

Defensive Mesh for Enterprise Agent Deployment

Bounded authority and verifiable operations in cybersecurity

SOCHQ | Technical position paper | 20 September 2026

Abstract

Enterprises need a way to deploy agents with useful authority while keeping their activity observable, constrained, and accountable. In cybersecurity, agents can investigate incidents, choose tools, and propose changes to production systems. Deploying them requires a way to constrain their effects when their reasoning is wrong, their evidence is incomplete, or their infrastructure is unavailable. Existing security platforms already automate important parts of response. The question is how to extend those capabilities to adaptive agents while retaining enforceable authority and observable outcomes.

We propose a defensive mesh: distributed sensing and enforcement connected through a policy control plane, with explicit contracts governing agent actions. Each contract binds an operation to its evidence, authorized scope, impact limits, expiry, and verification requirements. The architecture treats missing telemetry, partial execution, and disconnected components as normal operating conditions. It supports local executors and scoped service APIs, with inference placement determined by data policy and measured performance.

Our thesis is that enforceable action contracts and continuous operational visibility provide a foundation for enterprise agent deployment. Our contribution is a reference architecture and evaluation method, with defensive security as the first application. SOCHQ is an implementation of this direction; implementation maturity is distinguished from proposed requirements. We hypothesize that the architecture can reduce time to verified containment without increasing harmful intervention, compared with competent existing automation under matched conditions. That claim requires comparative measurement. The paper defines how to test it.

Executive summary

The commercial opportunity is to supply the control infrastructure that lets enterprises deploy agents into real operations. Defensive security is the initial application because it exposes the problem sharply: a useful agent may discover an attack path that a fixed workflow did not anticipate, while a mistaken response can interrupt the business. The deployment architecture must preserve flexibility in investigation while constraining production changes through independently enforced policy.

A defensive mesh links endpoint, identity, network, and security-platform controls through a shared action contract. Its value should be judged by how reliably it converts evidence into authorized intervention and checks the resulting state. Agent count and model sophistication are insufficient measures.

This deployment argument extends beyond a security copilot. An enterprise needs to know which agents are enrolled, what capabilities they possess, what they are doing, and whether their authority can still be withdrawn. Monitoring must cover the declared execution surface, including denials, failures, and gaps. Claims about governing other vendors' agents or arbitrary enterprise activity require separate integration and enforcement evidence.

The practical test is whether a security team can answer four questions for every consequential action: What evidence justified it? Who or what authorized its exact scope? What limited the possible damage? What observation established the outcome? The same answers must remain meaningful during failures and concurrent incidents.

SOCHQ reports that the system is running at a first customer and that a broader rollout is starting at a second, as of 20 September 2026. This establishes a deployment context for evaluation. It does not establish comparative effectiveness, the scope of deployed autonomy, or validation of every requirement in this paper. Those are separate claims with separate evidence needs.

1 The operational problem

Consider a compromised workstation whose credentials are used to reach a second host and a cloud service. Isolating the workstation may interrupt one path while leaving an active session elsewhere. A response system must connect observations across resources, select interventions with appropriate authority, and check which paths remain available. If a sensor disappears during the response, the system must report uncertainty rather than infer that the threat has disappeared.

This problem predates language models. Adaptive agents increase its urgency because they can generate and execute sequences of tool calls with less direct supervision. Anthropic's September 2026 threat report describes operators using parallel subagents and persistent campaign records in cyber operations. These are provider observations of selected cases, not a population estimate or evidence that thousand-agent intrusions are typical. [\[1\]](#)

Controlled research also demonstrates that planner and specialist agent teams can improve performance on selected exploitation tasks. Such experiments establish capability under defined conditions; they do not establish universal superiority over defenders or predict enterprise compromise rates. [\[2\]](#)

Defensive capability also has limits. The SecRespond preprint evaluates 23 models on ten post-compromise host environments represented by forensic snapshots. It reports incomplete detection and remediation planning across the tested models. Because its task uses snapshots and report-based assessment, it does not measure live containment or compare production defense architectures. It does support treating an agent's completed investigation as something to evaluate, rather than accept at face value. [\[3\]](#)

Our design responds to this combination: more adaptable tool use, incomplete knowledge, and consequential actions. It remains relevant whether an attacker uses a single script, a human operator, or a coordinated agent workflow.

2 Existing foundations and the proposed contribution

Control planes and automated response are established foundations. NIST's Zero Trust Architecture separates policy decisions, administration, and enforcement. OpenC2 defines machine-readable cyber-defense commands and responses. CACAO describes security playbooks, including workflows, agents, targets, authentication, and signatures. These are useful precedents for the interfaces in a defensive mesh. [4] [5] [6]

Existing products also execute coordinated response. Microsoft documents incident-level automatic attack disruption across multiple security domains, together with audit-mode validation, staged rollout, exclusions, and reversible response actions. This is vendor documentation of available capabilities, not an independent comparison with SOCHQ. It is sufficient to reject a blanket characterization of current platforms as dashboards that cannot act. [7]

The term security mesh also has prior industry usage for distributed security capabilities and shared identity, analytics, and policy. We use defensive mesh to describe the specific deployment and execution model developed here, without claiming to originate the broader mesh concept. [8]

The proposed contribution is the integration of explicit action contracts, evidence freshness, bounded authority, distributed failure behavior, and outcome verification for adaptive defenders. Whether this integration is distinct from a particular competing implementation requires direct comparison. This paper makes no first-of-its-kind claim.

NIST's February 2026 concept paper on software and AI agent identity and authorization independently identifies related questions around agent permissions and accountability. It is exploratory work, not a finalized standard or endorsement of this design. [9]

3 Scope and threat model

A defensive mesh comprises sensors, investigators, a policy authority, executors, and verification services. A control plane coordinates their work and maintains shared records. Mesh describes distributed defensive coverage; it does not require unrestricted peer-to-peer communication. An agent is a software component that selects or proposes successive actions. A deterministic executor that applies an approved operation need not itself use a language model.

The protected environment may include managed hosts, cloud workloads, identity services, network controls, and SaaS platforms. Some resources support a local agent; others expose an administrative API. The architecture requires an appropriate enforcement point, not an agent installed on every resource.

The adversary may control application content, filenames, logs, messages, or other evidence consumed by investigators. It may compromise a monitored host, steal a delegated credential, interrupt communication, or induce conflicting response proposals. Components may also fail without an attacker: events arrive late, clocks drift, services reject requests, and operators approve an action that later becomes inappropriate.

The design assumes that some enforcement and authorization roots remain trustworthy. A compromised privileged endpoint can falsify its observations or bypass local controls. A compromised authority that can change every policy and signing key defeats protections that depend exclusively on it.

Customer-controlled constraints and independent observation can reduce these risks, but the architecture does not eliminate the need to protect its own infrastructure.

The requirements in Sections 4 through 8 are proposed architecture requirements. They are not a claim that every control is currently implemented in SOCHQ. Section 11 describes the available implementation evidence separately.

4 Architecture and the action contract

The architecture separates probabilistic investigation from the authority to mutate production. Investigators may assemble evidence, test hypotheses through permitted reads, and propose actions. A separate policy mechanism decides whether a specific proposal is permitted. Executors enforce that decision at the relevant resource boundary.

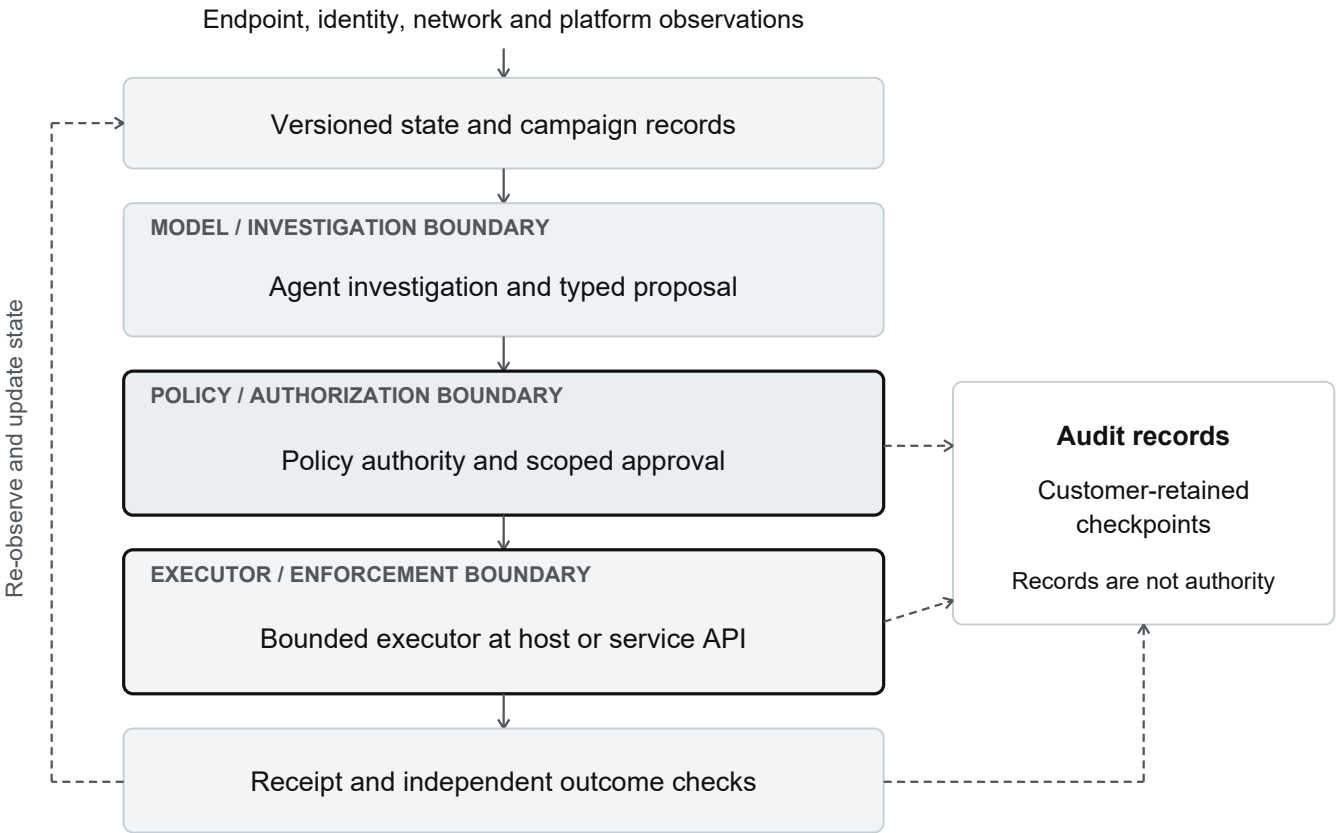


Figure 1. Proposed control flow. Evidence can inform a proposal but cannot grant authority. The audit service records decisions and observations; it is not the policy authority.

The shared state includes resource identities, relationships, observation times, provenance, and known coverage gaps. A campaign record links an incident hypothesis to its evidence and outstanding work. Facts, inferred relationships, and unresolved hypotheses remain distinguishable. The record persists independently of a chat session.

We describe this as stateful, policy-constrained control. The representation is incomplete and may be stale; it is not an omniscient model of the environment. Language-model proposals may vary.

Authorization must nevertheless follow explicit rules over a versioned proposal, relevant observations, and the applicable policy.

The execution contract

Every production mutation should carry enough information for both the authority and the executor to reject an inappropriate operation.

Contract element	Required meaning
Identity and target	Tenant, requesting identity, executor, resource identifiers, exact operation and arguments
Evidence and state	Evidence references, observation age, relevant state version, explicit unknowns
Authorization	Applicable policy version, approval or delegated capability, protected-resource restrictions
Impact bounds	Per-action scope plus reserved capacity within aggregate action and disruption limits
Time and replay controls	Issuance and expiry, unique action identifier, replay protection, idempotency behavior
Preconditions	Conditions checked again immediately before a production change
Outcome and recovery	Expected postconditions, verification source and interval, cancellation and recovery behavior

A signature authenticates an instruction; it does not establish that the instruction is safe. The executor must validate scope, expiry, local constraints, and relevant preconditions. Approval binds the exact plan and arguments. A material change requires renewed authorization.

This contract also addresses coordination. Concurrent proposals must reserve shared impact budgets and obey a resource conflict policy. Duplicate deliveries must not repeat a mutation. Cross-service response usually cannot be one atomic transaction; the system must record partial application, reconcile actual state, and execute only authorized recovery steps. The same recovery operation can itself require an action contract.

Inference placement and data handling

Inference may run on customer-controlled infrastructure or through an approved external provider. Placement depends on permitted data destinations, operational dependencies, model capability, and measured latency. Running reasoning locally does not automatically make it faster, private, or available; those properties require measurement and a complete account of data flows.

Data policy must cover prompts, retrieved evidence, embeddings, traces, diagnostics, support exports, and retention. Security event collection should continue independently of an inference outage where the

deployment supports it. For a claim that data never leaves a boundary, the evidence must address every relevant egress path.

Monitoring the enrolled workforce

The control plane should expose enrollment and software versions, active capabilities, workload assignments, tool requests, authorization decisions, execution attempts, results, and outcome checks. Each record needs a stable agent and task identity. Heartbeats, sequence gaps, stale observations, and missing receipts help distinguish inactivity from lost visibility.

An assertion that all agent activity is monitored requires mediation of every in-scope tool and execution path, including child processes, delegated agents, service APIs, and recovery actions. An agent with usable credentials or tools outside that boundary can bypass the record. A deployment must therefore inventory those paths, restrict unmediated access, test bypass attempts, and report uncovered paths explicitly. Monitoring enrolled SOCHQ components does not by itself establish oversight of unrelated software or third-party agents.

5 Authority and bounded autonomy

Autonomy should be granted to an action class in a defined environment. It should not be a general property assigned to an agent persona. A policy might permit limited diagnostic collection automatically, require approval for temporary isolation of an ordinary workstation, and prohibit automated changes to designated critical services.

In observation mode, an investigator produces proposals without production mutations. In approval mode, an authorized operator accepts a specific action. In bounded autonomous mode, a policy permits a narrow class of actions within explicit conditions. Promotion between these modes requires evidence of effectiveness and acceptable harm, including representative benign activity and failure conditions.

Agreement with an analyst is useful feedback, but agreement is not correctness. Evaluation must measure missed threats, unnecessary interventions, business disruption, and whether the intended outcome occurred. Material changes to models, tools, policies, or the environment should trigger a defined revalidation process. Evidence from one customer or action class should not silently authorize another.

Impact limits must account for composition. A system that offers only single-host isolation can still disconnect an entire fleet through repeated calls. The authority therefore needs limits on concurrent affected resources, action rate, campaign scope, and disruption of shared services. Pending operations and retries must consume the appropriate budget. Protected-resource rules should apply even when several individually plausible proposals arrive at once.

Human oversight also has capacity limits. A response design that depends on an immediate human approval under every burst of events may not meet its latency objective. Tests should measure approval queues and operator workload. Narrow preauthorization can reduce that dependency, but only for the specific actions whose risks and outcomes have been evaluated.

6 Failure behavior and revocation

A central authority simplifies policy consistency while creating a valuable target and an availability dependency. The architecture should separate investigation, dispatch, authorization, and signing where practical; isolate tenants; and distinguish task-signing keys from software-update keys. A compromised dispatcher should not be able to expand a locally enforced customer policy.

An emergency stop has several meanings. The control plane can immediately reject new authorizations. Reachable executors can acknowledge cancellation. A disconnected executor cannot receive an instantaneous stop message. Its remaining ability to mutate must instead be bounded by previously issued authority and locally enforced expiry.

The maximum revocation delay depends on authorization lifetime, connectivity, enforcement interval, clock behavior, and whether an operation can be interrupted safely. The system must report those assumptions and measure the bound. For irreversible operations already completed, revocation cannot undo the effect.

Condition	Required declared behavior
Control-plane connection lost	Stop new mutations when authority expires; specify whether sensing and buffered logging continue
Emergency stop received	Reject new mutations and apply the declared cancellation rule to in-flight work
Isolation timer expires	Apply the configured recovery policy and record that containment may have ended
Executor crash or host reboot	Recover containment state and expiry correctly, or explicitly report loss of assurance
Stale or contradictory evidence	Restrict affected actions, obtain fresh observations, or request review
Aggregate budget exhausted	Deny or defer additional affected actions, including concurrent proposals

Isolation expiry is a separate safety decision from authorization expiry. Automatically restoring connectivity can reduce the duration of an accidental outage while also reopening an attack path. Preserving quarantine has the opposite availability tradeoff. The deployment must choose and test this behavior for each resource class. A timeout alone cannot guarantee both continued containment and service restoration.

7 Evidence integrity and outcome verification

Three claims must remain separate. Authorization evidence establishes that an identified authority permitted an operation. Execution evidence records what an executor reports it did. Outcome evidence establishes whether an observation supports the intended effect within a declared scope and interval.

For example, acceptance of an isolation command is not evidence that network isolation took effect. A host-local firewall result is stronger execution evidence, but independent network observation can provide a separate check. Neither proves that stolen credentials or cloud sessions have been contained.

The system should retain distinct states for denied, expired, cancelled, partially applied, verification pending, verified, unverified, and contradicted operations. Verification must identify its source and independence limitations. If the only evidence comes from the potentially compromised executor, that dependency must remain visible.

Audit delivery failure also needs an explicit policy. A deployment may block a consequential action until a durable record exists, use a recoverable local journal, or permit an action with a visible recording gap. The choice affects both availability and accountability. A path that permits an unrecorded mutation cannot support an unconditional claim of complete action history.

A hash-chained record can support integrity checking against a trusted prior commitment. Recomputing an exported chain alone cannot establish that events were never omitted, that a suffix was not deleted, or that a privileged writer did not replace the history. Signed checkpoints retained outside the ordinary control plane, sequence tracking, and comparison across witnesses strengthen the evidence. Certificate Transparency provides relevant prior art for signed log commitments and consistency checking, although its application differs from defensive operations. [10]

Even a cryptographically consistent record can contain a false observation. Audit verification should therefore describe the property checked: consistency of recorded history against retained checkpoints. Security effectiveness requires additional evidence.

8 Protecting the defenders

Investigators consume potentially hostile material. An attacker can place instructions in telemetry, ticket text, retrieved documents, or other tool results. AgentDojo demonstrates the relevance of prompt injection to tool-using agents; its benchmark is not a measurement of SOCHQ. [11]

The architecture treats observed content as evidence with provenance, never as authorization. A model cannot grant itself a capability, rewrite policy, or remove an approval requirement. Typed proposals, argument validation, constrained tool credentials, independent policy enforcement, and adversarial evaluation are required controls. Prompt wording alone is insufficient as an enforcement boundary.

Read-only work also needs limits. Evidence collection can expose secrets, increase load, or exhaust API budgets. Investigation policy should bound data access, collection volume, concurrency, retention, and permitted destinations. Specialized investigators should receive the capabilities required for their task rather than inherit a shared administrative identity.

Untrusted threat intelligence and attack-simulation findings can inform investigation, but should retain their source and confidence. Promoting such material into trusted policy or detection logic requires a separate controlled process. The same distinction applies to lessons derived from previous agent runs.

9 Worked response scenario

The following scenario illustrates the proposed architecture; it is not a reported customer incident or measured result.

A workstation produces suspicious process activity. Related identity observations indicate use of the same account against a second resource. The mesh creates a campaign record containing the observed facts, the suspected relationship, and a gap in telemetry from a third resource.

An investigator proposes temporary workstation isolation and an identity-session intervention. The authority evaluates each proposal against resource criticality, observation freshness, current approval rules, and the aggregate impact budget. If the session action has no authorized integration, the record identifies that gap; host isolation does not close the campaign.

An operator approves the exact isolation plan. Before execution, the local executor checks that the resource identity and relevant preconditions still match. It applies the operation idempotently and reports the result. A separate observation tests whether the specified communication path is blocked. The campaign now records verified isolation for that scope, while the identity path remains unresolved until separately checked.

If communication with the control plane then fails, the endpoint follows its existing authority lease and declared isolation-expiry policy. The campaign record exposes the loss of current verification. If connectivity is restored on expiry, the system records that change rather than continuing to display containment as verified.

This example gives the mesh a concrete obligation: preserve the distinction between partial progress and a verified campaign outcome, including when the infrastructure itself fails.

10 Evaluation and falsifiable claims

Enterprise deployability has two dimensions to evaluate. First, does the system preserve its authority and visibility contract when an agent or dependency fails? Second, does the permitted work deliver sufficient security value at acceptable cost and harm? A faster response does not compensate for an unobservable or unauthorized execution path.

The primary hypothesis is that a governed defensive mesh reduces time to verified containment without increasing harmful intervention, relative to a competent baseline with comparable observations, permissions, response policy, and resources. A failure to improve this tradeoff would weaken the case for introducing another control layer.

Compare three configurations: the existing security stack with its supported automation; an agentic orchestrator using the same integrations and constraints; and the proposed mesh. Match telemetry and privileges for the architecture comparison. Evaluate any advantage from added sensors or privileges separately. Record model versions, compute and tool budgets, operator availability, network conditions, and configuration effort.

Use repeatable scenarios with independent ground truth and realistic benign activity. Include endpoint-to-identity movement, cloud-only activity, unmanaged resources, incomplete telemetry, and attacks against the defensive agents. Vary concurrent campaigns, event volume, proposal rate, and dependency latency. A thousand agents is useful only as a precisely defined workload, not as a proxy for difficulty.

Measure	Operational definition
Time to verified containment	Elapsed time from a predefined actionable observation to independent confirmation that declared attack paths are blocked
Containment success	Fraction of scenarios meeting the stated outcome before a fixed horizon, with failures and timeouts retained
Harmful intervention	Unnecessary or policy-violating actions, affected resources, and benign-service disruption duration
Residual exposure	Ground-truth paths or persistence that remain usable at the assessment horizon
Authority and revocation	Forbidden mutations and time until new actions cease after revocation under each failure condition
Verification quality	False verified outcomes, unresolved outcomes, and evidence-source independence
Monitoring coverage	Fraction of independently observed in-scope operations linked to control-plane records, with missing events and bypasses identified
Operational cost	Analyst effort, infrastructure and inference cost, host overhead, deployment and maintenance effort

Decompose response time into sensing, investigation, approval, dispatch, execution, and verification. Report end-to-end latency from a known attack start as well, to expose detection delays. Do not report latency only for successful trials. Predeclare the handling of censored failures and publish success proportions alongside latency distributions.

Use paired scenarios, repeated runs, held-out variants, and uncertainty intervals. Choose sample sizes and acceptable harm bounds before inspecting outcomes. Human reviewers should adjudicate ambiguous harm and security outcomes using independent evidence. A model's own success report is not ground truth.

Failure tests should cover duplicate dispatch, stale approvals, clock rollback, reboot during isolation, network partition, partial API failure, overlapping campaigns, injected instructions, forged receipts, compromised dispatch, and loss of an audit suffix. Publish the expected behavior and observed result for each test. Passing a finite suite is evidence for those conditions, not proof against every attack.

Ablations should remove one proposed source of benefit at a time: shared campaign state, local inference, outcome checks, or aggregate impact controls. This establishes which components improve effectiveness or safety, and which merely add overhead. None of these comparative results is asserted in this paper.

11 SOCHQ implementation and evidence status

SOCHQ combines a control plane with host execution, security observations, and enrolled specialist services. Its product vocabulary includes RMM for host management and execution, XDR for security observations, and stack stewards for monitoring and operating services such as Wazuh and Graylog. The proposed architecture gives those components a common authority and evidence model.

The implementation record below combines the supplied company inventory, the September 20 deployment update, and a bounded review of control-plane source and committed artifacts at revision 2942fd375a223435a118f37de961bf3f97bfc93f, dated September 9, 2026. Repository inspection is evidence of code and recorded tests; it is not a fresh execution of those tests or confirmation of a customer's deployed configuration. Deployment at a customer does not imply that all capabilities or autonomy modes are active there. [12]

Area	Available evidence	Boundary of the evidence
Customer use	20 September 2026 update reports a running first-customer deployment and a broader second-customer rollout starting	Deployment scope, versions, active permissions, dates, operating duration, incident and availability records
Authority controls	Source includes approval stages, task-expiry checks, optional idempotency handling, and signed-intent enforcement hooks	Enforcement depends on configuration and supported paths; source inspection alone does not establish deployment-wide coverage
Host isolation	Committed September 3 Windows lab receipt records isolation and release; a separate narrative records local dead-man release following a control-plane partition	Company-produced lab evidence; not independently reproduced or proof of customer containment, cross-OS behavior, or reboot recovery
Enrollment	September 3 canary report records a newly enrolled XDR component operating beside an existing RMM component	This report does not demonstrate a fresh two-component installation from one token
Audit	Source includes per-tenant chain verification; a run report describes a 2,000-entry export and deliberate mutation detection	The exported pack was not available in the reviewed repository; audit writes are best-effort, so complete durable recording is not established

Area	Available evidence	Boundary of the evidence
Stop and autonomy	Source gates newly claimed tasks on stop state; documents describe restricted internal containment and narrow autonomous operations	Previously dispatched work, disconnected executors, broader action classes, and automatic autonomy promotion require separate evidence
Residency and coverage	Documents describe inference placement and a broader monitoring architecture	No measured never-leaves residency result or comprehensive current customer coverage inventory was established in this review

The lab records support a concrete preliminary observation: the implementation has exercised host containment, local timeout recovery, and audit consistency checking in specified internal conditions. They do not yet establish the comparative hypothesis in Section 10. The distinction between an actuator test, a complete approval-to-outcome workflow, and a customer deployment must remain visible.

The September 20 deployment update extends the earlier inventory but does not establish which technical items changed. Published effectiveness claims should be accompanied by versioned artifacts and a defined observation period. A customer case study should identify the authorized scope, initial state, actual intervention, independent outcome, operational impact, and remaining uncertainty, with publication permission for customer information.

For readers evaluating deployment now, the immediate diligence question is which action contracts have supporting evidence in their environment. Broad architectural coverage should not be substituted for that answer.

12 Implications

A defensive mesh earns its place when it makes adaptive response both more effective and more controllable within a defined environment. Its practical contribution is the connection between investigation, scoped authority, distributed enforcement, and outcome checks. Existing security systems provide important parts of that foundation and should be evaluated as capable baselines and integration partners.

The next step for SOCHQ is to turn operational deployments into inspectable evidence: exact action contracts, failure traces, independent verification, and comparative results. If those results show faster verified containment at acceptable harm and cost, the case for the architecture becomes measurable. If they do not, the architecture and its claims should change.

The broader enterprise thesis is that useful agent authority needs control infrastructure for permissions, observation, intervention, and evidence. This paper specifies that proposition for defensive security and sets out its evaluation. Extending it to other enterprise workflows is a further application to validate, not a result established here.

References

1. Anthropic. [Detecting and countering misuse of AI](#) September 2026. Provider threat report; selected observed cases, including GTG-10007.
2. Zhu, Y., et al. [Teams of LLM Agents can Exploit Zero-Day Vulnerabilities](#). arXiv:2406.01637v2, revised 30 March 2025. Controlled research study.
3. Wang, L., et al. [SecRespond Benchmarking AI Agents for Real-World Post-Compromise Incident Response](#). arXiv:2607.26791v1, 29 July 2026. Preprint; forensic snapshot and report-based evaluation.
4. Rose, S., et al. [Zero Trust Architecture](#). NIST SP 800-207, August 2020.
5. OASIS. [Open Command and Control Language Specification Version 1.0](#). Committee Specification 02, 24 November 2019.
6. OASIS. [CACAO Security Playbooks Version 2.0](#). Committee Specification 01, 27 November 2023.
7. Microsoft. [Automatic attack disruption in Microsoft Defender](#). Product documentation, updated 23 June 2026. Vendor capability description.
8. Gartner. [Gartner Security and Risk Management Summit India Day 1 Highlights](#). 17 March 2021. Primary industry terminology source.
9. NIST NCCoE. [Software and AI Agent Identity and Authorization](#). Project accompanying the February 2026 draft concept paper; exploratory work.
10. Laurie, B., et al. [Certificate Transparency Version 2.0](#). RFC 9162, December 2021. Experimental RFC; used as audit-design prior art.
11. Debenedetti, E., et al. [AgentDojo A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents](#). arXiv:2406.13352v3, revised 24 November 2024.
12. SOCHQ. [Control-plane repository snapshot](#). Revision 2942fd375a223435a118f37de961bf3f97bfc93f, 9 September 2026. Private source and company-produced artifacts reviewed 20 September 2026; supporting materials are not publicly released. Relevant records include I3_CANARY_RECEIPT_2026-09-03, I3_STEP7_PARTITION_RECEIPT_2026-09-03, D1P_LIVE_PACK_EXPORT_2026-09-03, and A1_LIVE_MESH_CANARY_2026-09-03.

External sources checked 20 September 2026. Company reports, source inspection, and recorded internal tests are distinguished above; no independent production effectiveness study is claimed.