

Host Level Authorization at the Act

The terminal layer of authorization for autonomous software.

Authorization for autonomous software is being rebuilt around the credential. A credential must predict, at issuance, the set of actions that will be legitimate, and autonomy is precisely the property that defeats prediction. Above the kernel an act is legible in terms a policy about acts can speak, and every observation of it is a claim. Below the kernel observation becomes unforgeable and the act stops being legible as an act. One band holds both properties at once, and that makes it the terminal boundary of authorization rather than a preferred one. Everything above it is a prediction about a moment only it can see.

The premise here is shared, not contrarian. Agents hold real credentials, act at machine speed, and the authorization model built for hands on keyboards does not survive them. Establishing who an agent is, issuing it short-lived least-privilege credentials, and binding it to an accountable human are all necessary. The question this paper asks is narrower: whether that layer can also carry enforcement.

An agent holding a valid identity and a valid credential is steered by a prompt injection into reading a second repository its granted authorization does not name. The credential has not changed, so every control above the act permits what comes next. The kernel sees the act reach outside the granted region, and the read is denied before it commits.

SentoClaw builds that layer. Per-act authorization, enforced where the act commits, deployed as software beside the identity and policy systems a team already runs. This paper is the argument behind it.

PROPOSITION 1

The credential is a prediction, and it was a good one while software was deterministic.

A credential encodes, at the moment of issuance, the set of actions its holder may legitimately take. For deterministic software this is nearly free. The program's action set is fixed at compile time, so the prediction is trivially correct and the credential is a faithful summary of what will happen.

Every control above the act inherits that property. Policy engines reason about the credential. Gateways route on it. Continuous evaluation re-checks the conditions under which it was granted. The entire stack is an elaboration of one bet: that what a principal is reliably determines what it will *do*.

Nothing in that architecture is wrong. It is load-bearing on an assumption that held for forty years and stopped holding when the workload became non-deterministic.

PROPOSITION 2

Autonomy defeats prediction, and the degradation is specific.

The clearest marker is not a security finding. It is how long-horizon agents are being built. Emergence AI's autonomy laboratory runs agents continuously for fifteen to sixteen days against more than 120 tools, with the design point stated explicitly: the environment forces dynamic discovery and chaining rather than pre-specification. An agent that

discovers and chains its own tool sequences over two weeks has an action set nobody enumerated. Rule violations inside the simulation ranged from zero to 683 across models. None of it required an attacker.

The incident side gives the same answer. A single autonomous agent's intrusion was reconstructed at roughly 17,600 actions in about 6,280 clusters over two and a half days. Human review sat underneath every reactive control in that stack, and at that tempo it is already gone.

Under those conditions the credential model degrades in three ways that compound rather than resolve.

Scope goes vacuous before time does. The response to agent risk has been to shorten credential lifetime, but the failure lives on the scope axis. A short-lived credential for a broad scope is still a broad grant, and as an agent's action space widens, the scope required for it to work converges on whatever it might need. Shortening the time axis does not touch this.

Continuous evaluation becomes sampling. Re-checking on signals is bounded by signal rate. Action rate is not. As the ratio of acts taken to evaluations performed grows, continuous becomes a sampling interval with a better name.

Delegation has no standard. The April 2026 survey of AI identity standards is unambiguous: no production-ready standard traces the authorization chain back to the originating human across multiple hops, and no deployed protocol enforces that each delegation step narrows rather than widens permitted actions.

That survey also names the category error directly. It identifies the assumption that cryptographic correctness implies semantic correctness as false, notes that trusted execution environments authenticate containers rather than content, and concludes that these are "boundary conditions, not engineering backlogs."

PROPOSITION 3

Authorization requires two properties, and they move in opposite directions through the stack.

The act must be legible: the system must be able to tell what is about to happen in terms a policy about acts can speak. And the observation must be unforgeable by the actor: the party being authorized must not be the source of the evidence used to authorize it.

High in the stack an act is fully legible and every observation of it is a claim made by the thing being judged. Low in the stack observation becomes incorruptible and the act dissolves. A hypervisor sees vCPU exits. Firmware and silicon attest which code is running. Neither can say what that code is doing in terms a decision can use.

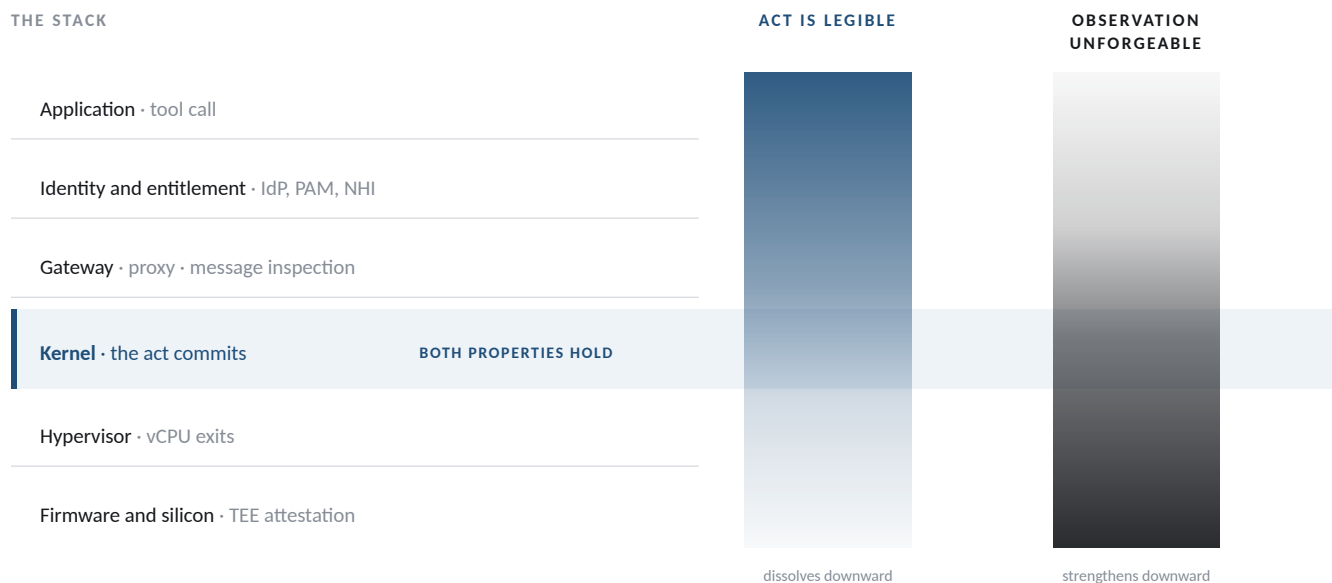


Figure 1. The two properties authorization requires move in opposite directions through the stack. Legibility is strongest at the top and dissolves downward. Unforgeability is weakest at the top and strengthens downward. They overlap in one band. That overlap is where the boundary falls.

The kernel's view of the act is the band where both properties hold. It is the last place an act is still legible in terms a policy about acts can speak, and the first place the actor cannot author the evidence. The kernel offers more than one observation point inside that band, with different legibility and identical unforgeability. The band is what the argument needs. The syscall is the representative case and the one easiest to reason about.

The layer trusts the kernel. A kernel compromise or a container escape forges the observation, which is why measured boot and hardware attestation belong underneath this layer rather than in competition with it. They answer which code is running. This layer answers what that code is doing.

This is also why the layer is not a location in one company's stack. If agent activity migrates into a managed runtime or a chain of third-party services, every chain of calls still terminates in something durable: a write, a transfer, a dispatch. That commit happens on someone's kernel. The layer follows the act.

PROPOSITION 4

What makes it a control point is not depth. It is independence.

Being beneath the identity layer would only make this a component. What makes it load-bearing is narrower, and worth stating exactly: it never consumes a claim presented by the actor at runtime.

Every control above the act inherits the errors of the controls above it. If an agent registry is poisoned, if a credential is mis-issued, if a delegation chain attributes to the wrong origin, every downstream system enforces the wrong thing faithfully and logs it perfectly. That is not a failure of those systems. It is what correct systems do with incorrect inputs.

An observation-bound layer does not inherit that error, because it never reads the identity claim. It takes its own measurement of which process acted on which object, at the moment the act commits.

It does inherit authoring errors in the grant, the same as everyone. A grant that says the wrong thing is enforced exactly as written. The claim here is about runtime, and about what the actor can put in front of the decision.

What changes is blast radius. An error made above the act is bounded to the scope of one act rather than to a credential's lifetime multiplied by its scope.

That is also the answer to whether this is defense in depth. Defense in depth is layers that fail together when a shared assumption fails. This layer shares no runtime assumption with the ones above it.

PROPOSITION 5

The reverse path is where the credential model has no object to reason about.

A credential is issued to a principal before the causal chain exists. When a tool's response re-enters the agent and the agent then acts, the credential has not changed, so the system permits the new act on the strength of authority granted before the influencing content existed.

There is no provenance field in a credential, and there could not be one. The credential predates the thing it would need to describe. The reverse path belongs to an *interaction*, and the credential model has no object for an interaction. It has principals.

World models make this worse rather than better. A text-driven agent's poisoned tool output influences the next few steps, and that proximity is what makes detection viable. A world model consumes returned content as an update to learned state, and the plan derived from that state can produce actions arbitrarily far downstream with no textual path back. A classifier watching tool responses cannot flag content whose effect materializes four hundred actions later through a planner.

Detection exists on the reverse path. Revocation exists there, after the fact. Authorization does not, because the unit of authorization does not move when the reverse arrow fires.

Application-layer schemes that tag tool output with capabilities and propagate the taint forward are the honest counterexample, and they are reverse-path authorization. They fail the test in Proposition 3 rather than the test of existence: the tags are authored by the runtime being judged, so the evidence comes from the party the decision is about.

PROPOSITION 6

Adversarial pressure scales with model capability on one side of the boundary and not the other.

Any architecture that reads a presented artifact gets weaker as models get more capable, because a more capable model produces a more convincing presentation: better-formed attestations, more plausible justifications, better-chosen moments.

Capability still helps an adversary at an observation-bound layer. A more capable model is better at finding the acts that sit inside a grant and composing them toward an outcome nobody wanted. That is real, and it is bounded by how narrow the grant is.

What capability cannot do is forge the observation. The measurement is taken by a party the agent cannot influence. There is nothing to persuade.

So the adversary above the boundary gets better at the evidence itself. The adversary at the boundary gets better only at working inside the grant. That difference is what determines which architectures survive scaling.

PROPOSITION 7**Per-act authorization is a category that neither identity nor detection can express.**

Identity and policy cannot express it because their object is a principal and a permission. A grant good for exactly one act, bound to the process about to take it, is not a statement about a principal.

Detection cannot express it because its input is the act. A control whose evidence is the thing that happened cannot return a verdict that prevents it.

The nearest production answer deserves a straight one. eBPF adjudicates at the act and it can deny. Tetragon's Override action means the function “will never be executed and, instead, a value (typically an error) will be returned to the caller.” We run Tetragon ourselves. The same has been true of mandatory access control for two decades, where SELinux decides every access against policy and the access vector cache exists so those rules need not be consulted each time.

Neither has a grant. Both adjudicate a standing rule, loaded ahead of the act, against an observed call. The unit is a rule matched to a call, not an authorization issued for one act at one moment. That is not a deficiency in either. It is what in-kernel policy is.

Capability systems are the closer relative and deserve naming. Capsicum, seL4 and CHERI all carry unforgeable authority that is spent at the act, which is more than a standing rule does. The difference is custody. A capability is held by the actor, so possession is authority, and that is the property the next section rejects. A grant here is held outside the actor and bound by observation to one act.

| The kernel has always been able to decide at the act. What it has never had is a grant to decide against.

That places the layer. It sits after identity and policy have decided who and what may, and before detection can observe what did.

PROPOSITION 8**What per-act authorization closes.**

Non-human identity, NHI, takes as its object a principal and a permission. That is the right object for the question it answers, and it is a statement made before the act, about a holder. What it cannot express is a relationship between an authorization and one particular act, taken by one particular process, at one particular moment, because a principal is not an act and issuance is not a moment.

The observed binding supplies the missing term. Identity and policy decide who may and what may, and that decision is what authors the grant. The binding ties that grant to the act as the kernel measured it, at the instant the act commits. Neither piece works alone. Without the grant there is nothing to decide against, which is Proposition 7. Without the binding the grant is a permission again, and permissions are held rather than spent.

Six failure modes follow from that difference. The NSA's May 2026 design considerations for the Model Context Protocol, MCP, name most of them from the outside, and conclude that the protocol's posture “remains uneven and highly dependent on implementation discipline rather than protocol guarantees.”

Possession stops being sufficient. MCP servers rely on bearer tokens “without specifying protocol-level token lifecycle management,” and rotation is a recommended practice rather than a requirement. The incumbent answer is

shorter lifetimes, which narrows the window without closing it, because any live token is still a holder that may act. A grant spent by the act it authorized separates holding a credential from being able to act with it. A credential without the matching observed origin cannot be redeemed, so theft gets you the artifact and none of the authority.

The name stops standing in for the act. The brief describes “tool invocation path confusion,” where naming collisions let an actor “trick the system into loading actor-controlled code.” Every control that authorizes by tool name sits downstream of that substitution. A grant written against the durable class of the act means a substituted tool still has to produce the act, and the act is what was authorized.

Authority stops carrying forward. The brief records a server that “advertised a benign instruction at the time of installation and switched to a malicious instruction after the MCP server's second usage,” and notes that capability changes to an already-trusted server often proceed without fresh approval. Consent granted once is authority carried forward. The reverse path is the same failure at machine speed: a tool response steers the agent, the agent acts, and the act is permitted on authority granted before the influencing content existed. Deriving authority per act removes the carry in both cases. It does not remove influence, and Proposition 5 says so.

Delegation stops widening as it travels. The standards survey finds no deployed protocol that traces authorization back to the originating human across hops, and none that forces each step to narrow rather than widen. The NSA brief names the matching operational failure, where one agent's output is treated as valid by the next without verification. Where the origin of an act is established by observation at each hop rather than taken from what a relay presents, and re-established for every act rather than inherited from the last, a hop cannot credit itself as the origin, and a relay cannot enlarge a scope that was fixed when the grant was issued.

The tool does not have to cooperate. The governed process is unmodified and holds no credential of its own. There is no SDK to adopt, no sidecar to run and no framework to integrate, which is what makes this deployable against tools a team did not write and cannot change.

The decision stops requiring a round trip. The brief's egress recommendation is a filtering proxy or a data-loss-prevention (DLP) solution, which is sound wherever there is an egress path to filter. Air-gapped estates, edge and embedded deployments on intermittent links, and regulated environments where an authorization decision is not permitted to leave the boundary have no such path, and a cloud-brokered decision there either fails open or does not deploy. Evidence taken where the act commits allows the verdict to be reached there too, with no call off the host.

What this layer does not address is real and worth naming: resource exhaustion and denial of service, discovery of unauthorized servers on a network, vulnerability management in the implementations themselves, and the protocol specification work the brief calls for. Where code does execute inside an authorized process, the position is narrower than prevention. We do not stop it from running. Its acts are adjudicated on the same terms as any other, because authority binds to the writer the kernel observed rather than to the process that holds the credential.

Several questions an engineer will reach for are not answered here: how policy is authored, what a denial looks like in operation, and how the grant is structured and verified. Those are withheld on purpose rather than missing, and they are the substance of a design-partner conversation.

PROPOSITION 9

What we authorize, and what we do not.

The common objection is that an agent's real blast radius is external, a transfer executed against a payments API over TLS, and that the kernel sees only a socket write to an address. The objection assumes the decision happens at the network edge. It does not.

The decision precedes the act. Every term of it is bound to something the kernel observed rather than something the actor supplied, which is what makes the verdict unforgeable by the party it governs. The grant is good for that one act. And the decision requires no call off the host, which is what makes the layer deployable where the network is absent, restricted, or not permitted to carry an authorization decision.

We authorize which counterparty may be reached, by which process, under which act, before anything leaves the host. We do not authorize what is said to a counterparty that is already authorized.

That second sentence is the boundary. A transfer of the wrong size to an authorized destination is inside the grant. That is disclosure semantics, and it belongs to a composed layer rather than to this one.

WHAT IT DOES NOT DO

Influence, as distinct from authority. A tool response can still inform a plan the agent then executes inside its legitimate scope. Since influence cannot be prevented, containing authority is the control that remains. It is not a claim that influence is contained.

Meaning. The kernel sees unforgeable facts, not intent. The design decision that follows is to bind on what was observed rather than parse what was presented, which is why no payload inspection and no proxy appear anywhere in the architecture.

Tenancy. The layer requires control of the host. A tenant inside someone else's managed runtime cannot deploy it, though the operator of that runtime can, underneath them.

WHAT WOULD MAKE THIS WRONG

One condition, and it is structural. The derivation rests on legibility and unforgeability being anti-correlated, which is what keeps their overlap inside the kernel's view of the act. If a mechanism made a high-layer observation genuinely unforgeable, whether through hardware-attested application-level conduct or a verifiable-computation scheme that proves a semantic act occurred rather than that trusted code ran, the overlap widens past the kernel, the boundary stops being terminal, and architectures closer to the meaning win.

That is the one to watch. Everything else in this argument is downstream of it.

THE EVIDENCE BEHIND THIS

Nothing here is a plan. SentoClaw built this layer and runs it. We have reduced per-act authorization at this boundary to practice and run it under sustained load on two processor and kernel generations, including deliberately induced failure conditions and captured-credential attempts. Every architectural statement in this paper resolves to a registered reduction-to-practice record or a dated run in our engineering corpus.

What a verdict costs. We have measured it on both processor architectures, and we are not publishing a figure here. A number without the conditions it was taken under invites the wrong comparison, and the conditions are part of the answer. The shape we will state, because the shape is structural. The decision is local compute with no network call in its path, so it is paid in the same currency as the work the process was already doing. Our own records put the time around the verdict rather than in it: on the deny path the internal decision cost sits below the resolution of the round trip that carries it. The measurement ledger and its run logs open up on engagement.

Those are our own measurements on our own infrastructure. They are not independently audited, and we would rather say so than imply otherwise. Run logs, test records and the underlying evidence are available upon engagement.

SentoClaw is patent pending.

ABOUT SENTACLAW

SentoClaw, Inc. is a Delaware company building per-act authorization for autonomous workloads. The enforcement layer is offline by design: the decision is local, there is no call home, and the same posture runs on cloud fleets, on edge and embedded hardware, and inside air-gapped estates. It works alongside identity providers and policy engines, with no modification to the agents or tools it protects.

We are selecting a small number of design partners: teams running agents on infrastructure they control, who want a direct line to the people building the layer. Run logs, test records and the underlying evidence open up at that point.

Cody Parker Opstedal, cody@sentoclaw.ai · sentoclaw.ai

REFERENCES

1. *AI Identity: Standards, Gaps, and Research Directions for AI Agents*. arXiv:2604.23280, April 2026. arxiv.org/abs/2604.23280
2. Emergence AI. *Emergence World: A Laboratory for Evaluating Long-horizon Agent Autonomy*. emergence.ai/blog/emergence-world-a-laboratory-for-evaluating-long-horizon-agent-autonomy
3. Emergence AI. *Emergence World: A Platform for Evaluating Long-Horizon Multi-Agent Autonomy*. arXiv:2606.08367. arxiv.org/abs/2606.08367
4. Hugging Face. *Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident*. huggingface.co/blog/agent-intrusion-technical-timeline
5. Cilium Tetragon. *Enforcement*. tetragon.io/docs/concepts/enforcement/
6. Red Hat. *SELinux Architecture*, SELinux User's and Administrator's Guide. docs.redhat.com
7. National Security Agency, Artificial Intelligence Security Center. *Model Context Protocol (MCP): Security Design Considerations for AI-Driven Automation*. Cybersecurity Information Sheet, May 2026. nsa.gov/Portals/75/documents/Cybersecurity/CSI_MCP_SECURITY.pdf

SentoClaw, Inc. (Delaware) · Cody Parker Opstedal, Chief Executive Officer · Patrick Rademacher, Chief Technology Officer
sentoclaw.ai · cody@sentoclaw.ai · Position paper. SentoClaw is patent pending.