Treat Your AI Agents Like You Treat Untrusted Code

Why the proxy architectural design pattern — semantic enforcement, subnet isolation, per-Actor identity — is the engineering answer to agent security

Attribit-ID · April 2026

The moment your agent can send an email, it can forward confidential data.

In 2025–2026, critical CVEs were assigned to Microsoft Copilot (CVSS 9.3), GitHub Copilot (CVSS 9.6), and Cursor IDE (CVSS 9.8) — all exploiting the same weakness: AIgentic Actors operating with ambient trust. Agents are in production. The attacks are real.

Traditional security controls cannot evaluate what an agent intends to do.

A firewall knows IPs, ports, and packet signatures. None of them can determine whether your recruiting agent is querying an approved job board or forwarding candidate data to an attacker's endpoint. That's a semantic problem. Syntactic tools cannot solve it.

The semantic proxy screens every agent action before it reaches the network.

An HTTP proxy intercepts all agent traffic and routes it through a separate LLM that evaluates each action against defined policy — then blocks at the network layer, without the primary agent knowing the control exists.

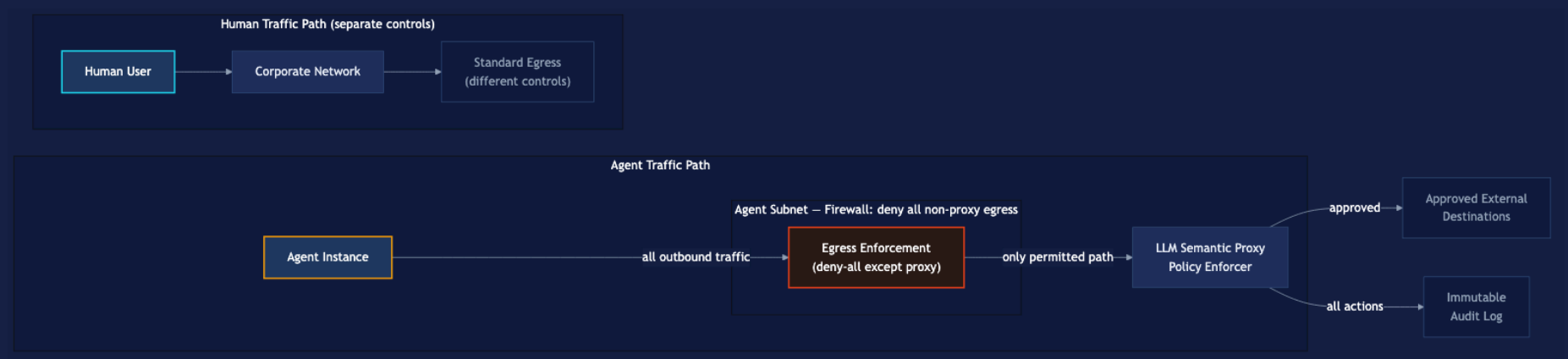
An allowlist policy fails closed — every novel attack is blocked by default.

The proxy enforces behavioral policy: only explicitly approved actions pass. A denylist blocks known-bad patterns but fails open — gaps become exploitable. An allowlist fails closed. When an agent encounters a novel attack vector, the default answer is no.

Subnet isolation makes the proxy mandatory; per-Actor identity makes it forensic.

Firewall rules force all agent egress through the proxy — no bypassing by a compromised instance. mTLS certificates scoped to individual task instances enable granular revocation, role-specific policy, and audit trails attributable to specific instances.

The three layers in practice: topology makes the proxy non-optional



Three independent failure modes compound to a 2,300-fold improvement in attack containment.

Basic prompt defenses alone: 7% of attacks through. Add content inspection: 0.2%. Add prompt injection detection: 0.003%. All three layers must fail simultaneously for a breach to succeed.

The monitor LLM can be targeted, and policy completeness is never finished.

An adversary who understands your policy can craft requests that pass screening while still causing harm. The architectural response: keep policy opaque to the primary agent and log everything — patterns visible only in aggregate can be detected retrospectively.

Every security leader whose organization runs agents in production is already in scope.

This architecture is not future-state. AIgentic Actors touching sensitive data, executing workflows, or calling external APIs today need proxy enforcement, subnet isolation, and per-Actor identity today.

Sources

- vectra.ai/topics/prompt-injection — CVEs: Microsoft Copilot CVSS 9.3, GitHub Copilot CVSS 9.6, Cursor IDE CVSS 9.8
- reddit.com/r/cybersecurity/comments/1q95y5p/ — layered defense compounding: 7% → 0.2% → 0.003% (2,300×)
- agatsoftware.com/blog/ai-agent-security-2026-google-forecast/ — Google 2026 AI threat forecast

You don't have to solve AI alignment. You have to build a boundary you can trust.

Read the full engineering breakdown — semantic proxy, subnet isolation, per-Actor identity — with implementation guidance and sources.

attribit-id.com/writing/treat-agents-like-untrusted-code