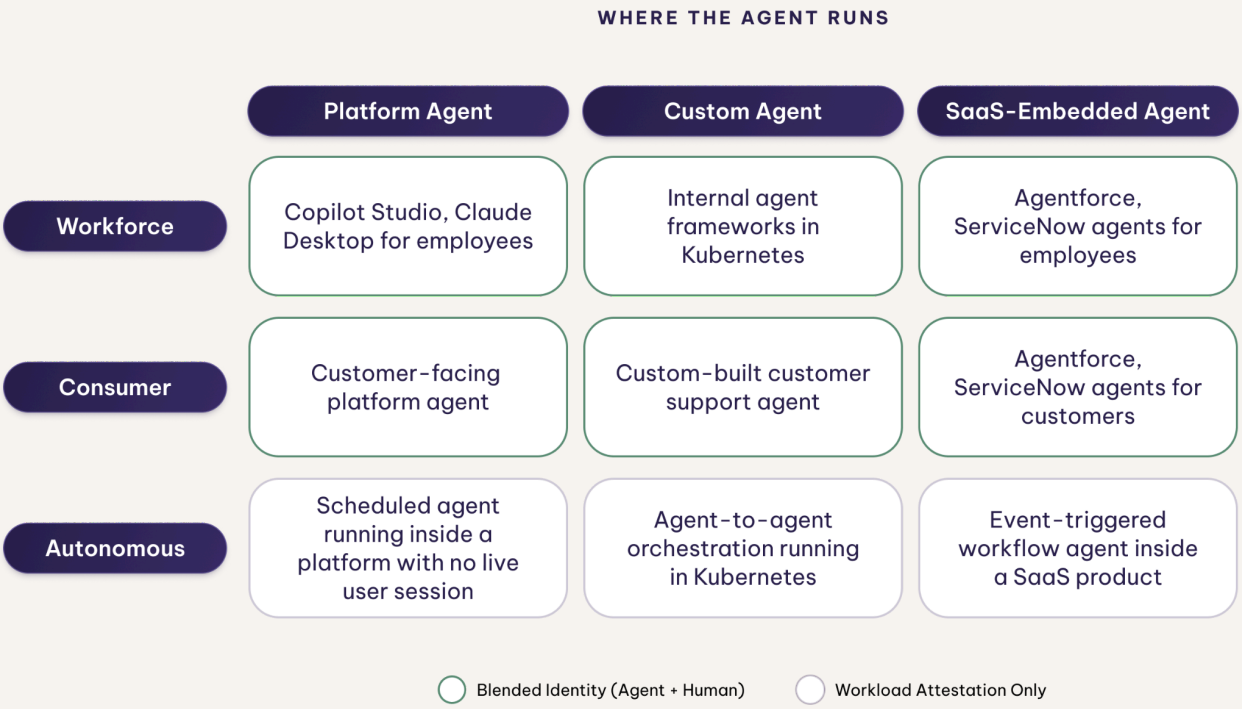
3. A Framework for Agentic Identity: *Two Vectors That Matter*

Treat AI agent identity as one undifferentiated thing, and you'll end up building a partial solution that fails somewhere in the critical path of each use case. The complete answer requires organizing around two vectors — who the agent acts for, and where it runs — and building a system that covers every combination, not just the one your first pilot happened to use.

The first vector is who the agent acts for. **Workforce agents act on behalf of an employee** — a Copilot Studio agent pulling a report for a finance analyst, a Claude Desktop session an engineer is using against internal tools. Consumer agents act on behalf of an external user — a support agent handling a customer's account, a chatbot embedded in a product your customers use directly. Autonomous agents act on behalf of no one in particular — they run on a schedule, respond to an event, or invoke other agents as part of a larger workflow, with no individual tied to any given request.

The second vector is where the agent runs. Platform agents run inside an AI platform or cloud service you didn't build yourself — Microsoft Copilot Studio, AWS Bedrock AgentCore, Gemini. Custom agents are software your own team builds and operates, typically in Kubernetes or on VMs. SaaS-embedded agents live inside a third-party product you already use — Salesforce Agentforce, a ServiceNow agent — where you don't control the runtime at all.

Cross these two vectors and you get nine identity problems, not one:



Each cell needs something different from an identity layer. A workforce platform agent needs **blended identity** — binding the agent's identity to the specific employee using it, so a policy can tell the difference between an engineer using Claude Desktop for Jira and someone trying to use it against a system they shouldn't touch. A SaaS-embedded consumer agent needs strong abuse and impersonation controls, since the runtime isn't yours to instrument and the biggest risk is an external party manipulating it. An autonomous agent orchestrating other agents in Kubernetes needs the opposite: there's no human session to bind to, so its own workload attestation has to carry the entire access decision on its own. We go deeper on how the identity model should change by scenario in **what kind of identity should your AI agent have**.

This grid is the system this paper builds toward, not just a way of describing the problem. When we get to use cases later, we map them back to it directly, and when we describe Aembit's architecture, we show which components cover which cells.

4. Why Traditional IAM Breaks for Agents — and *Blended Identity as the Answer*

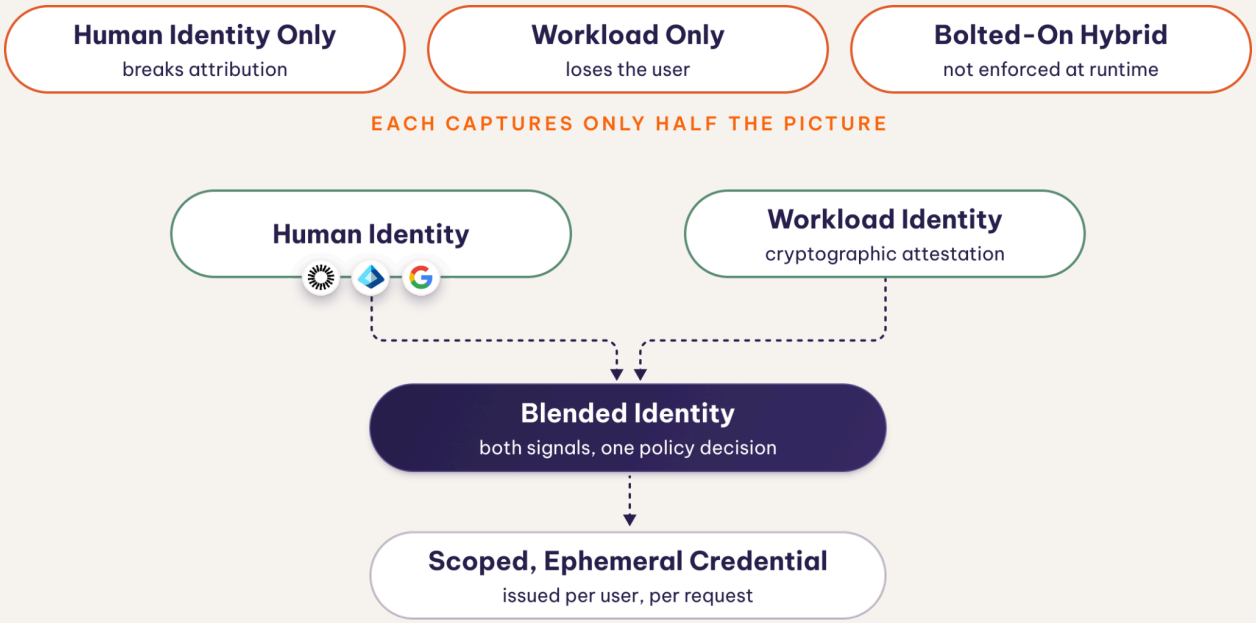
Section 1 introduced the core mismatch: workforce identity providers authenticate a person logging into a session, and workload identity systems authenticate a known, stable service. Neither was designed with the other in mind, because until recently nothing needed both at once. A human logging into Okta doesn't need a workload identity. A microservice authenticating to a database doesn't need a human behind it. AI agents break that separation, because a single request can carry both a workload that needs verifying and a person it's acting for.

Aembit's research into this describes **three ways an AI agent's identity tends to get modeled** in practice today, and each one is incomplete on its own:

Human Identity	The agent is handed the user's own credentials and impersonates them outright. This breaks attribution immediately: once the agent is acting under a human's login, you can no longer tell from the credential alone whether the person or the agent performed the action. It also hands the agent the human's entire access footprint, not just what the task in front of it actually requires — an agent impersonating a finance analyst to pull one report can reach everything else that analyst can touch.
Strictly Non-Human Identity	The agent is treated as a pure workload, authenticated like any other service account. This works for background automation, but it throws away the one signal that matters most for a user-driven agent: which specific person it's currently acting for.

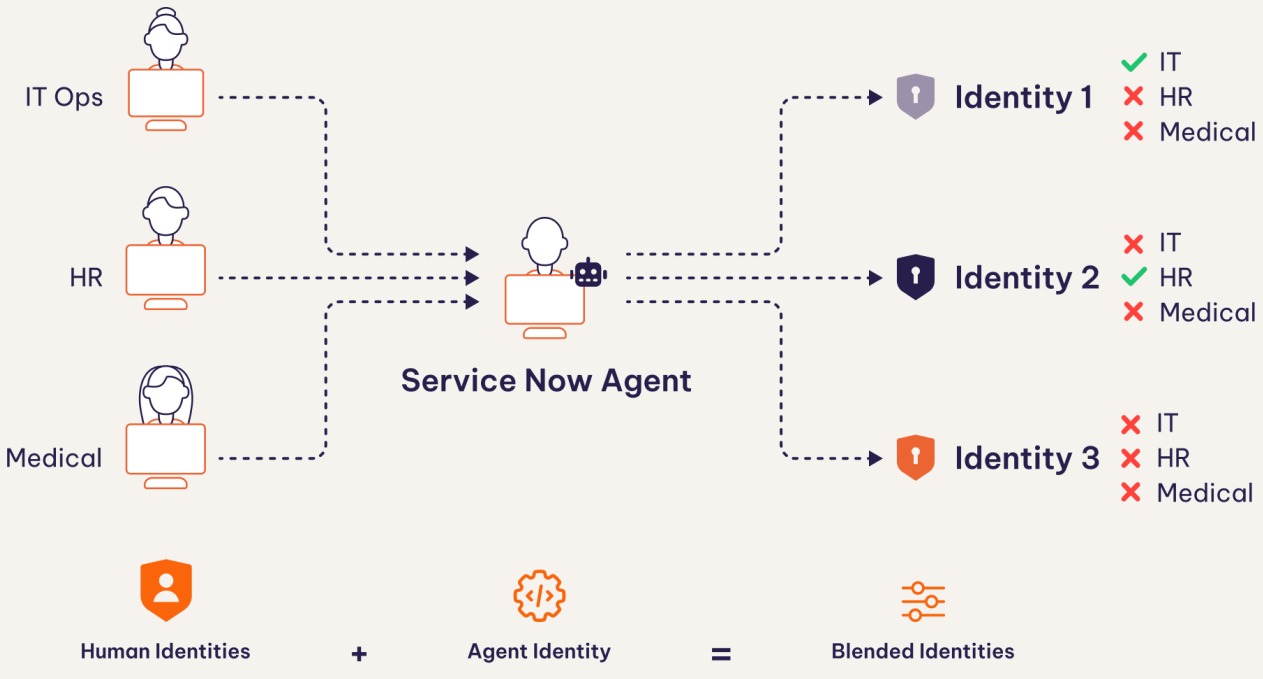
Hybrid Identity

Some vendors attempt to bolt a user field onto a workload credential after the fact, without evaluating the two together as part of the actual access decision. The identity exists on paper but isn't enforced at request time.



For workforce and consumer agents — the user-driven half of the framework in the last section — the answer is to evaluate both signals together, not pick one. That's what we call blended identity: an access model that checks the AI agent's workload identity and the identity of the human operating it in a single policy decision, made at the moment of the request. In practice, that supports policies like:

- Engineers can use Claude Desktop to reach Jira, but only the security team can use it to reach the vulnerability scanner.
- Each user gets their own scoped, short-lived credential when the same agent connects to a downstream system on their behalf — the agent itself never holds a credential broad enough to act as anyone else.
- One user's access can be revoked immediately without touching any other user's session or rotating a shared credential.

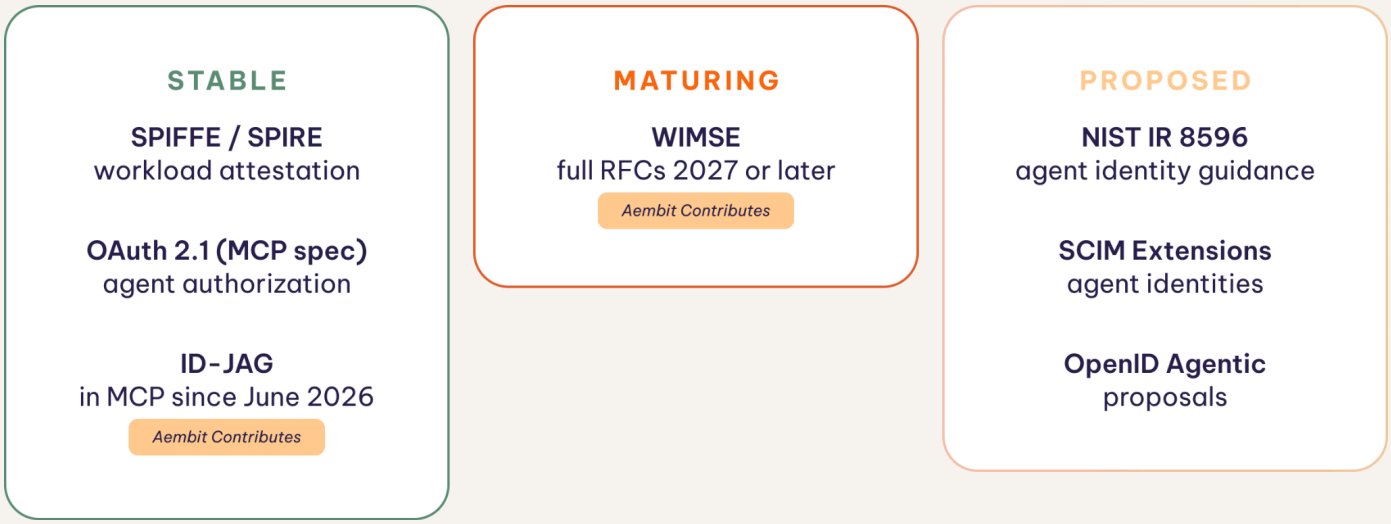


Autonomous agents, the third category from the framework, don't get this treatment because there's no human to blend in — their identity resolves to workload attestation alone, evaluated with the same rigor Aembit applies to any non-human identity.

None of this requires replacing your workforce or consumer identity provider. Aembit consumes the context you already have in Okta, Entra, or Google, and binds it to the agent's workload identity at the moment of the request, rather than asking you to stand up a second identity system. The payoff shows up directly in the audit trail: every access event carries both the human and the agent that acted, which is what makes it possible to satisfy **SOC 2, HIPAA, and PCI** requirements for agentic access in a way that a workload-only or user-only log never can.

5. Standards and Foundations — *Necessary, Not Sufficient*

Aembit's position on standards isn't neutral: we believe they're essential to shaping identity in the AI era, and we don't just implement them, we help write them. Our team is active in IETF discussions on agent and workload identity, and Aembit co-chairs the Identity Defined Security Alliance (IDSA) working group on Agent and Machine Identity. We support these standards in the product because the whole ecosystem benefits when the protocols improve, and we're positioned to help drive that rather than wait for someone else to.



A foundation, not a workflow — none includes a policy engine or a unified audit trail

Two standards (and a pseudo-standard) do most of the real work behind agentic AI identity today:

- **SPIFFE/SPIRE** — the CNCF standard for cryptographically attesting workload identity through short-lived, verifiable identity documents (SVIDs) issued within a trust domain. It's the closest thing to a foundation for the custom-agent cell of the framework in section 3: an agent running as your own code in Kubernetes or on a VM.
- **OAuth 2.1** — specifically the **authorization code flow defined in the Model Context Protocol specification**, which governs how an MCP client and server negotiate access today. This is the protocol Aembit's own MCP Authorization Service implements.

Workload Identity Federation (WIF) deserves a strong mention here. It's not a standard, but a widely used concept for cloud and AI platforms that allow third parties to exchange a verified identity token for a scoped access token. All providers (e.g., AWS, GCP, Azure, Claude, and OpenAI) implement WIF differently. This naturally presents challenges in implementation consistency and the ability to centrally control access.

A handful of others have moved further than most security teams realize. Aembit is closely tracking these proposals and working with customers to understand how they're implementing them in practice:

- **ID-JAG (Identity Assertion JWT Authorization Grant)** — progressing through the IETF OAuth working group, ID-JAG lets an identity provider hand an agent a scoped, short-lived token for another app's API by reusing the SSO trust relationship that's already in place, instead of a long-lived credential or a separate consent screen for every hop. It's no longer theoretical: MCP shipped enterprise-managed authorization built on ID-JAG in June 2026, and it's already running inside Claude, VS Code, and a

growing list of SaaS apps, with vendors shipping their own, varied implementations under the name Cross-App Access.

- **WIMSE (Workload Identity in Multi-System Environments)** — the IETF working group most directly aimed at the cross-domain problem from section 2. Its architecture, workload identifier, and workload credential drafts are moving through working group review now, with a dedicated draft on WIMSE's applicability to AI agents specifically, though the full stack isn't expected to reach RFC until 2027 or later.
- **NIST IR 8596, SCIM extensions for agent identities, and OpenID's agentic identity proposals** — round out the rest of the landscape. Directionally important, but none have progressed as far as ID-JAG or WIMSE.

Where Standards Stop Short

Even fully adopted standards give you a foundation, not an end-to-end workflow. SPIFFE gives a workload a cryptographically verifiable identity — it doesn't give you a policy engine to decide what that identity should be allowed to do, or a way to change that decision at runtime as conditions change. OAuth 2.1 gives an agent a scoped, short-lived credential for one MCP request — it doesn't give your security team a central place to see that access, correlate it across every agent and server in the environment, or tell a human's action apart from an agent's in the resulting log. ID-JAG is the clearest illustration of the limit: even now that it's shipping inside MCP, it solves the cross-domain token exchange, not the decision of whether that exchange should be allowed in the first place, or a single place to see it correlated across every agent and app it touches. Real building blocks, every one of them. None is an access management framework on its own.

The adoption problem compounds this. Standards only work once every party in a transaction has implemented them, and that's the same failure mode as **cloud provider IAM breaking down**

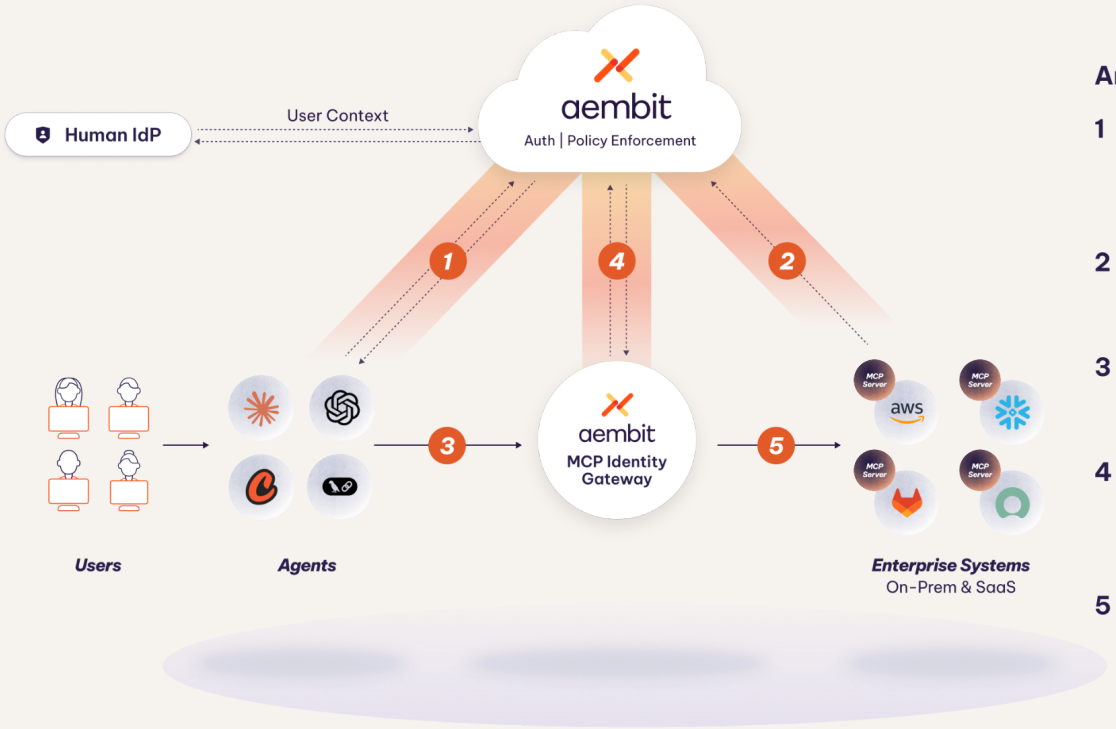
at boundaries, just applied to protocols instead of clouds. A 15-year-old on-premises application with a hardcoded service account was never going to add SPIFFE support. A SaaS vendor with no roadmap for the MCP authorization spec isn't adding it because your security team asked. You can mandate a policy internally; you can't mandate a vendor's roadmap, and you can't rewrite software nobody is maintaining anymore.

That's why enterprise IAM for agentic AI can't be "implement SPIFFE and OAuth 2.1 and call it done" — even coming from a vendor that helps shape those very standards. It has to broker across whatever each system actually speaks today — cryptographic attestation where it's available, credential injection where only a static key will ever exist — while still presenting one consistent policy and audit model on top, regardless of what's underneath. That's the architecture we cover next.

6. Architecture: How Aembit *Secures Agentic Access*

Aembit is a Workload Access Management control plane: the runtime layer that authenticates non-human identities, brokers the credentials they need, and enforces access policy across however many clouds, platforms, and services those identities have to reach. Gartner's April 2026 **Reference Architecture Brief: IAM for AI Agents and Other Workloads** names Aembit as an example Workload Identity Provider technology, in the same category as the workload-specific offerings from AWS, Microsoft, and Okta. The same research is direct about what that means for agents: "An AI agent strategy must be human-first and anchored in a wider Workload IAM reference architecture... their needs are similar to other workloads."

What follows is how that control plane is actually built — one architecture that governs AI agents and workloads through the same set of primitives.

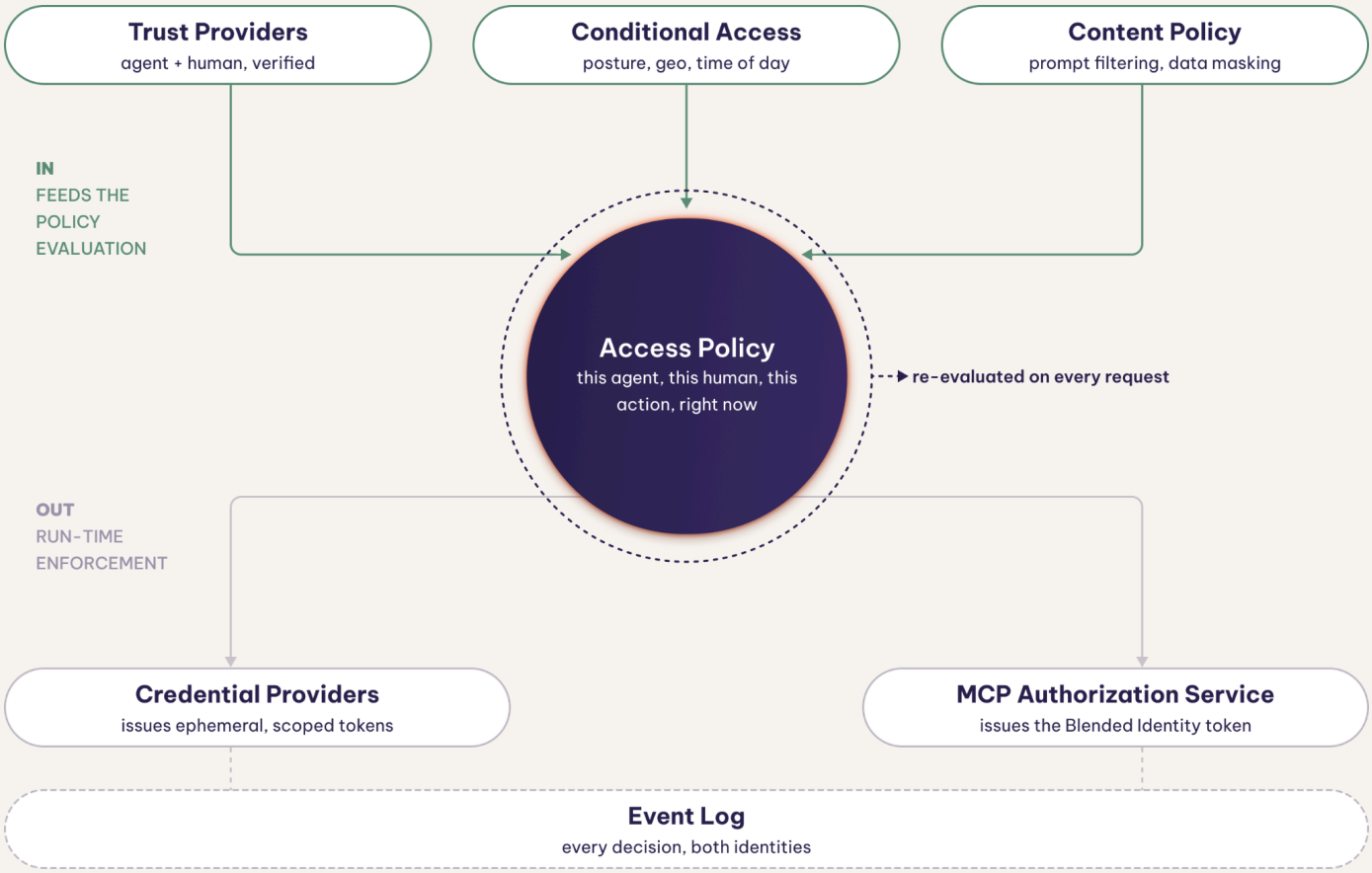


Architecture

1. **Authentication:** User and Agent authenticate to Aembit. Aembit returns a **blended identity token**
2. Aembit acquires **access credential for the service** (3LO)
3. Agent authenticates to Gateway using blended identity token
4. **Token Exchange:** Gateway exchanges blended identity token for access credential
5. Gateway uses access credential **to access the service**

The Core Model: Policy as the Organizing Principle

Every access decision Aembit makes, whether the requester is a microservice or an AI agent, is organized around one thing: the access policy.

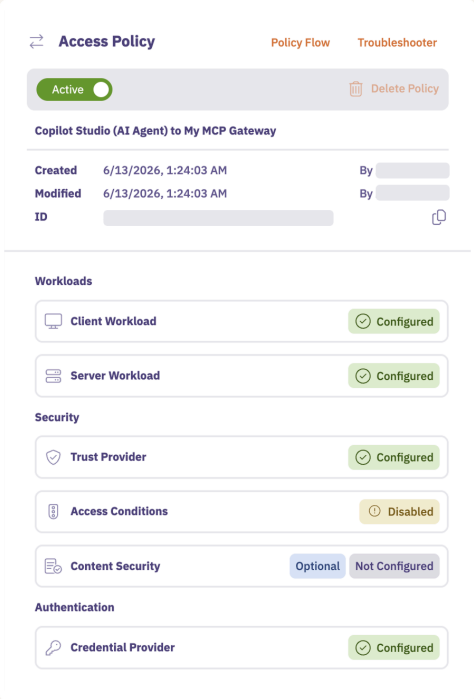


Everything else in the model exists to either feed that policy the context it needs, or to carry out what the decision produces:

- **Access policies** are where Aembit decides whether a request gets anything at all — evaluating who or what is asking against what they're allowed to do, optionally combined with real-time conditional access signals. This is the piece customers

actually configure and live in day to day; trust and credentials are what a policy consumes and produces.

- **Trust providers** establish who or what is actually asking. Aembit federates with the agent or client's own runtime environment and uses its attestation process to cryptographically verify both the agent and the human behind it at the moment of the request, not once at boot time and then trusted indefinitely. That verification is what feeds the policy decision, and depending on the runtime it might take the form of an AWS instance identity document, a Kubernetes service account token, a GitHub Actions or GitLab CI token, a SPIFFE SVID from your own SPIRE deployment, an OIDC token issued by an agent platform to attest to an AI agent's identity, or — for the human half of a blended identity decision — an OIDC or SAML assertion from Okta, Entra, or Google.



- **Conditional access** layers real-time posture into the same decision — device or workload security posture from CrowdStrike or Wiz, geolocation, time of day — so a policy can require more than just "this identity is valid" before granting access.

- **Content policy** allows you to route requests to a content filtering / prompt filtering capability as part of the access process. While Aembit natively provides the interception and policy enforcement, it leverages your existing prompt filtering tools like CrowdStrike AI-DR to provide the functionality.

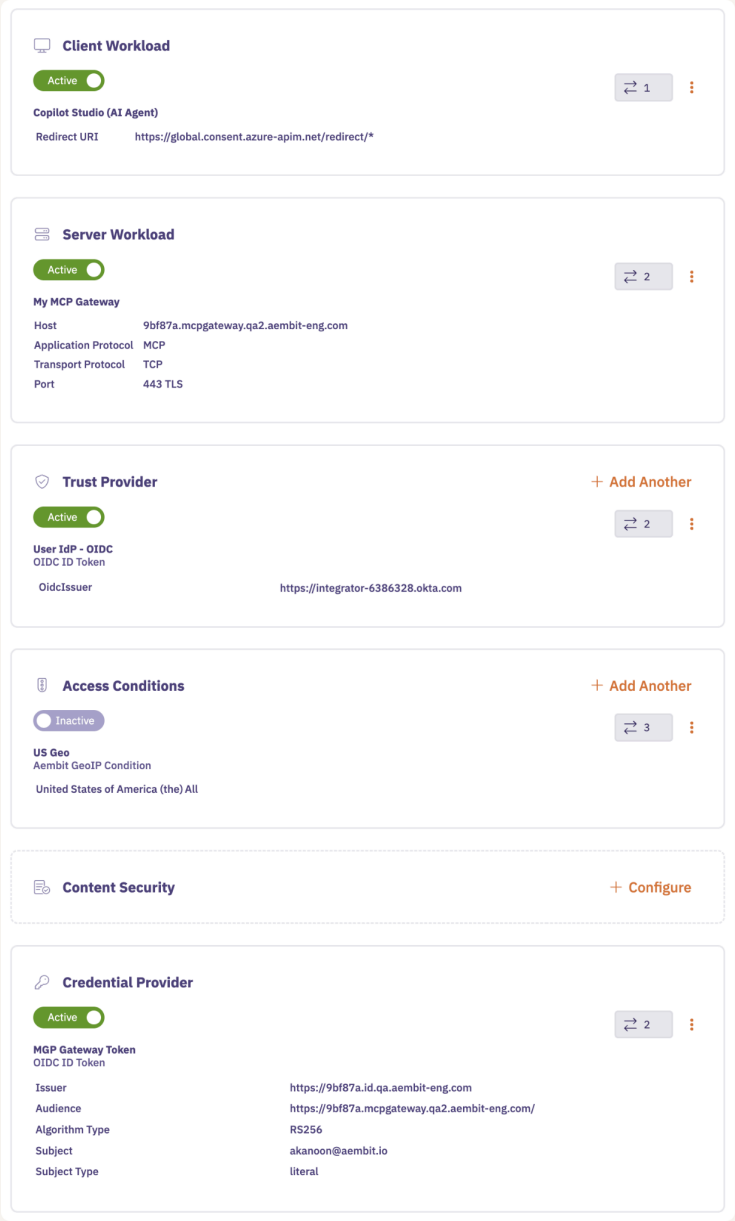
- **Credential providers** issue or exchange the credential the destination system actually accepts once a policy approves the request — a short-lived OAuth 2.0 token, a scoped API key, a workload identity federation exchange with AWS, GCP, or Azure — so the requester never needs to store one. Aembit can mint credentials and manage their lifetimes directly, request them from the target resource at the moment

of access, or take a "bring your own vault" approach and pull credentials from a third-party source you already run.

- **The MCP Authorization Service** applies Aembit's trust and credential-provider capabilities specifically to agents operating over MCP — it implements the OAuth 2.1 authorization code flow defined in the MCP specification, federating with the OIDC and SAML identity providers you already run to pull in who the agent is acting for and issuing the resulting token.

This is the model Aembit has run in production at Fortune 500 companies for workload IAM for years. Agentic AI didn't require a new architecture — it required extending trust providers, policy, and credential providers to a new protocol and a new kind of identity.

Each policy is a straightforward description of the agent environment, target resource, and the conditions that have to be met in order for Aembit to issue a short-lived, just-in-time credential at runtime.



Enforcing the Model at the Edge: Why Aembit Built the MCP Identity Gateway

A policy decision and a token only matter if something enforces them on the actual request. That's the job of the **MCP Identity Gateway** — it validates the signed identity token an agent presents, checks it against your access policies (including any conditional access rules), and exchanges it for the access credential the downstream MCP server actually accepts.

Traditional workloads never needed a dedicated gateway for this, and it's worth being precise about why. A workload is deterministic: it's built to do a fixed set of things, so the permissions it needs are constant, and validating one workload identity and injecting one credential per connection is enough to cover its entire lifetime. Enforcement there is exactly what the host-based proxy or SDK described later in this section already does. An agent isn't deterministic in the same way. What it does next depends on the prompt in front of it, so the permissions it needs shift from one action to the next, and the same flexibility that makes an agent useful is what can be pushed, through prompt injection or plain scope creep, into exercising permissions it was never meant to use on that particular task. That's a fundamentally different enforcement problem: not one identity validated once, but the same identity re-evaluated against policy on every individual action it takes.

MCP reflects that directly in its own design. It's an authorization protocol as much as a tool-calling one, built around an OAuth-issued token that's supposed to reflect a policy decision made per tool invocation, not once per connection, and that decision has to account for the human behind a blended identity as well as the agent. Enforcing that consistently takes a component built around MCP's own request shape rather than a generic proxy passing bytes through, which is exactly what the MCP Identity Gateway is: the same enforcement role Aembit has always played for workloads, purpose-built for a protocol where policy gets evaluated per tool call instead of once at connection time, because the requester itself isn't predictable the way a workload is.

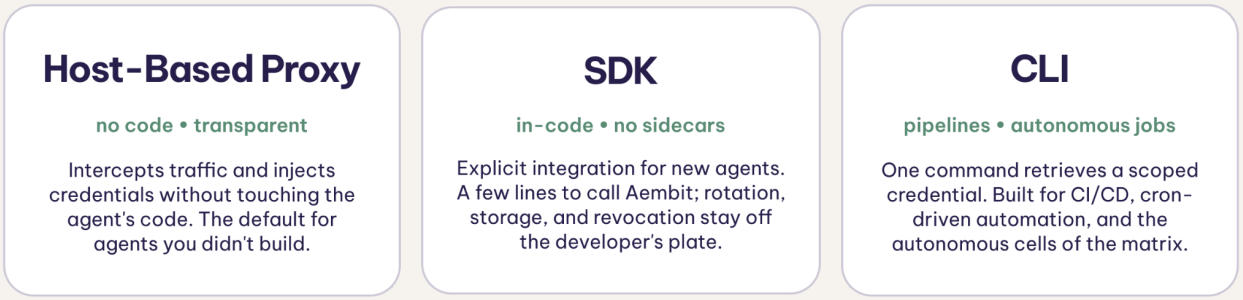
Aembit built this gateway to enforce IAM decisions on MCP traffic, not to sell a general-purpose API or AI gateway. If you already run, or would rather standardize on, a general-purpose gateway — Kong, Apigee, a dedicated AI gateway — Aembit integrates with it instead of asking you to replace it, enforcing trust attestation, policy, and credential exchange at that layer. Customers mix and match Aembit's gateway and general-purpose gateways as it suits their environment; what has to stay constant is that every one of those enforcement points is checking the same policy, against the same identity, through Aembit.

It's also critical to understand the reverse perspective: Why isn't a gateway sufficient to solve the identity problem? Gateways are designed to front applications and APIs, controlling throughput and routing requests. They assume identity has been provided and verified. Essentially, they outsource the IAM challenge to an access manager or control plane.

For a walkthrough of Aembit in action, see our demo on **securing MCP servers with Aembit**, which covers how blended identity fits into this architecture end to end.

How Agents Actually Connect: Host-based Proxy, SDK, and CLI

Aembit offers three integration surfaces for agents, designed for compatibility with deterministic workloads as well. Forcing one model onto all of them is exactly the kind of friction that gets a security control bypassed. The design goal behind all three was never just to cover every agent architecture — it was to take access and credential management off a developer's plate entirely, regardless of which surface they end up on.



- **Host-Based Proxy** – a transparent, no-code integration that intercepts traffic and injects credentials without touching the agent's code. A developer building or maintaining that agent never sees a credential, writes an integration, or changes a line of code to get secured access; it's invisible to them. The right default for existing or third-party agents you didn't build, and for custom agents where you'd rather not touch the codebase at all.
- **SDK** – explicit, in-code integration for teams building new agents from scratch who want fine-grained control over exactly when and how an access decision gets made, rather than relying on interception. The developer adds a few lines to call Aembit, but never handles a raw secret, builds credential storage, or writes rotation and revocation logic themselves – that work stays with Aembit.
- **CLI** – built for scripts, pipelines, and operational tooling rather than live agent traffic. The `aembit credentials get` command retrieves a scoped credential for a specific workload in a single call, so whoever wrote the pipeline never has to stand

up a vault client or hardcode a secret just to get the job done. The natural fit for CI/CD jobs, cron-driven automation, and the autonomous-agent cell of the framework, where there's no live session to intercept in the first place.

Which surface makes sense depends on the runtime column from section 3's framework — platform, custom, and SaaS-embedded agents each tend to reach for a different one — but in every case, the developer's job stops at picking the surface, not managing what happens after.

Event Logging: The Record Every Decision Leaves Behind

Every trust, policy, conditional access, and credential decision above is captured as a structured event, with full identity context, for audit and SIEM correlation. For agentic access specifically, that record does something a workload-only or user-only log can't: it carries both the human and the agent that acted on their behalf, so an auditor or an incident responder can always answer not just what happened, but who was ultimately accountable for it.

Aembit is natively integrated with the SIEMs and security platforms security teams already run — Splunk, CrowdStrike, Google, AWS, and more — so this record lands where your team is already watching, instead of becoming one more console to check.

How Blended Identity Actually Gets Evaluated

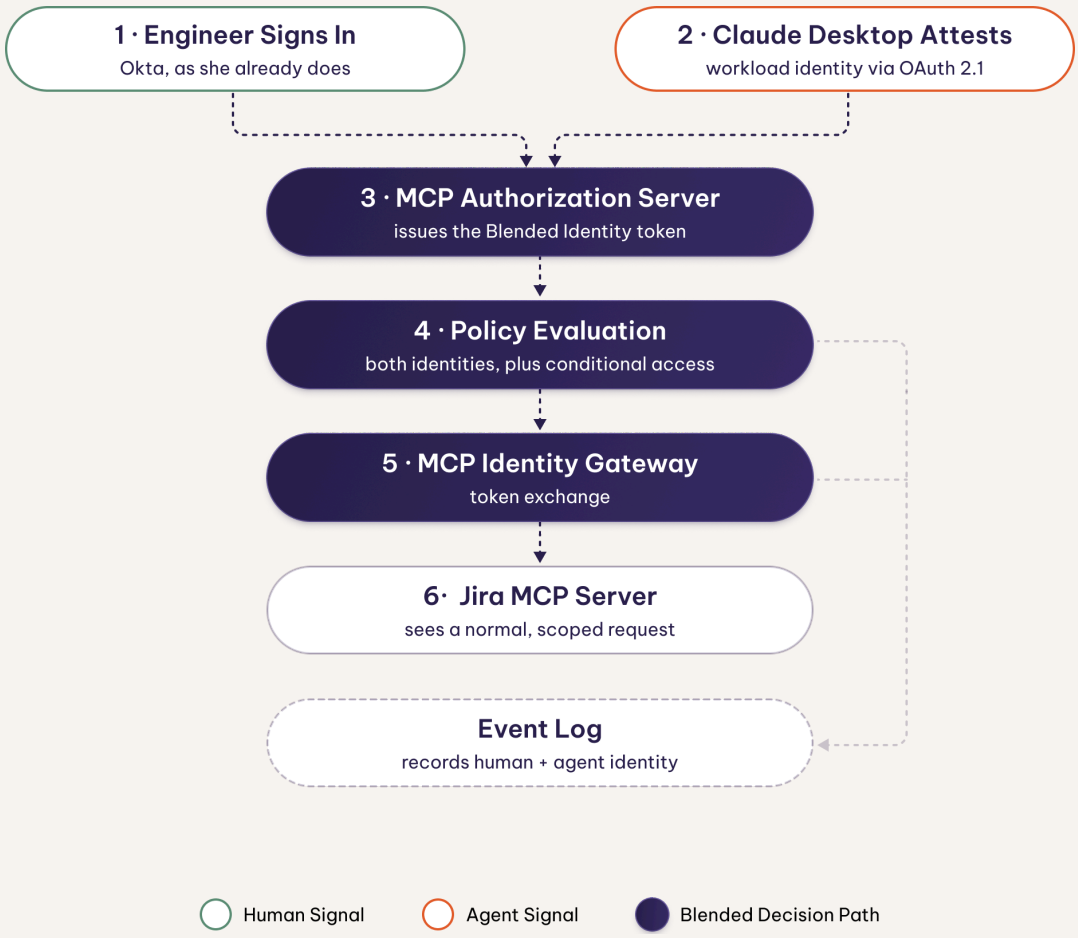
Blended identity is used whenever a policy decision has two identities to evaluate instead of one. When a workforce or consumer agent makes a request, Aembit invokes two trust providers: one validates the agent's own workload identity — an OIDC token from the agent platform, a SPIFFE SVID, whatever its runtime provides — and the other validates the human it's acting for, an OIDC

or SAML assertion from Okta, Entra, or Google. Both have to be resolved before the policy evaluates anything; an agent with a valid workload identity but no verifiable human behind it is treated differently than one with both, and a policy can require exactly that distinction.

The access policy then evaluates the pair together, as a single unit — "this agent, acting for this specific person, requesting this specific action" — rather than checking one identity and then the other. That's what makes a policy like "engineers can use Claude Desktop against Jira, but only the security team can use it against the vulnerability scanner" enforceable at all: same agent, same workload identity, different outcome, depending entirely on which human token showed up alongside it. Conditional access and credential issuance both happen after that combined decision, using whichever identity, or both, the resulting credential needs to carry.

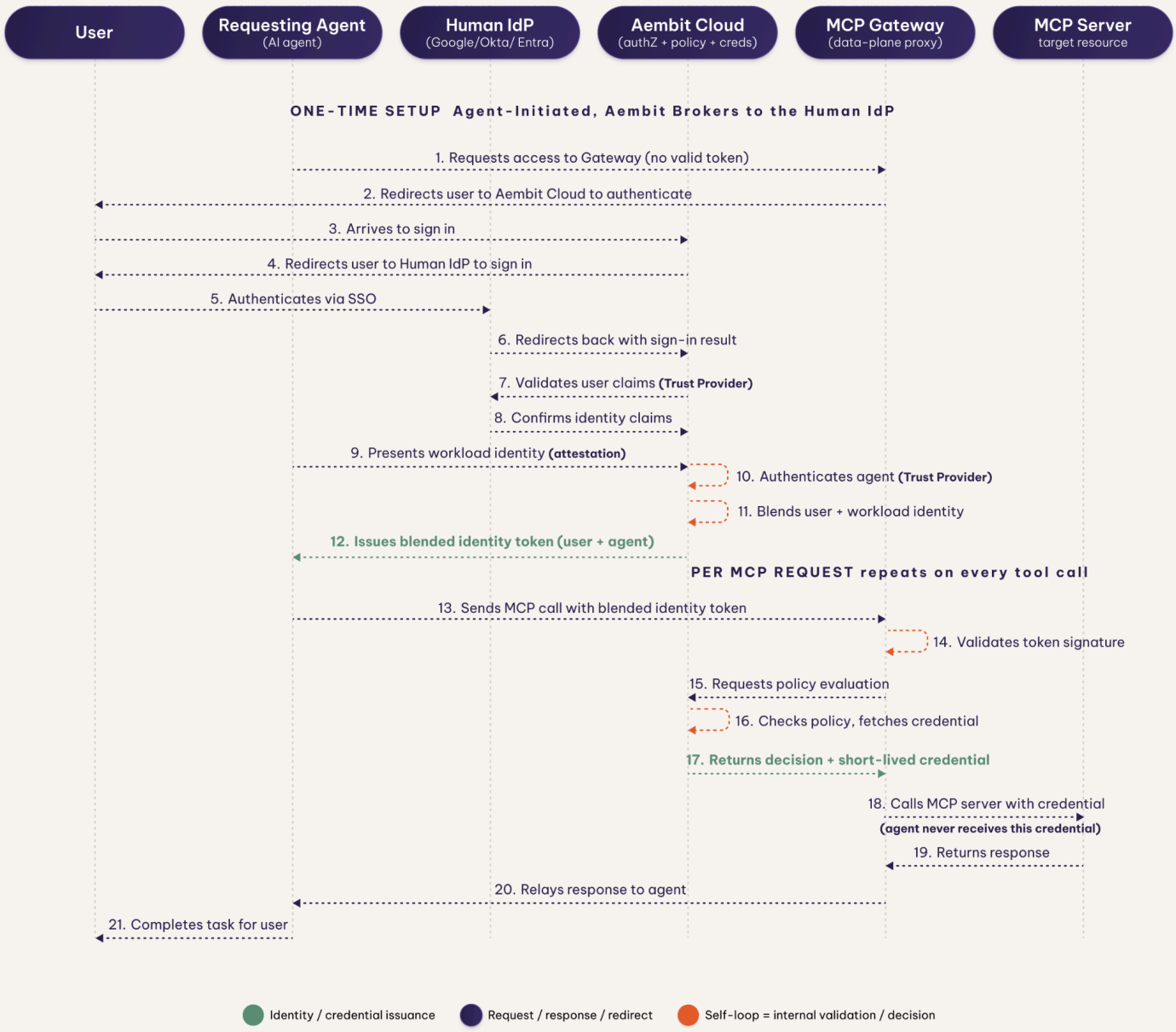
Concretely, that combined decision gets expressed as a single OIDC-based identity token, with custom claims added to carry both halves — one set of claims for the agent's workload identity, another for the human it's acting for — rather than two separate tokens a downstream system would have to reconcile on its own. Anything consuming that token downstream, whether it's a policy check further along the chain or the audit log, gets both identities in one place, already bound together.

A Request, Start to Finish



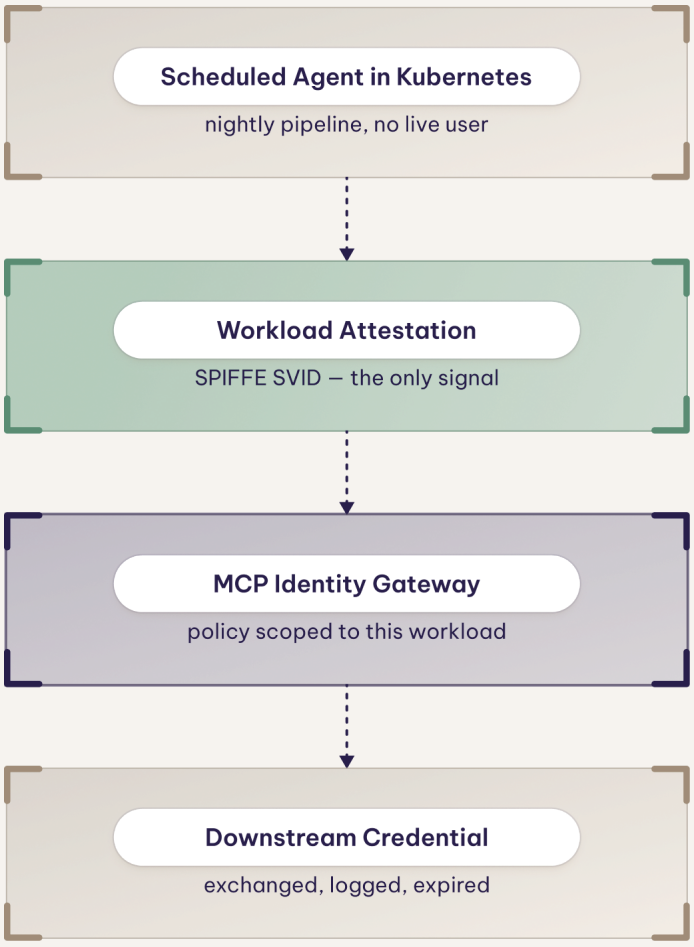
Take an engineer using Claude Desktop against Jira. She authenticates to her laptop and to Okta as she already does every day — that's the human identity half. Claude Desktop, acting as an MCP client, opens the OAuth 2.1 authorization code flow against Aembit's MCP Authorization Service, which federates with Okta to confirm who she is and validates Claude Desktop's own workload identity at the same time. Those two signals resolve into the blended identity described above, and the Authorization Server issues a scoped, short-lived identity token back to Claude Desktop. When Claude Desktop calls the Jira MCP server, that call passes through the MCP Identity Gateway, which validates the token, checks it against the access policy (including any conditional access rules, like device posture from CrowdStrike), and exchanges it for the

credential Jira actually accepts. Jira sees a normal, scoped API request; Aembit's event log sees both her identity and the agent's.



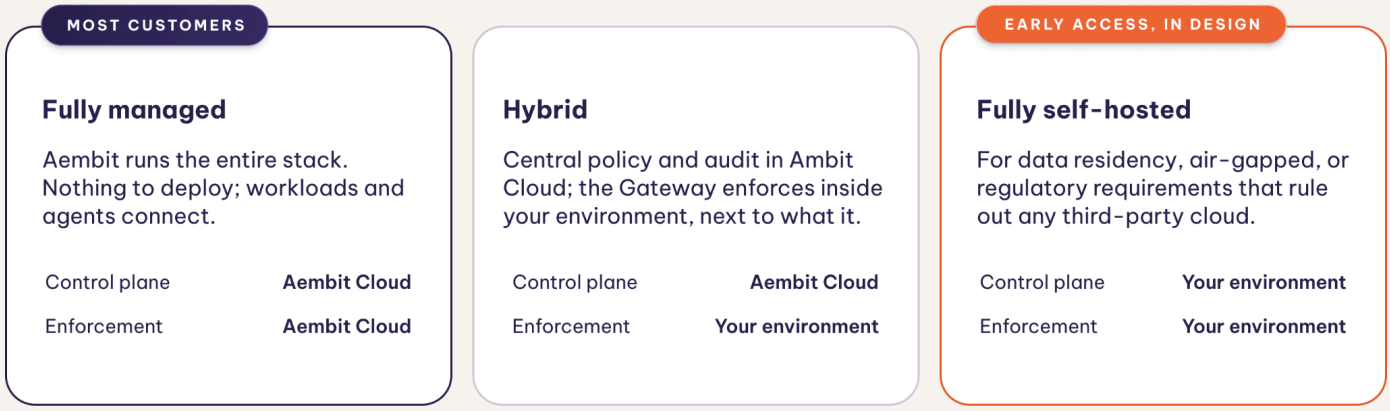
An autonomous agent looks similar with one less requirement. A scheduled agent orchestrating a nightly data pipeline in Kubernetes has no human session to blend in, so its SPIFFE SVID is the only trust provider that fires. The gateway validates that identity against a policy scoped to that

specific workload, exchanges it for the downstream credential, and logs the event — the same enforcement path, just without a human half to combine.



Deployment Flexibility: Aembit Cloud and Self-Hosted Enforcement

Aembit doesn't force a single deployment model on every customer — it offers three, and which one an organization picks depends on how much of the stack it wants to operate itself versus hand to Aembit.



- **Fully managed** – Aembit runs the entire stack, control plane and enforcement alike, as a hosted service. Policy administration, the MCP Authorization Service, and the MCP Identity Gateway all run in Aembit Cloud, with nothing to deploy on your side; your workloads and agents simply connect out to it. The right fit for teams that do not want to operate IAM infrastructure themselves.

- **Hybrid: cloud control plane, self-hosted gateway** – Aembit Cloud hosts the control plane– policy administration, the MCP Authorization Service, and the audit trail every event lands in – while the MCP Identity Gateway deploys inside your own environment, so interception, policy enforcement, and credential exchange happen next to the workloads and agents they're protecting. It's the same centralized-governance-with-decentralized-enforcement pattern Gartner's own research documents as best practice for workload IAM.

- **Fully self-hosted** – both the control plane and the enforcement layer run inside your own environment, for organizations with data residency, air-gapped, or regulatory requirements that rule out any dependency on a third-party cloud, even for policy administration. (As of Q2 2026, fully self-hosted is in design. Contact Aembit to participate in our early access program.)

Across all three, it's the same architecture and the same policies. What changes is where each piece physically runs, not what it does.

The Full Capability Stack

Beyond the core architecture, the rest of what's available in the product day to day falls into two categories: enforcing access and authorization at the moment of the request, and operating Aembit across an organization at scale.

Access and Authorization Enforcement

Kill Switch	Revoke a specific agent's access, or every agent's access to a specific system, in one action, without rotating a shared credential everyone else also depends on.
Content and Prompt Security	Aembit doesn't build this directly; it depends on specialist partners to deliver it within the framework of an Aembit policy. A policy can route an authenticated request to a partner like CrowdStrike's AI-DR product for content-policy enforcement, rather than Aembit trying to be a content-inspection engine itself.
Tool-Level Filtering **(In beta)**	Restrict which specific tools or actions an agent can invoke within a given MCP server, rather than granting or denying access to the entire server at once.
Human-in-the-Loop Approval **(In beta)**	Require explicit human user sign-off before an agent completes a specific high-risk action, rather than relying on the policy decision alone.
Desktop / Endpoint Support **(Request early access)**	Enforce the same trust, policy, and credential model on agents running directly on an employee's laptop, like Codex or OpenClaw, not just on server-side workloads in the cloud or on premises.

Operating Aembit at Scale

Policy as Code	Manage access policies through Terraform or Aembit's API, so policy changes go through the same review and deployment process as everything else your team ships.
Resource Sets	Group workloads, agents, and destinations so a team can manage and enforce its own slice of the environment day to day, while security still administers policy and visibility centrally across every group at once. Centralized enforcement, decentralized operations — the same divide as the deployment models above, applied to how teams actually work in the product rather than to where components run.
Security, Reliability, and Scale	How Aembit itself stays available, secure, and performant as a control plane sitting in front of every access decision. This is critical enough to warrant its own treatment; the next section covers it in full.

7. Reliability, Scalability, and Security by Design

Aembit has been running in production, at scale, inside some of the most heavily regulated companies in the world, for multiple years. This isn't a new architecture rushed out to catch the agentic AI wave — it's the same control plane that's been protecting workload access at Fortune 500 companies since before "agentic AI" was a category, now extended to cover agents too.

SOC 2 Type II	ISO/IEC 27001:2022	99.99%	Billions
independently audited	certified	availability target, cell-based multi-region control plane	of transactions supported in production today

On scale: **Aembit's architecture already supports billions of transactions** today, built on a control-plane model designed to scale by orders of magnitude further without expanding what any single tenant is exposed to. That's the same architecture from section 6 — a managed control plane paired with lightweight enforcement points — not a monolith that has to grow linearly with every workload or agent you add.

On security: Aembit is SOC 2 Type II and ISO/IEC 27001:2022 certified, backed by independent penetration testing, data encryption in transit and at rest, strict access controls, and secure tenant isolation. Visit our **Trust Center** for up-to-date information on our compliance status and performance.

Reliable by Design

Certifications and quotes describe the outcome. The engineering underneath them is what actually produces it:

Multi-Region, Active-Active, Not Just Failover

The control plane runs as multiple independent cells across regions, all simultaneously serving live traffic rather than one primary region backed by a cold standby. A cell is a self-contained unit of key resources that make up the Aembit control plane, and those cells can be replicated across regions or even within a cloud region. Latency-based health routing sends each request to the healthiest, closest region automatically, so the same architecture that protects against a regional outage also cuts latency for whoever's nearest to it. Four-nines availability is the explicit target for that layer, and growing that regional footprint further, so more customers sit closer to a live region over time, is an ongoing commitment rather than a one-time build.

Enforcement That Runs Next to the Workload

Aembit Edge runs the proxy locally, alongside your workloads and agents — VMs, containers, bare metal — keeping enforcement on the same network hop as the request instead of routing through a distant intermediary, and scaling horizontally with your own infrastructure rather than a shared tier. It also caches each short-lived credential for its validity window, so repeated calls only need a fresh round trip once that credential expires.

Continuous Self-Testing of the Live Request Path

Automated checks run every minute, in every region, actually performing the login flow, the credential retrieval, and the policy evaluation a real customer would — not just pinging an endpoint — layered on top of nightly stress testing and regression and performance benchmarks on every single code change.

Progressive Rollouts That Shrink the Blast Radius of a Bad Release

New versions bake in on a small slice of traffic first, with a deliberate observation window, before gradually shifting the rest of production over, replacing what used to be an instant, all-at-once cutover.

Deliberate Reduction of Dependency Surface

Aembit caches critical third-party data, like identity provider signing keys, locally, so an outage in a customer's own identity provider doesn't take down Aembit's ability to attest trust. The underlying engineering principle: every new external dependency is a potential failure point, so they're added sparingly and only when they earn their place.

Secure by Design

Reliability describes what keeps the control plane up. Security describes what keeps it trustworthy while it's up, and that shows up in the same kind of unglamorous engineering discipline:

Cryptographic Isolation per Tenant, Not Just Per-Tenant Configuration	Every tenant's data is protected by its own dedicated, auto-rotated encryption key hierarchy, and access to the underlying systems runs through IAM roles rather than shared credentials. Cross-tenant isolation isn't just an annual pentest checkbox — automated tests continuously verify that a credential or session for one tenant can never reach another's.
Perimeter Defense Tuned for This Specific Traffic	Every request passes through a web application firewall and load balancing layer with rate limiting applied both at individual enforcement points and in aggregate, tuned and continuously monitored against the actual traffic patterns Aembit sees, not a generic off-the-shelf ruleset left on defaults.
Least-Privilege Access to Aembit's Own Environment, Not Just Customers'	Production access is restricted to a small, named set of engineers, and direct database access is prohibited outside of narrowly scoped paths the software itself uses. Aembit holds itself to the same least-privilege standard it enforces for every agent and workload it protects.

Continuous Vulnerability and Software Supply-Chain Scanning	Every component, cloud and edge alike, is scanned as part of the build itself, with a software bill of materials generated and bundled into every container image so a vulnerability in an underlying package gets caught and flagged automatically, not discovered after the fact. Dependencies are updated on a regular cadence rather than only when something breaks, and the administrative console itself is built and shipped as static content, which by design closes off an entire category of runtime web vulnerabilities before they're ever a possibility.
Independent Verification, Continuously, Not Just Annually	Third-party penetration testing runs quarterly against a production-like environment, layered on top of automated tests that run continuously in the background specifically trying to cross tenant boundaries, confirming a credential or session for one tenant can never reach another's data.

The clearest proof, though, is a live deployment: a $300 billion investment firm running Aembit to secure Claude access across its enterprise, inside one of the most heavily regulated corners of the financial industry. We cover that deployment in full later in this paper — the pattern it followed, and the results it produced, hold up under exactly the kind of scrutiny a regulated enterprise brings to a new security control.

8. The Aembit *Difference*

Nobody adopting Aembit is starting from a blank slate. Every enterprise already has secrets managers, an identity provider, maybe a PAM tool, maybe an IGA platform — and the honest answer to "how does Aembit fit with what we already have" isn't the same for all of them. Some of that stack Aembit replaces outright. Some of it Aembit integrates with directly. Some of it Aembit simply runs alongside, because the two tools are solving different problems that both need solving.

Category	What It Was Built For	Where Aembit Fits
Secrets Managers / Vaults HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, CyberArk Conjur	Store, rotate, and audit static credentials for applications.	**Replaces or Integrates** Aembit removes the need for a workload or agent to ever fetch a stored secret, but plugs directly into a vault you already run wherever you'd rather keep one as the credential source.
Workload Identity Federation / SPIFFE-SPIRE	Protocol-level exchange of a trusted identity token for short-lived credentials, built into major clouds as a primitive.	**Integrates and Works Alongside** Aembit consumes these tokens as a trust provider (section 6), rather than competing with the primitive itself.
User IAM / Identity Providers Okta, Entra ID, Ping, Google Cloud Identity	Authenticate humans, run SSO and MFA, grant federated access to applications.	**Integrates and Works Alongside** Aembit federates with the IdP you already run for the human half of a blended identity decision, instead of duplicating it.

PAM CyberArk, BeyondTrust, Delinea	Control, monitor, and audit privileged access for human administrators and operators.	**Works Alongside** PAM governs the humans; Aembit governs the agents and workloads acting on their behalf.
IGA SailPoint, Saviynt, Omada	Inventory, certify, and govern the access lifecycle for human users.	**Works Alongside** A different plane entirely, governance of human entitlements versus runtime enforcement for non-human ones.
NHI-Focused Governance Oasis Security, Astrix, Entro Security, Veza	Discover, risk-score, and track the lifecycle of non-human identities and machine credentials across the environment.	**Replaces, Integrates, or Works Alongside** Depending on scope. Aembit is often both a source of the identities being governed and, for narrower point tools, a replacement.
API Security / API Gateways Kong, Apigee, cloud-native gateways	Route, rate-limit, and enforce authentication policy on API traffic at the network edge.	**Integrates and Works Alongside** Aembit provides attested identities and manages token exchange, so the gateway can focus on fronting APIs.

The pattern across all seven categories: Aembit isn't trying to be every one of these tools at once, and it doesn't ask you to rip out the ones already doing their job well. It sits in the one place none of them cover — the runtime decision of whether this specific agent or workload, acting for this specific human or acting on its own, gets this specific credential, right now — and connects outward to whatever you already have everywhere else. The full vendor-by-vendor breakdown, updated as the stack evolves, is maintained at **Aembit vs. Your Stack**.

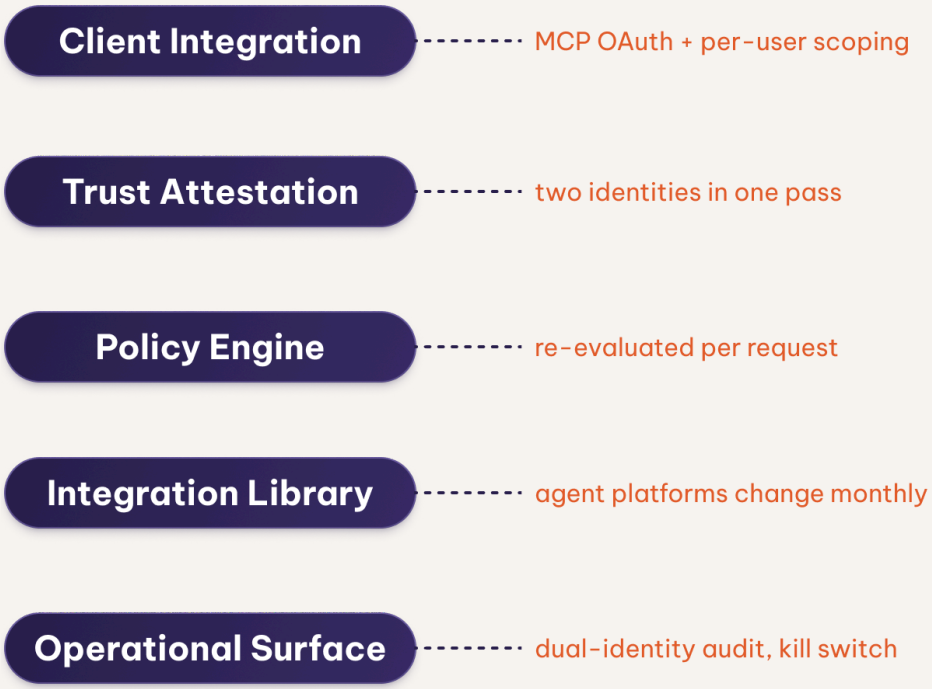
9. Build vs. Buy

Several of the organizations now running Aembit in production didn't start there. They started by building the access infrastructure their AI agents needed themselves, treating it as a smaller problem than it turned out to be, and only came to Aembit after concluding that a strategic partner would get them there faster, and at a lower total cost, than continuing to maintain it in-house. That pattern is common enough to be worth walking through, because the part that actually breaks isn't the part most teams scope for.

AI agents break a homegrown system in a way a traditional workload never did: the same request has to resolve two identities in one decision. A pilot that hands Claude Desktop a static credential for one team works fine on day one, because there's only one agent and one human to reason about. Day two is where it breaks — a second team wants the same agent, a platform agent shows up inside Copilot Studio needing the identical blended identity check, a custom agent built on your own stack goes live in Kubernetes, and an Agentforce agent gets embedded in a SaaS product where you can't touch the runtime at all.

Every one of those is a different cell in the framework from section 3, and every cell still needs the same underlying decision: this agent, acting for this specific person, requesting this specific action. Most homegrown systems weren't built to ask that question even once, let alone consistently across four different runtimes.

That gap breaks down into five engineering layers, and for AI agents specifically, every layer is more difficult than its workload-only equivalent:



Layer	Why It's Harder for AI Agents
Client Integration	Has to speak MCP's OAuth 2.1 authorization code flow and inject a credential scoped to which human the agent is currently acting for, not just relay a static key – and it has to work across proxy, SDK, and CLI integration surfaces to reach hosted agent runtimes and SaaS platforms that won't accept a sidecar at all.
Trust Attestation	Has to verify two identities in one pass – the agent's own workload identity and the human it's acting for – not the single-identity attestation a traditional workload needed.
Policy Engine	Has to evaluate per request, because an agent's next action isn't fixed the way a traditional service's call pattern is: the same agent can get a different outcome from the same policy depending on which person is driving it right now.
Integration Library	Has to keep pace with MCP servers and agent platforms that change monthly, on top of the stable cloud and SaaS APIs a traditional integration library already had to cover.
Operational Surface	Has to log every event with both the agent's and the human's identity attached, and support agent-specific controls like a kill switch, tool-level filtering, and human-in-the-loop approval – not just uptime and cert rotation.

Two problems tend to surface only after a team is already committed. Delegation chains: an agent that hands part of its task to another agent needs that chain of custody carried through the policy decision, not just the first hop — something almost no homegrown design anticipates, because it wasn't a workload problem before agents existed. Audit for blended access: a SOC 2 or HIPAA reviewer now asks whether you can tell a human's action apart from an agent's in the same log, and a system built around a single identity per event structurally can't answer that after the fact. The traditional problems, a static vault credential an agent can leak or be prompt-injected out of, or the secret-zero question of what authenticates the thing fetching its own credential, still apply too, but they're the workload-era problems layered underneath, not the new ones agents actually introduce.

6-24 mo.

for a production-grade SPIRE rollout — before any of the agent-specific gaps are addressed

Open-source maintainer experience

SPIFFE/SPIRE is a legitimate answer to workload attestation, and section 5 already covers why standards give you a foundation rather than the full stack. For agents specifically, the gap is bigger than it is for a traditional service: SPIRE's entire model assumes a single, stable workload identity, so it has nothing to say about the human half of a blended identity decision, and nothing to say about MCP's OAuth 2.1 flow, the protocol actually governing most agent-to-tool access today. It's also operationally heavy even for what it does cover — eight to ten interdependent components, including a SPIRE agent process on every node, a highly available server, an Envoy proxy with its own Secret Discovery Service, a policy engine SPIRE doesn't include, and a management UI it doesn't ship. That's consistent with what open-source maintainers in this space describe from experience: a production-grade rollout realistically takes six to twenty-four months depending on environment complexity, all before a single one of the AI-specific gaps above is addressed.

None of this means building is always the wrong call. It means the decision has to be evaluated against what agents actually require: blended identity, delegation chains, per-request policy, agent-aware audit, and against the scope of the workload problem a team may have originally solved for its traditional services. The organizations that migrate off a homegrown system tend to do it before that gap catches up with them, not after — **a full breakdown of what actually breaks, layer by layer, is here**.

10. Case Study: A $300 Billion Investment Firm *Secures Claude Access*

BEFORE
• THREE STOP-SHIP GAPS •

- ✗ No agent identity — Claude acted under each analyst's full identity and access rights
- ✗ Secrets, tokens, and API keys stored in Claude and MCP server configs
- ✗ No central access control; no way to tell human action from agent action

AFTER
• WHAT AEMBIT PUT IN PLACE •

- ✓ Blended Identity binds each user to the agent in one policy decision
- ✓ Runtime least-privilege policy per agent, per MCP server, on existing Okta
- ✓ Just-in-time, secretless tokens replace every stored credential
- ✓ Every agent action logged, attributed, forwarded to CrowdStrike SIEM

$300B	2 weeks	<6 hours
investment firm, heavily regulated.	from start to production deployment.	of security team time required.

This firm's security team faced the exact mismatch section 1 opened with, at enterprise scale. Claude was going out as a personalized AI assistant across the entire workforce, reaching financial datasets for research, email and calendars for productivity, and sensitive company data through SharePoint and Microsoft 365, with financial analysts and executives as the first critical group — inside one of the most heavily regulated corners of the economy.

Three gaps made that a stop-ship problem for security. First, no agent identity, only user identity: Claude reached MCP servers under the analyst's full identity and access rights, with no distinct agent identity, no audit attribution, and no coverage for anything outside Okta's own governance. Second, long-lived credentials in the wrong places: MCP authentication meant storing secrets, tokens, and API keys directly in Claude or in MCP server configurations that were never designed to be credential stores. Third, no centralized access control or auditability: policy enforcement was piecemeal, tied to human identity alone, with no way to tell whether a human or the agent acting for them had taken a given action.

What Aembit put in place maps directly onto this paper's framework. Blended identity, as described in section 4, binds the specific user to the specific agent in a single policy decision, rather than the agent inheriting the human's access wholesale. Runtime policy enforcement grants granular, least-privilege access per agent, per MCP server, built on the firm's existing Okta identity foundation, with no second identity provider to stand up. Just-in-time, secretless access replaces every long-lived stored credential with a short-lived, policy-scoped token issued per session. And full auditability means every agent action is logged and attributed, forwarded into the firm's CrowdStrike SIEM, giving security the same visibility into agent behavior it already had into human activity.

The architecture underneath is exactly what section 6 describes. Aembit's cloud control plane handles policy administration and logging; the MCP Authorization Service issues blended identity tokens through a standards-compliant OAuth 2.1 with PKCE flow; the MCP Identity Gateway enforces policy and exchanges tokens on every tool invocation, logging each one. Human identity stays anchored to the firm's existing Okta instance, so there's nothing new for employees to learn and nothing new for the identity team to manage. Deployment took two weeks, with under six hours of the security team's time required to stand it up — the same numbers referenced in this paper's introduction.

Three expansion tracks are already underway: widening the footprint of agents and MCP servers the existing deployment covers, securing agent-to-agent communication as multi-agent workflows start to emerge, and extending Aembit's workload IAM beyond AI agents entirely, into CI/CD pipelines, DevOps credential management, and broader secretless access across the firm. The same architecture that secured one agent for one team is, by design, the architecture that scales to the rest of the organization.

The full case study is available at **A $300B Investment Firm Secures Claude Access with Aembit**.

11. Summary

The 97% of security leaders who expect a material AI agent-driven incident in the next year, against the 6% of security budget allocated to prevent one, is the gap this paper set out to close. It isn't closed by picking a side between workforce IAM and workload IAM, or by treating agents as a special case bolted onto whichever one you already have. It's closed by organizing around the two vectors that actually determine what an agent needs, who it acts for and where it runs, and building one architecture that evaluates blended identity for the agents acting on someone's behalf, workload attestation alone for the ones acting on their own, and applies the same trust providers, policy, and credential providers to both, across every cloud, SaaS product, and on-premises system an enterprise actually runs.

97% expect an incident. **6%** of budget to prevent one.

Aembit has run its access management solution in production for years, at Fortune 500 companies, in some of the most regulated industries in the world, now extended to a new protocol and a new kind of identity. It's also built on the same philosophies that have driven IAM for humans for two decades, identity first, least privilege, policy over static trust, applied to non-human identities for the first time with the same rigor. A $300 billion investment firm proved it at enterprise scale in two weeks, with under six hours of security team time. The standards work underway across the industry is real, but standards alone were never going to be the whole answer. Building the remaining capabilities yourself means taking on five engineering layers, although many initial estimates account for only one of them.

None of this requires a large upfront commitment to find out if it's the right fit. Aembit's Starter tier is free and covers everything from discovery to your first blended identity policy. **Sign up for the free Starter tier**, **talk to an engineer** about a specific environment, or go straight to **the AI Guide in Aembit's docs** to see what standing up your first agent policy actually looks like.

START FREE

The Starter tier covers discovery through your first Blended Identity policy.

Sign up for the free starter tier

TALK IT THROUGH

Bring a specific environment and walk it with someone who builds this.

Talk to an engineer

SEE IT IN THE DOCS

What standing up your first agent policy actually looks like, step by step.

Open the AI Guide

aembit

Gartner COOL VENDOR 2022

CSO Cybersecurity startups to watch for in 2023

2024 WINNER CYBER SECURITY EXCELLENCE AWARDS

RSAC Innovation Sandbox 2024 FINALIST

2024 SINET16 INNOVATOR AWARD