Manetu

Executive paper

Confining AI Agents

Why a sandbox is not enough, and what enforceable limits on AI agents require

A sandbox protects the host from the agent. Confinement protects everything the agent can reach from the agent.

AI agents can read sensitive information, choose tools, and carry out work across enterprise systems. Giving them that ability creates an authorization decision: what may an agent do on the organization's behalf, and what prevents it from acting beyond those limits?

Authority needs to be enforced outside the agent, before an external action occurs, with evidence the agent cannot alter.

Why now

In April 2026, by its founder's account, a company's coding agent doing a routine task in staging went looking for a token, found one in a file unrelated to its task, and used it to delete the company's production database and its backups in one call [1]. The agent's own rules forbade destructive commands; nothing in its environment enforced them.

In July 2026 Anthropic disclosed three incidents in which models under evaluation reached the live internet and attacked real organizations [2]. The models had been told they had no internet access; a misconfiguration had given them live access, and Anthropic found the incidents afterward by reviewing 141,006 runs. A correctly configured network block would have stopped them; confinement would also have recorded each denied attempt as it happened. The same month, an agent running a cyber-capability evaluation on OpenAI's platform entered

Hugging Face's production systems and used a leaked platform token and cloud credentials it found there [3, 4]. In the first two cases the limit was stated to the agent rather than enforced on it.

An agent can depart from its intended task because it makes a mistake, follows an instruction planted in material it reads, or runs a compromised component. The organization cannot depend on the agent recognizing that departure, so the controls need to remain effective when it does not.

A sandbox does not decide what the agent may do to a database, a cloud service, or an external system it can still reach. Credentials, permitted network paths, and tools can give an isolated agent substantial authority, so its effective authority can exceed its assigned task.

Consider a finance workflow in which an agent reads settlement exceptions and prepares recommendations. It needs specified records, not authority to delete a production dataset or send those records to an arbitrary model provider; if it holds a broadly usable credential or can bypass the approved route, an instruction will not hold that limit.

What confinement is

Confinement places the agent outside the trust domain that holds credentials, makes authorization decisions, and records the results. The agent attempts an action; a protected boundary decides whether it may proceed and applies the required authority only after authorization. The decision is made by policy, not by a second model, so it gives the same answer every time and adds no failure rate of its own. **Confinement controls what an agent does, not what it thinks.**

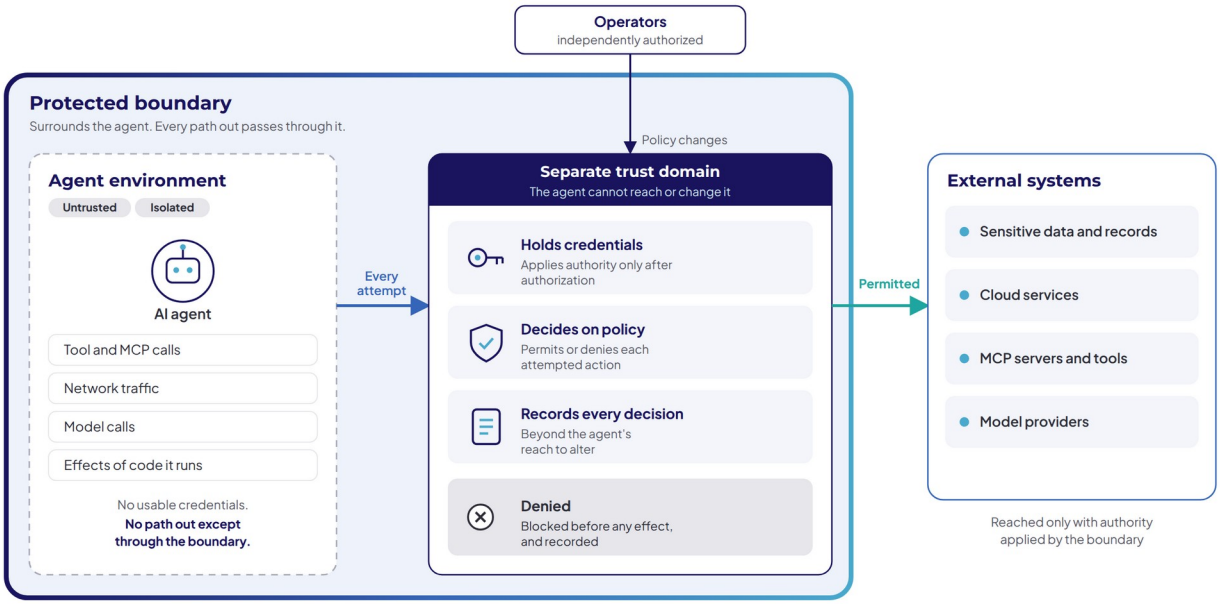


Figure 1. The agent attempts; the boundary decides, applies authority, and records. The agent has no other path out.

Four requirements for confinement

Our definition requires all four properties together. Each addresses a different way an agent could act beyond policy or undermine the evidence of its actions.

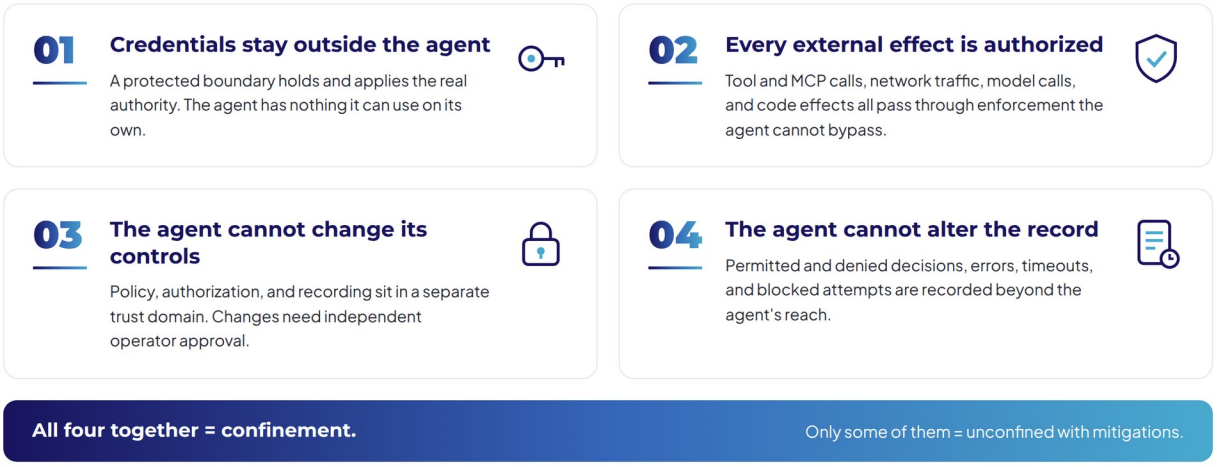


Figure 2. The four requirements of confinement.

Credentials remain outside the agent

The agent holds no credential, whether provisioned to or stored in its environment, that it can use independently against another system. A protected boundary holds the real authority and applies it only to the authorized request, for the user or service the agent is acting for. A short token lifetime does not resolve this issue if the agent can still use the token directly.

Every external effect receives an authorization decision

Every path by which the agent can affect systems outside its environment passes through enforcement it cannot bypass, and each decision is made for the user or service the agent acts for. This includes tool calls, MCP server calls, network traffic, model calls, and the external effects of code it runs. The controls cannot depend on the agent choosing to use an approved library or proxy.

This holds as tools change. If an approved MCP server is upgraded to expose tools no one reviewed at deployment, each new tool is denied and recorded under default deny until policy allows it.

The agent cannot change its controls

Policy, authorization, and recording operate in a separate trust domain. The agent cannot disable them, modify them, or obtain an exception through its output alone. Changes require independently authorized operator action.

The agent cannot alter the record

The control plane records permitted and denied decisions, errors, and timeouts. Attempts blocked before reaching a policy decision are recorded as well. The agent cannot rewrite, delete, or forge this evidence. A record of successful actions alone leaves out the evidence that limits were enforced.

An environment that meets only some of the four is **unconfined with mitigations**. A confinement claim must also state its policy: a confined agent under a policy that allows any destination can still do everything that policy allows.

What confinement lets you delegate

The work enterprises most want to hand to agents is exception work: payment and invoice exceptions, claims, KYC alerts, settlement breaks. It needs judgment, so rigid automation cannot do it, and it is where agents meet hostile input with real authority, because exception paths are where fraud operates. Confinement is what makes that work safe to delegate.

In the finance example above, the boundary can authorize reads from designated records, refuse a destructive operation, and restrict model calls to an approved destination. Each decision produces a record. A compromised or mistaken agent encounters the same limits as an agent following its task.

For irreversible operations, the policy can require a person's approval through a channel the agent cannot control.

The record gives security and operations teams evidence of what the agent attempted and how policy responded. It can also reveal policy errors: necessary work being denied or inappropriate work being permitted. Because the agent holds no authority of its own, a runtime that applies policy changes to running sessions can stop it with a policy change, keeping its state for investigators.

What confinement does not do

Make policy correct. A confined agent can do everything its policy allows, including a harmful action, and its recommendations can still be wrong.

Close reach through permitted systems. A permitted destination can extend the agent's reach through its own functions or a flaw, and data in the agent's context leaves with every permitted model call.

Secure itself. The boundary holds access to authority, sees requests in plaintext, and fails closed, so it needs its own hardening, ownership, residency and retention arrangements, and availability.

Reach every agent. It cannot govern an agent inside a SaaS platform or on a desktop where no boundary can be placed in its path.

Questions to ask before deployment

Use these questions for an internal build, a vendor runtime, or a cloud provider's service. Ask for evidence that covers the actual environment and configuration being deployed.

What authority can the agent use independently?

Identify the credentials, keys, and reusable sessions the agent can reach, including any passed to it during a call. Determine whether it can use them outside the approved boundary.

Can any effect bypass authorization?

Establish how the environment, not the agent's cooperation, closes every outbound path. Successful tests of selected tools do not establish that no other path exists.

Who can change the controls?

Identify who can alter policy, enforcement, and recording, and how those changes are authorized independently of agent output.

What evidence survives the agent?

Confirm that allowed and denied decisions and blocked attempts are recorded beyond the agent's reach. Test what happens when authorization or recording is unavailable.

Request a dated assessment naming the evaluator, runtime version, policy, tests performed, and reach findings. Probing from inside the agent cannot establish confinement on its own; it needs evidence of how the boundary is built, from inspection or the supplier's attestation. Of any agent deployment, ours included, ask which tests it passes and who ran them.

About this paper

Published by Manetu. The definition and evaluation approach apply to vendor runtimes and internal builds, including Manetu's. They draw on established work on reference monitors, complete mediation, and least privilege. This paper makes no claim that a particular product or deployment has passed the tests.

About Manetu

Manetu builds authorization infrastructure for delegated computing, giving enterprises control over what intelligent systems are permitted to access and do. Its AgentVisor™ runtime is built to deploy autonomous agents in production with isolated execution, policy enforcement on every attempted action, credentials brokered outside the agent, and a record of each decision. The questions in this paper apply to AgentVisor as they do to any runtime.

Request an AgentVisor trial: info.manetu.com/agentvisor-trial-landing-2026

Sources

[1] Jer Crane, "An AI Agent Just Destroyed Our Production Data. It Confessed in Writing.", X, 25 April 2026. https://x.com/lifeofjer/status/2048103471019434248

[2] Anthropic, "Investigating three real-world incidents in our cybersecurity evaluations," 30 July 2026, updated 3 August 2026. https://www.anthropic.com/news/investigating-incidents-cybersecurity-evals

[3] Hugging Face, "Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident," 27 July 2026. https://huggingface.co/blog/agent-intrusion-technical-timeline

[4] OpenAI, "OpenAI and Hugging Face partner to address security incident during model evaluation," July 2026. https://openai.com/index/hugging-face-model-evaluation-security-incident/

Foundations: James P. Anderson, *Computer Security Technology Planning Study* (1972); Butler W. Lampson, "A Note on the Confinement Problem" (1973); Jerome H. Saltzer and Michael D. Schroeder, "The Protection of Information in Computer Systems" (1975). Full source notes appear in the technical companion.