The Semantic Proxy Pattern

A defense-in-depth architecture for enterprise AI agent authorization

Attribit-ID · April 2026

Agents with tool access execute any instruction they receive — including ones from adversaries.

When an agent processes a malicious instruction embedded in a document, email, or API response, it executes with the agent's full permissions. The audit trail records the action as legitimate automation. Standard endpoint detection does not catch it.

Firewalls, RBAC, and model-level guardrails all fail for the same reason: they cannot evaluate intent.

Firewalls see IPs, not intent — exfiltration through an approved endpoint is invisible. RBAC assigns permissions, not contextual judgment. Model-level guardrails are probabilistic: 7% of adversarial prompts pass, and the guardrail can be overridden by the attack it's meant to block.

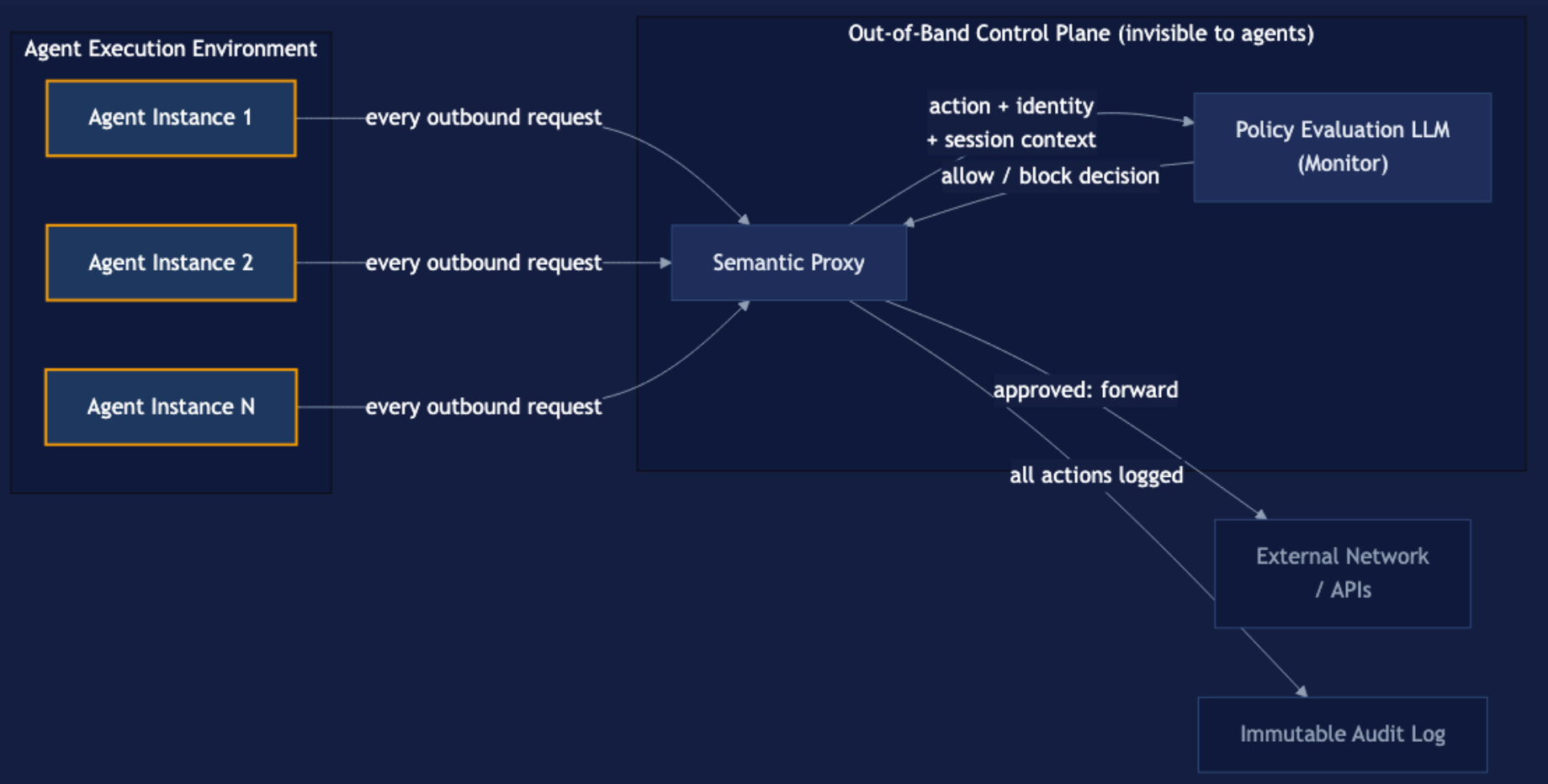
The semantic proxy pattern enforces authorization in three independent layers, each targeting a distinct threat vector.

One layer evaluates agent actions against policy before they reach the network. One makes that evaluation topologically mandatory. One attributes every action to a specific instance. All three must fail simultaneously for a breach; each covers what the others cannot.

The semantic proxy intercepts every agent action out-of-band and evaluates it against allowlist policy.

The proxy intercepts every agent request before the network. A separate policy LLM evaluates each against allowlist policy — pass proceeds, fail is blocked and logged. The agent cannot perceive this control. Allowlist semantics: unknown actions are blocked.

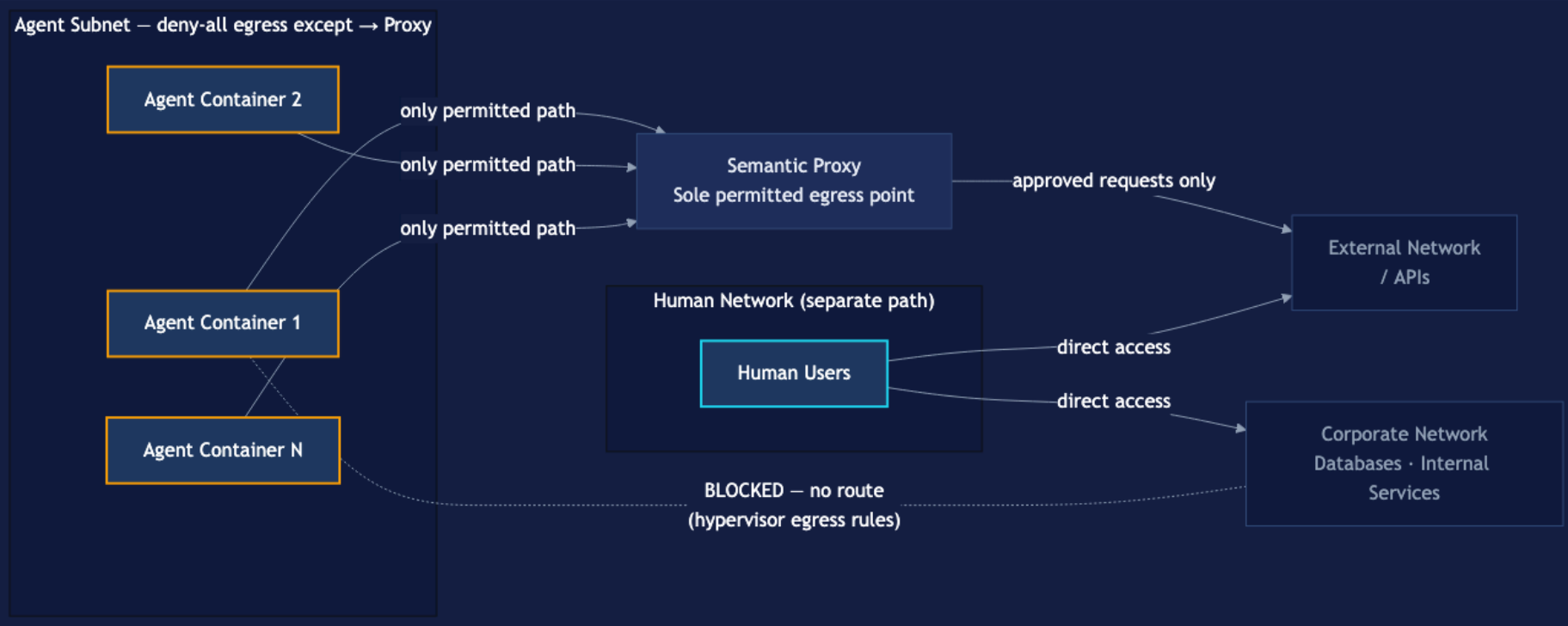
The semantic proxy: out-of-band, invisible to the agent, fail-closed



Subnet isolation converts proxy traversal from advisory to mandatory — a topological control that holds even under software compromise.

All agent execution environments sit in a dedicated subnet — deny-all egress, single permitted path to the proxy. Enforced at the hypervisor layer, outside any agent's control. A compromised agent's blast radius is bounded by the proxy's allowlist policy.

Subnet isolation topology: the proxy is the only egress path



Per-Actor cryptographic identity enables instance-level attribution, role-specific policy, and granular revocation.

Agent instances receive short-lived cryptographic identities at spawn — scoped to the specific task, expiring at task end. The proxy validates on every request, not just session start. This enables instance-level attribution, per-role policy, and revocation without fleet disruption.

Three independent failure modes compound — all three must fail simultaneously for a breach to succeed.

Prompt injection: proxy blocks the action — model compromise doesn't reach the network. Lateral movement: topologically impossible. Credential theft: per-instance tokens expire in minutes, real credentials stay at the proxy. Impersonation: mTLS rejects requests without the correct certificate.

Three constraints remain that this architecture does not fully close: adversarial monitor, policy completeness, and latency.

The monitor LLM can be targeted by an adversary who knows the policy. Policy completeness requires constant review as capabilities expand. LLM evaluation adds latency. Multi-agent systems require proxy coverage at each delegation hop, including Agentlets.

Sources

- vectra.ai/topics/prompt-injection — EchoLeak CVE-2025-32711, GitHub Copilot RCE CVSS 9.6, Cursor IDE CVSS 9.8
- reddit.com/r/cybersecurity/comments/1q95y5p/ — 300,000-prompt study; 7% baseline attack-through rate
- genai.owasp.org — OWASP Top 10 for Agentic Applications (December 2025)

The proxy does not make agents trustworthy. It makes their actions verifiable and their failures containable.

Read the full technical whitepaper — three-layer reference architecture, threat analysis, implementation roadmap, and regulatory alignment.

attribit-id.com/writing/semantic-proxy-pattern