144 TO 1

Why Every Workload in Your Cloud Is Already Exposed

An Executive Perspective

AVIATRIX®

Five major open-source ecosystems compromised in thirty days. Not by one actor, by at least four, operating independently, across three continents. TeamPCP hit Trivy, Checkmarx KICS, LiteLLM, and Telnyx in a cascading supply chain operation that turned trusted security tools into credential harvesters. North Korea's UNC1069 backdoored the Axios npm package, one hundred million weekly downloads. Lapsus$ exfiltrated 2.66 gigabytes of AstraZeneca data using stolen credentials. The Vect ransomware group distributed affiliate keys to roughly three hundred thousand BreachForums members. The first confirmed ransomware deployment has already occurred.

This is not a spike. It is the new operational tempo. Software supply chain attacks more than doubled in 2025. Losses reached sixty billion dollars globally and are projected to hit eighty billion in 2026. Over seventy percent of enterprise cloud environments experienced at least one supply chain or third-party security incident last year.

Paper 1 in this series introduced the toxic combination. Three forces converging. Attackers industrializing. AI putting frontier offensive capability into millions of hands at consumer cost. Cloud insecure by default by design. This paper is about where the toxic combination is concentrating its damage. The bullseye is AI workloads in the cloud. And the structural reason the bullseye exists is the 144 to 1 ratio.

How the New Model Works

Consider LiteLLM, the AI gateway proxy used by thirty-six percent of enterprise cloud environments at the time of compromise. On March 24, TeamPCP published a malicious version to PyPI. The poisoned package planted a .pth file, a Python path configuration hook, that executes automatically on every Python process startup, regardless of whether LiteLLM is imported. Within three hours, approximately forty thousand environments had pulled the compromised version. The payload harvested environment variables, Kubernetes configurations, and cloud tokens across AWS, GCP, and Azure, then exfiltrated them to attacker-controlled infrastructure.

This was not a vulnerability that required exploitation. It was not a misconfiguration that required discovery. It was a trusted package, installed through a trusted channel, running through a trusted pipeline. The malicious code executed with the same permissions as every other workload in the environment. To the infrastructure, it was legitimate.

Now ask the question that matters: where in the traditional security stack would this have been stopped?

Your Zero Trust Network Access did not see it. No user authenticated to pull the package. Your cloud-delivered perimeter proxy did not inspect it. The traffic was workload-to-registry, not user-to-application. Your centralized inspection point may not have seen the exfiltration at all, because the compromised pod routes through a node NAT gateway, and that traffic never crosses the chokepoint. Your endpoint detection platform was not running on the container. Your identity governance platform verified the human who wrote the Dockerfile, not the workload that ran it.

The entire security stack, purpose-built to verify users and inspect user-initiated traffic, was architecturally blind to a compromised workload communicating with an attacker's command-and-control server. These tools are not failing. They are succeeding at a task that no longer covers the full threat surface. The question they were designed to answer, should this user be allowed access?, is not the question this attack posed. The question was: should this workload be allowed to reach this destination? And no tool in the stack was positioned to answer it.

The blast radius of the compromise was determined not by the sophistication of the payload, but by the communication paths available to the workload it ran on. In environments where the workload could reach any external IP over HTTPS, the exfiltration completed. In environments where workload communication was governed, the POST to the attacker's endpoint hit a deny rule and the credentials never left. Same payload. Same vulnerability. Fundamentally different outcomes, determined entirely by architecture.

Can this workload reach that destination?

That is the only question that would have mattered in the critical window between compromise and detection. It is a question about Communication Governance. And for most enterprises, the honest answer is: we do not know, because nobody governs that path.

144 to 1

The scale of this blind spot is not incremental. It is structural.

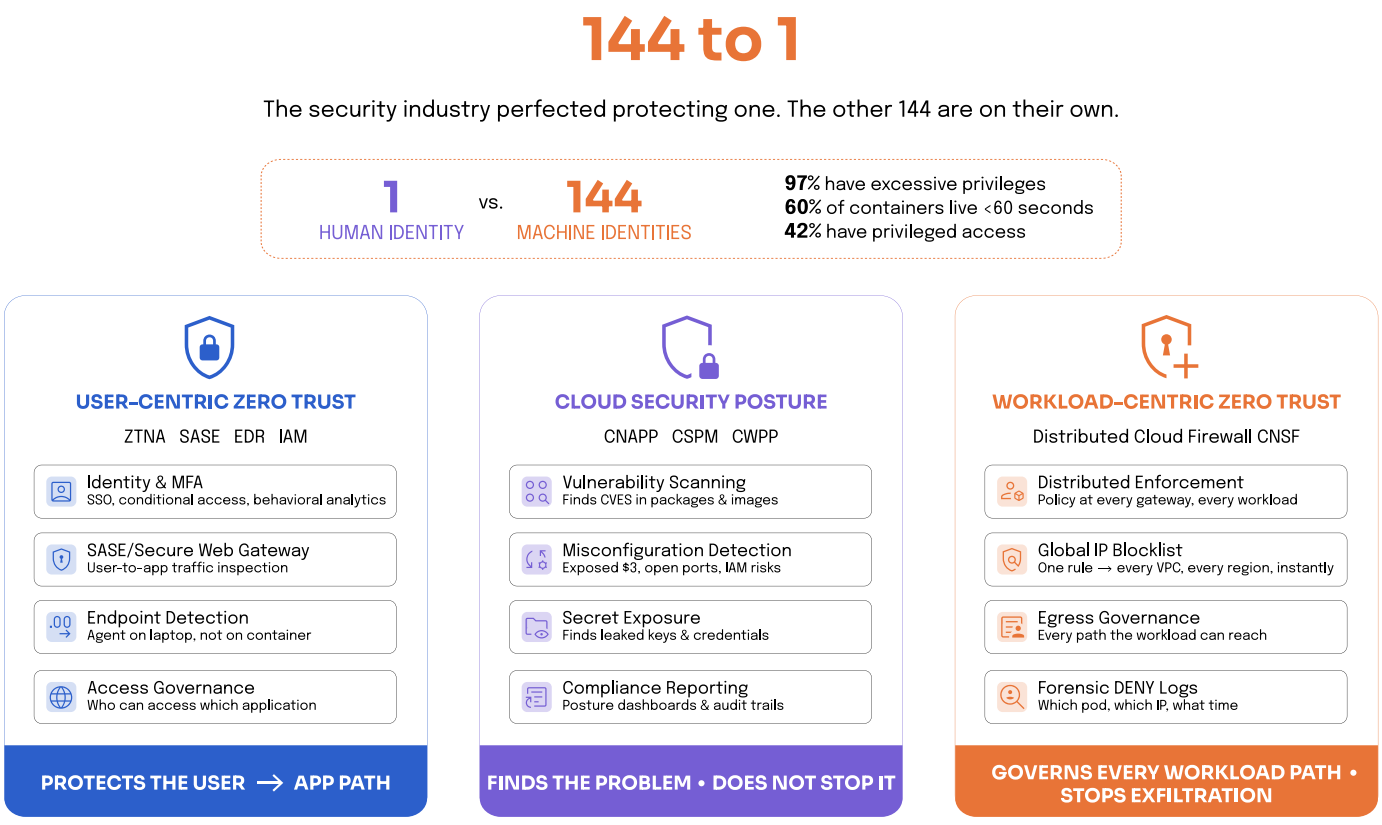

Figure 1: 144 to 1

In the average enterprise cloud environment, machine identities outnumber human identities by a factor of 144 to 1. That ratio has grown fifty-six percent in the last twelve months alone, and in advanced cloud-native environments, Sysdig reports it reaches as high as 40,000 to 1. Every one of your employees has a managed identity with single sign-on, multi-factor authentication, conditional access policies, behavioral analytics, and session management. Twenty-five years of tooling maturity protects the human side of your identity surface.

Now consider the other side. Forty-two percent of machine identities have privileged or sensitive access. Ninety-seven percent have excessive privileges. Seventy-one percent are not rotated within recommended timeframes. And here is the number that reframes the entire conversation: sixty percent of containers live for sixty seconds or less. They spin up, pull dependencies, execute, and terminate before your identity governance platform has completed its evaluation cycle. There is no MFA for a Kubernetes pod. There is no behavioral analytics baseline for a Lambda function that exists for three seconds. There is no conditional access policy for a workload that was created by an automated pipeline and destroyed before the next scan cycle ran.

For every identity you govern, there are 144 you do not. They are ephemeral. They are privileged. And they are exactly where the attack lands.

The ratio is accelerating, and AI is the accelerant. Every AI agent, every LLM gateway, every RAG pipeline, every autonomous coding assistant is a machine identity, ephemeral, privileged, and communicating outbound at unprecedented scale. Unlike a container that pulls a dependency and terminates, an AI agent makes decisions about what to reach next. In an enterprise deploying AI at scale, the 144-to-1 ratio is not static. It is compounding. And every new agent represents an ungoverned blast radius unless the communication paths available to it are governed by architecture.

Why AI Workloads Are the Bullseye

The 2026 breach data is unambiguous about where the toxic combination is concentrating its damage. Read the incident inventory and one pattern dominates.

LiteLLM was the headline of The Cascade. Three weeks later, the Bitwarden CLI compromise carried a payload that specifically enumerated the configuration directories of Claude, Cursor, Codex CLI, and Aider, treating developer AI assistants as a concentrated source of cloud and repository credentials. GrafanaGhost weaponized Grafana's AI assistant to silently exfiltrate data through an authorized rendering channel, with no anomalous signal for any detection tool to find. The Vercel breach moved through Context.AI, a third-party AI productivity tool that had itself been compromised via Lumma Stealer. Cursor saw three malicious npm packages compromise more than 3,200 developer workstations. Anthropic's MCP architecture had a "by design" RCE. A systemic MCP flaw exposed 200,000 servers and 150 million downloads of AI agent frameworks, IDE extensions, and automation tooling.

This is not coincidence. It is structure. AI workloads concentrate risk for three reasons.

They are ephemeral. A container may live for sixty seconds. A serverless inference function may live for three. An agent invocation may complete in milliseconds. There is no time to install an agent, baseline behavior, or run an identity governance evaluation cycle. The defender's traditional toolset cannot reach them.

They are highly privileged. AI agents need broad cross-service access to do their jobs. They call other services. They reach data stores. They communicate outbound at unprecedented scale. When a workload that holds an over-scoped non-human identity

is compromised, the attacker inherits its identity by construction, and that identity can often be exfiltrated and replayed elsewhere. Machine identity compromise is not adjacent to credential compromise. It is credential compromise.

They are rapidly shipped. Every enterprise is racing an AI roadmap. Every business unit is standing up RAG pipelines, agentic frameworks, MCP-connected tools, and inference endpoints. The maturity gap between deployment velocity and security review is the widest it has been in twenty years. Security teams are catching up to AI workloads from behind, and the workloads are still accelerating away.

The world is over-rotated on AI. The greatest concentration of risk is also AI. The architecture that defends AI workloads is the architecture that defends cloud workloads generally. Containment is not AI security. It is the architecture AI workloads happen to need most urgently, because they sit at the intersection of all three legs of the toxic combination.

Why User-Centric Zero Trust Cannot Be Extended to Workloads

The natural instinct is to say: we built Zero Trust for users, now we extend it to workloads. Apply the same principles and the problem is solved.

It is not that simple, and understanding why is the key to understanding the architectural gap.

User-Centric Zero Trust works because user identities are persistent, observable, and slow-moving. A human logs in once, maintains a session for hours, accesses a predictable set of applications, and behaves in patterns that can be baselined over weeks. The identity exists long enough to verify. The session exists long enough to inspect. The behavior exists long enough to analyze.

Workload identities have none of these properties. A container may live for sixty seconds. A serverless function may live for three. A Kubernetes pod may be created, execute, and terminate before any identity-aware control plane has registered its existence. These workloads do not authenticate through your perimeter gateway. They communicate workload-to-workload, workload-to-API, workload-to-registry, workload-to-internet, all through paths that were never instrumented for user-centric security.

The problem is not that user-centric tools are bad. They are excellent at what they do. The problem is that what they do is protect a surface area that represents less than one percent of the identity population in a modern cloud environment. Extending them to the other ninety-nine percent is not a roadmap item. It is a physics problem. The entity does not persist long enough for identity-based controls to operate on it.

This is why the defense model for workloads must be different. Instead of verifying the entity and then trusting the path, you must govern the path regardless of the entity. Because the entity might not exist long enough to be verified, but the path to the attacker's command-and-control server will exist for as long as you leave it open.

The $32 Billion Question the Market Got Half Right

In 2025, Google acquired Wiz for thirty-two billion dollars. The market signal was unambiguous: cloud workload security is a massive, unsolved problem. The market was right about the problem.

Wiz, and the broader CNAPP category it represents, answers a critical question: Do you have a problem? Posture management tools scan your cloud environment, find misconfigurations, flag vulnerable packages, and surface compliance gaps. This is valuable. What it does not do is stop a compromised workload from reaching an attacker's endpoint in the minutes before the scanner catches up.

Posture tells you what is wrong. Enforcement stops what is happening. These are different capabilities, operating at different points in the kill chain, answered by different architectures. Thirty-two billion dollars bought the best answer in the world to do you have a problem. It did not buy the answer to can the compromised workload reach its destination.

What One Enterprise Proved

One Aviatrix customer, a Fortune Global 500 company with forty-six billion dollars in revenue, five hundred eighty thousand employees, and operations across thirty countries, was running LiteLLM in their AI development environment when the TeamPCP compromise occurred.

The Aviatrix Threat Research Center published an advisory identifying four command-and-control IP addresses associated with the campaign: 46.151.182.203, 83.142.209.203, 83.142.209.11, 45.148.10.212. That last one is the confirmed LiteLLM credential exfiltration server. Their security team reached out to their cloud infrastructure team: can we block these? The answer was immediate. They already had an Aviatrix Cloud Native Security Fabric with a Global IP Blocklist policy in place. The infrastructure engineer added the four known command-and-control IP addresses from the Aviatrix Threat Research Center advisory to the existing rule.

The best practice today is even simpler: enable threat blocking in Distributed Cloud Firewall and known command-and-control infrastructure is blocked automatically, with no manual IP management required. The Aviatrix Threat Research Center continuously publishes threat intelligence that the firewall can consume directly. This customer acted on an advisory manually and stopped a live exfiltration. With threat blocking enabled, the block would have been automatic.

One policy update. It propagated to every gateway across their entire cloud footprint simultaneously, every VPC, every region, including the Kubernetes environments where LiteLLM was running. Within minutes, the DENY logs started appearing: compromised pods were attempting to reach the C2 endpoints, and every attempt was blocked. Zero credentials exfiltrated.

The detail that matters most is the honest one. This customer is not yet running default-deny Workload Containment with every workload mapped to an explicit allowlist. They are building toward that posture. What they had was a partial implementation of the architecture, a propagating blocklist policy with no agent on the workload and no human in the response loop. That partial implementation stopped a live supply chain exfiltration in production. The infrastructure engineer did not even know what those IP addresses were connected to at the time of the block. He received the indicators, applied them to a framework that was already operational, and the architecture handled the rest.

And critically: this block could not have happened at a centralized inspection point. LiteLLM is a Kubernetes workload. When a compromised pod exfiltrates credentials, that traffic routes through the node's NAT gateway. The communication path never crosses the chokepoint. In a modern cloud environment, these are not edge cases. They are the majority of the traffic.

This enterprise is building toward where every security team wants to be: workloads fully mapped, an explicit allow list in place, and everything else denied by default. They are not there yet. But they did not need to be. A blocklist policy running in production, one piece of a broader enforcement architecture, was enough to stop a live supply chain exfiltration. That is the leverage of getting the architecture right early.

Every DENY entry became a forensic record: which pod, which IP, what time. The response was measured, targeted, and complete, because the architecture provided both the enforcement and the evidence.

This is what containment architecture looks like in practice. Not a framework. Not a maturity model. Not a scanning dashboard. A policy that governs every path, applied once, enforced everywhere, with a forensic record of everything it stopped.

The Question Every CISO Should Be Asking

The security industry built a two-hundred-billion-dollar architecture around protecting users. That architecture is necessary. It is not sufficient. For every human identity it governs, there are 144 machine identities it does not, ephemeral, privileged, and running on paths that bypass every user-centric control. AI is accelerating that ratio every quarter.

The question that matters now is: Does our enforcement govern every path a workload can take to any destination?

If the answer requires a qualification, except for Kubernetes pods, except for serverless functions, except for east-west traffic, except for the regions where policy has not propagated yet, then the answer, functionally, is no. And the gap between that answer and the operational reality of Q1 2026 is where the next breach will occur.

Zero Trust was the right idea. We built it for the wrong surface first. It is time to build it for the one that is actually under attack.

This is Paper 3 of 4 in the Containment Era series.

Paper 1: The Containment Era – Why the Threat Model Outgrew the Architecture.

Paper 2: The Containment Platform – How Cloud Native Security Fabric Closes the Architectural Divide.

Paper 4: The Priority Inversion – Why the SANS Mythos Report Has the Order Wrong.

Aviatrix

Aviatrix® is pioneering the Cloud Native Security Fabric (CNSF) – the architecture the Containment Era requires. CNSF governs every workload communication path across every cloud, every VPC, every Kubernetes cluster, and every serverless function, from a single policy plane. One rule. Universal propagation. Enforced at the workload, not at a chokepoint. Trusted by more than 500 of the world's leading enterprises. For more information, visit aviatrix.ai.