GOVERNANCE & ORCHESTRATION

Agentic Safety: Inside the Keep

Every approach to agent safety tries to make the model behave. Keep assumes the agent will go rogue, and makes it not matter.

July 2026 · v1.0 · 18 min read

01 The thesis: assume the agent goes rogue, make it not matter
02 The invariant principle: the boundary lives at the gateway
03 The outer ring: Moat and Barbican, free and open
04 The enforcement heart: Rampart feeds the Warden
05 The credential oracle: a handler that never holds a key
06 The dual-gate authority model
07 Why "un-bypassable" is true
08 The signed, tamper-evident record
09 The claim-tier ladder: engineered, advisory, measured

Executive summary

Every other approach to AI-agent safety tries to make the model behave — prompt guardrails, alignment filters, steering vectors. All of them are advisory: they reduce the probability of a bad action, but a sufficiently clever jailbreak, a hallucination loop, or a novel injection eventually slips through, because the model is the thing being asked to police itself.

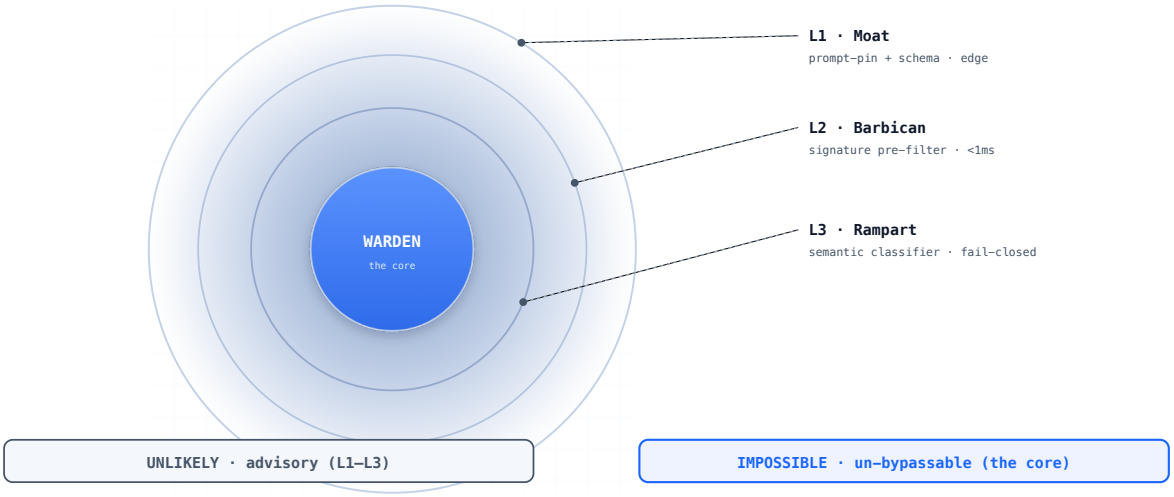
Keep takes the opposite stance: we assume the agent will go rogue, and make it not matter. The agent executes inside an envelope the infrastructure has already bounded — it can think anything, but it can only do what the gateway permits, and every attempt is attributed and recorded. Safety stops being a probability and becomes a property of the system.

This paper defends one sentence: the agent operates inside the envelope the gateway already bounded — it cannot widen it. It walks the enforcement architecture layer by layer, shows the exact point where "unlikely" becomes "impossible," and draws the precise line between what is correct by construction, what is probabilistic defense-in-depth, and what is a measured number rather than an adjective.

01 The thesis: assume the agent goes rogue, make it not matter

Keep is a composable product, and its product boundary is its claim boundary — held precisely everywhere, because conflating the two is how a security paper loses a technical reader. Standalone, Keep is the Moat and the Barbican (free, open-source, at the edge) plus Rampart (a hosted, billable classifier). It works with any backend and any model, no migration required. This is layered defense-in-depth: it raises the cost of an attack and catches the known and the semantic-obvious. It is not called un-bypassable, because a developer who embeds it can also choose not to route through it.

Keep plus Enforced Boundary is a different claim. The agent runs inside the platform sandbox, where the Warden is the sole path to any mutation. This is the un-bypassable guarantee — not a policy, a structural fact, because there is no other egress. Said plainly: Moat, Barbican, and Rampart make a rogue action unlikely. The Warden, inside Enforced Boundary, makes it impossible. The first three are the product a developer adopts in five minutes; the Warden is the product a regulated enterprise deploys when "unlikely" is not good enough.

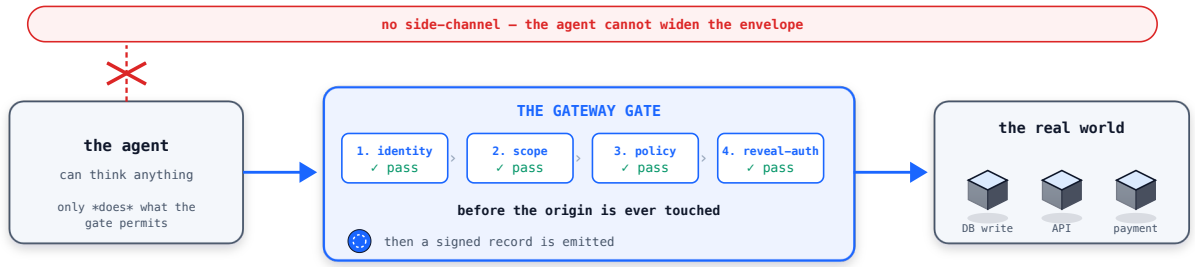


Keep, standalone — Moat and Barbican at the edge, Rampart as a hosted classifier — is layered, advisory defense-in-depth: it makes a rogue action unlikely. The Warden, running inside Enforced Boundary, is the sole path to any mutation: it makes a rogue action impossible, by construction, not by policy.

02 The invariant principle: the boundary lives at the gateway

The trust boundary lives at the gateway, not in the agent. An autonomous agent — however it was prompted, whatever it hallucinated — reaches the real world only through a single enforced path. That path checks identity, scope, policy, and reveal-authority before the origin is ever touched, and emits a signed record every time. The agent cannot widen the envelope; it can only operate inside it.

This is the one sentence the whole paper defends: the agent operates inside the envelope the gateway already bounded — it cannot widen it. Everything that follows is the mechanics of how that envelope is built and enforced, layer by layer.

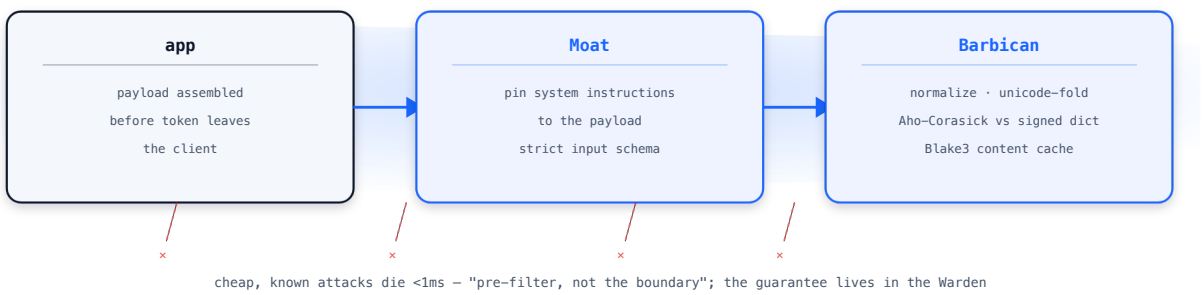


The agent can think anything. The only arrow from it to the real world — a database write, an API mutation, a payment — passes through a gate that checks identity, then scope, then policy, then reveal-authority, before the origin is ever touched, and then emits a signed record. The agent cannot widen the envelope on its own.

03 The outer ring: Moat and Barbican, free and open

The first defense runs before a token leaves the application. The Moat SDK pins the developer's system instructions to the payload — so a downstream "ignore all previous instructions" cannot override them — and enforces a strict input schema at the client border, rejecting malformed or shape-shifted inputs immediately. It needs no account and no network round-trip; structurally flawed requests die at the edge, for free.

Behind it, the Barbican is an ultra-low-latency WebAssembly filter that normalizes the payload — unicode-folding, de-obfuscation — then runs a multi-pattern scan against a signed, hot-reloadable dictionary of known-malicious injection vectors, with a content-hash cache to short-circuit repeats. Precisely framed: this is a pre-filter, not the security boundary — static signatures are evaded by sufficiently novel obfuscation, which is precisely why the dictionary is signed-and-updatable and why the real guarantee lives further in, at the Warden. The Barbican's job is to drop the cheap, obvious attacks in under a millisecond so the expensive layers see less load.



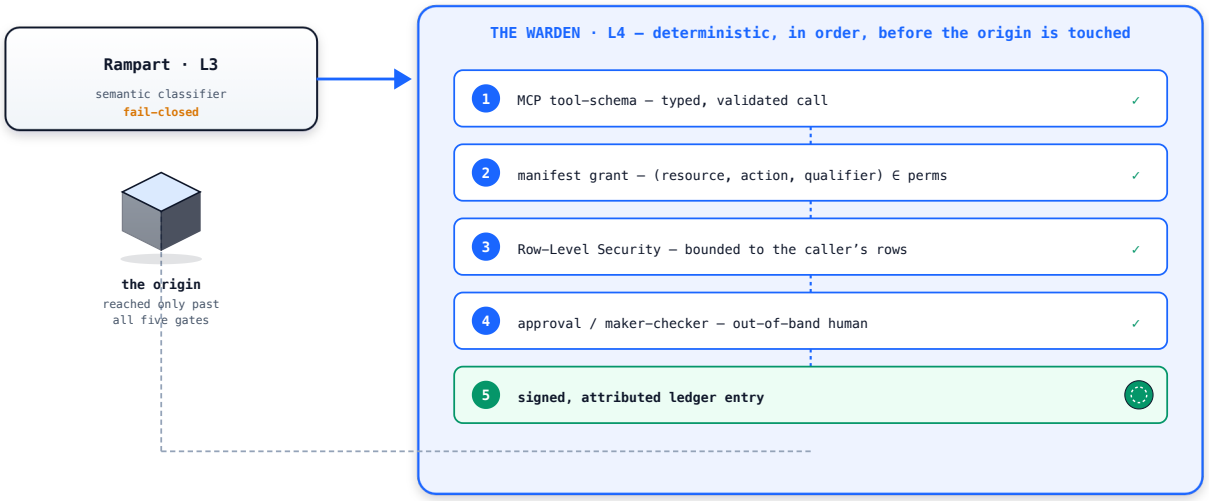
App → Moat (pins system instructions to the payload and enforces a strict input schema, before the token leaves the client) → Barbican (normalizes the payload, then runs an Aho-Corasick scan against a signed, hot-reloadable dictionary, cached by content hash for repeats). Cheap, known attacks die in under a millisecond — a pre-filter, not the boundary.

04 The enforcement heart: Rampart feeds the Warden

Behind the wall sits semantic risk. Rampart is a non-generative sequence classifier that scores the payload for prompt-injection and jailbreak intent — the social-engineering attacks a string-match can't see — and returns a verdict the gateway acts on. Classification is probabilistic, so it is defense-in-depth, not a gate, and it is fail-closed: if the classifier is unavailable or uncertain, the request is blocked by default, never waved through.

The Warden is the keep itself, and it is the product. It is not a filter that scores and hopes — it is deterministic enforcement on every agent action, built correct-by-construction into the gateway kernel.

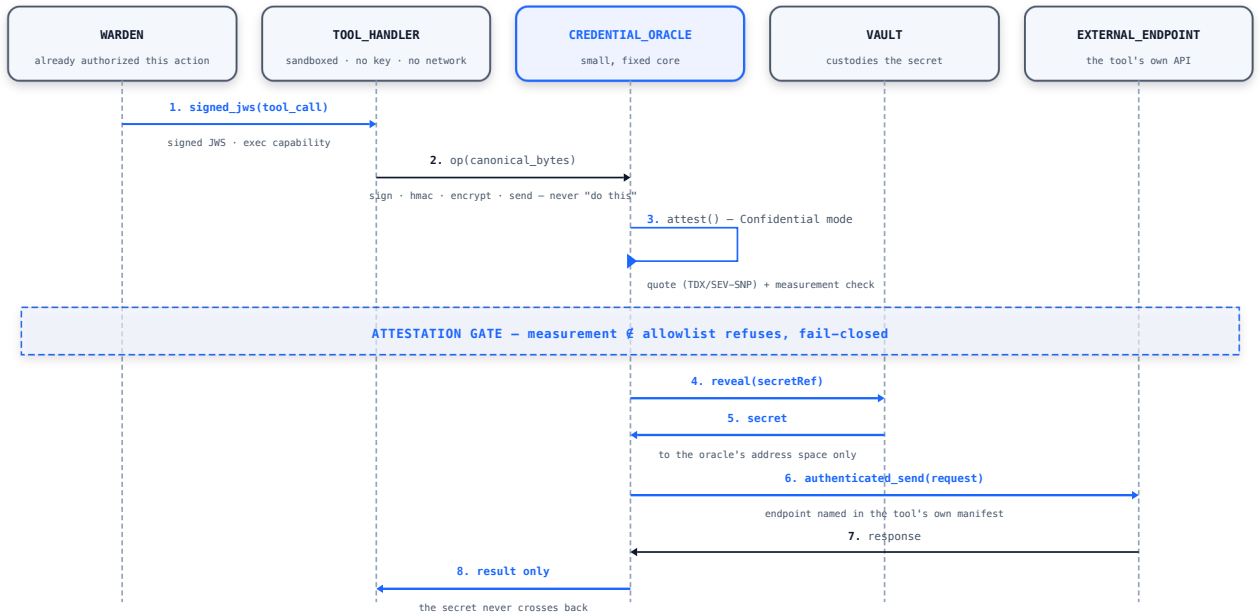
It canonicalizes agent communication through an MCP tool-schema, so every tool call is a typed, validated, declared transaction, never a raw command, then enforces, in order, before the origin is touched: the manifest grant (the action's resource, action, and qualifier must be in the caller's bound permissions, or it is refused), row-level security (bounded to the rows the caller's identity may touch), the approval gate for high-stakes action classes (an explicit, out-of-band human attestation the agent cannot self-issue), and finally the signed, identity-attributed ledger entry.



Behind the wall, Rampart scores semantic risk and is fail-closed: unavailable or uncertain, the request is blocked by default, never waved through. It feeds the Warden — an ordered, deterministic gate stack, all before the origin is touched: MCP tool-schema, the manifest grant, row-level security, the approval/maker-checker gate, then a signed attributed ledger entry.

05 The credential oracle: a handler that never holds a key

Clearing the Warden's gate stack answers whether an action is authorized. It doesn't answer how the tool call itself reaches the outside world safely — third-party, per-tool handler code, the least-trusted part of the system. Keep isolates that handler in a credentialless, sandboxed slot with no key and no network of its own. When the tool's own authentication needs a keyed operation — an HMAC, a SigV4 request, an mTLS handshake — the slot cannot perform it: it hands the fully-formed bytes back to a small, fixed credential oracle running in its own address space, over a narrow, fixed set of operations. The oracle, not the handler, selects the credential and endpoint from the tool's own verified manifest. The secret never leaves the oracle's boundary; only the result comes back.



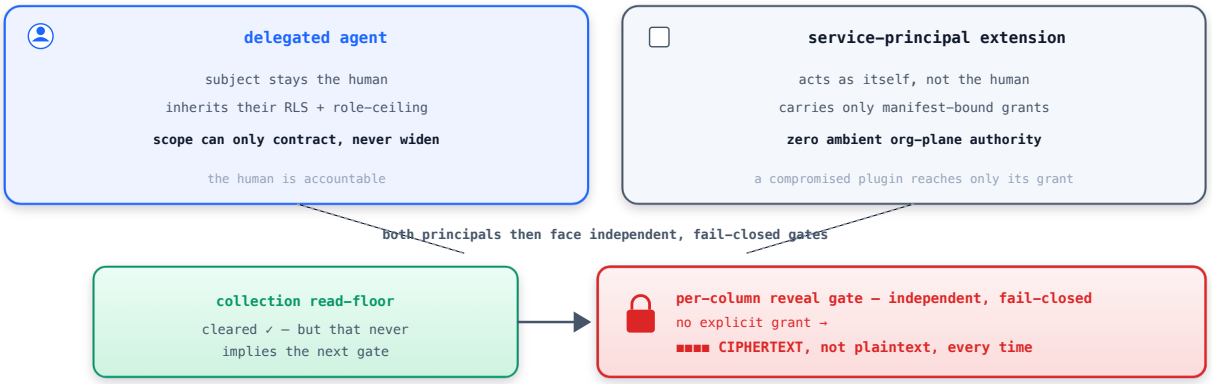
The Warden, having already authorized the action, hands the handler a signed JWS carrying an exec capability (1). The credential-less tool handler then sends canonical bytes over a fixed verb set (2). In Confidential mode, the oracle attests itself (3) and release is gated on the quote's measurement matching a project-scoped allowlist — any mismatch refuses, fail-closed. Only past that gate does the oracle reveal the secret from the vault that actually custodies it (4, 5), make the authenticated call to the endpoint named in the tool's own manifest itself (6), take the response (7), and return the result alone — the secret itself never crosses back (8).

In Confidential mode, the oracle itself has to earn the right to hold that secret. Before any secret is released to it, the oracle generates a real hardware attestation quote — a TDX or SEV-SNP quote, signed by the hardware itself, not a cloud provider's attestation service — and release is gated on that quote's measurements matching an allowlist scoped to the project. A missing quote, a platform mismatch, a measurement below the required tier, or a measurement simply absent from the allowlist all refuse the release, through the same fail-closed path. Off a genuine confidential VM, the oracle cannot produce a valid quote at all, so the credential channel to it never opens.

06 The dual-gate authority model

When a workspace fires an agent action for a user, the platform mints a short-lived delegated token whose subject stays the user — the agent acts as the human, inheriting that human's identity, row-level security, and role-ceiling, never an ambient super-user. On top of that ceiling, the agent's declared scope can only contract the grant, never widen it: a targeted resource outside its allow-list is refused. An agent is, by construction, at most as powerful as the human who invoked it, usually less — and an extension minting its own delegation, a privilege-escalation attempt, is structurally refused, because delegation is platform-issued only.

A second kind of principal exists alongside the delegated agent: the service-principal, a third-party extension acting as itself. It carries only its manifest-bound grants and zero ambient org-plane authority. For a buyer worried about a supply-chain attack through a malicious plugin, this is the line: a compromised extension cannot reach beyond exactly what the customer granted it.



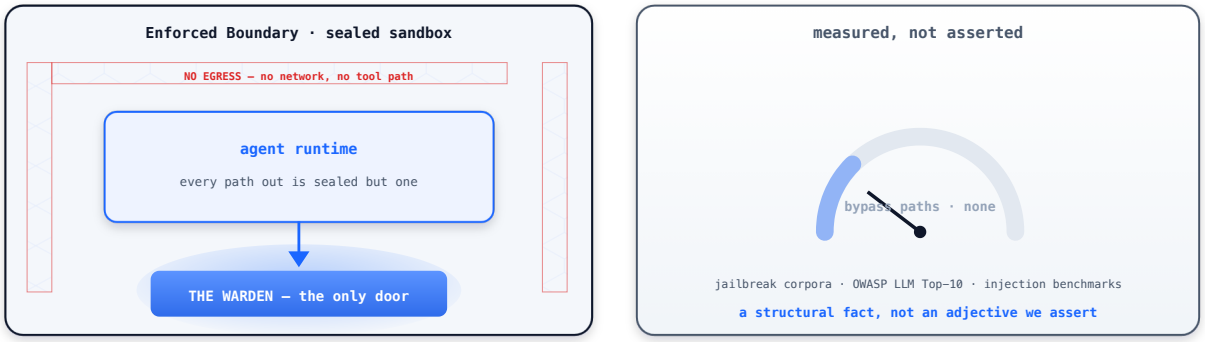
Two kinds of principal, cleanly separated. The delegated agent's subject stays the human — it inherits their row-level security and role-ceiling, and its scope can only contract, never widen. The service-principal extension carries only its manifest-bound grants and zero ambient org-plane authority. Both then face independent, fail-closed gates: clearing the collection read-floor never implies the per-column reveal gate, which returns ciphertext by default.

The two gates are independent and fail-closed. Even an extension that clears the broad collection-level read floor still hits a separate, explicit, per-column reveal rule: if the customer has not manually granted reveal on that column to that principal, the extension receives ciphertext, not plaintext — by default, every time. Clearing one gate never implies the next.

07 Why "un-bypassable" is true

Inside Enforced Boundary, the Warden is the sole egress from the agent’s sandbox — the runtime has no other network or tool path to the outside, so "route through the Warden" is not a rule an agent could break but the only door in the room.

We do not ask a reader to take that on faith, and we do not assert an adjective. The claim is structural: enforcement sits on the sole egress path, so it does not depend on a model behaving well, and it is not described by a bypass rate.

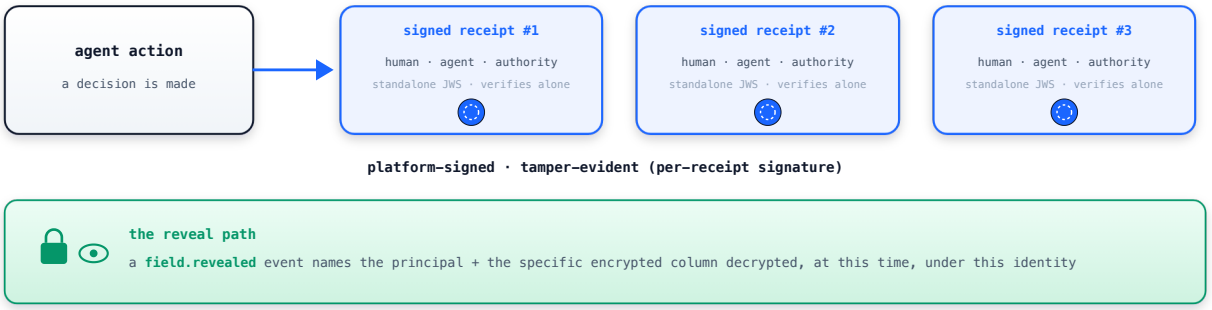


Inside Enforced Boundary, the Warden is the sole egress from the agent’s sandbox — the runtime has no other network or tool path to the outside, so routing through the Warden is not a rule an agent could break but the only door in the room. Un-bypassable is a structural fact, not an adjective asserted.

08 The signed, tamper-evident record

Every agent action lands on a per-project audit ledger bound to an identity: the human, plus the agent attribution, or the extension principal acting as itself. "Who did this — which human, via which agent, under what authority" resolves to a single record, not a reconstruction. This is the cleanest agentic-liability data trail available, and the foundation of an insurance-grade SLA story.

When an agent reads protected data, the Warden emits a discrete `field.revealed` event — a signed, tamper-evident record naming the principal and the specific encrypted column decrypted. Audit moves from the vague ("the agent queried a database") to the forensic ("the agent decrypted exactly this protected column, at this time, under this identity"). Every look inside an encrypted wrapper leaves a mark.



Every agent action resolves to a single signed, identity-attributed ledger entry — which human, via which agent, under what authority. Each receipt is a standalone signature by the platform's ledger key and tamper-evident: any change to a receipt breaks its signature, and an attacker without the key cannot forge one. On the reveal path, a `field.revealed` event names the principal and the specific encrypted column decrypted, at this time, under this identity.

Each ledger entry is individually signed and tamper-evident, not tamper-proof — a forged or altered entry does not verify against the platform's published key. (Kilimanjaro's separate per-project custody ledger, covering key-lifecycle events like rotation and erasure, additionally uses a gap-free hash-chain; the Warden's action-and-reveal audit is receipt-only, matched to its far higher event volume.)

09 The claim-tier ladder: engineered, advisory, measured

The Warden enforcement line — delegated-token identity binding, the role-ceiling, scope-narrowing, the manifest-grant gate, row-level security, the approval and maker-checker gate, the fail-closed per-column reveal, and the signed attributed ledger — is part of the platform gateway. Each element can be demonstrated in the SDK and the ledger.

Moat, Barbican, and Rampart are defense-in-depth and probabilistic: they reduce risk, and they are not the guarantee. Any bypass rate sits on the bottom rung deliberately — measured, not asserted, never a marketing absolute.



Correct by construction: the Warden enforcement line, demonstrable in the SDK and the ledger. Defense-in-depth, probabilistic: Moat, Barbican, and Rampart reduce risk but are not the guarantee. Measured, not asserted: a bypass rate is a red-team measurement, never a marketing absolute.

What we disclose

Correct by construction: the Warden enforcement line — delegated-token identity binding, the role-ceiling, scope-narrowing, the manifest-grant gate, row-level security, the approval/maker-checker gate, the fail-closed per-column reveal, and the signed attributed ledger. Each element can be demonstrated in the SDK and the ledger.

Defense-in-depth, probabilistic: Moat (pre-token hygiene), Barbican (signature pre-filter), and Rampart (semantic classifier) reduce risk; they are not the guarantee. Un-bypassable is a claim reserved for the Warden running inside Enforced Boundary, where it is the sole sandbox egress — never for standalone Keep. Enforced Boundary is an eligibility state, not a blanket mode: it is granted only to a deployment confined entirely to the sealed, sole-egress runtime. A deployment that also runs functions in a separately-sandboxed runtime is not eligible — the sole-egress guarantee belongs to the sealed runtime, and Enforced Boundary is the state of being wholly confined to it.

Measured, not asserted: the guarantee is structural — enforced by sole-egress architecture — and any bypass rate is a red-team measurement, never a marketing absolute.

The `governedOp` primitive is deterministic progress telemetry — an observability-and-billing event, never a security or gating term. Enforcement is the Warden's word; governedOp is the meter's word; this paper never crosses them.

The un-bypassable guarantee assumes the enforcement point itself is not compromised — the Warden is the thing enforcing the boundary, not something outside it. Hardening that enforcement point is a deployment property, addressed by the platform's own operational security. The credential oracle's isolated, attested environment separately confines a compromised tool handler to that handler's own credential and endpoint — but that containment is scoped to a malicious or subverted tool, not to a compromise of the enforcement point itself.

See where the boundary actually sits in your stack

Talk to a Sovereign Engineer about which of your agent workflows run as standalone-Keep advisory, and what moves into Enforced Boundary's sole-egress guarantee.

Talk to a Sovereign Engineer →

See how Keep works