THE CONTAINMENT ERA

Why the Threat Model Outgrew the Architecture

An Executive Perspective

AVIATRIX®

Executive Summary

Your security architecture was built for a threat model that no longer exists. The threat model changed. The architecture did not. This paper explains why that gap is now structural, what it means for every organization operating in the cloud, and what the architecture that closes it must look like.

Three forces are converging, and each one accelerates the others. Attackers are industrializing, with The Cascade serving as the visible signature of a criminal economy that has matured to industrial scale. AI is putting frontier offensive capability into millions of hands at consumer cost, on a curve that does not slow down. And cloud is insecure by default, by design, because the providers optimized for developer velocity. We call this the Toxic Combination, and the bullseye it is producing is AI workloads in the cloud.

For fifteen years, the cybersecurity industry invested over two hundred billion dollars building an apparatus designed to answer one question: can we detect the attack before it causes damage? That question assumed the attack would be visible. In March 2026, a supply chain operation attributed to TeamPCP—called The Cascade—proved otherwise. Five software ecosystems compromised in twelve days. Credentials harvested from trusted packages, through trusted channels, using trusted update mechanisms. The detection stack saw nothing anomalous. Because trusted code executing in a trusted pipeline is not an anomaly. It is Tuesday.

This is not a temporary failure. It is a structural condition. The threat model has fundamentally outgrown the security architecture that most enterprises rely on. We call the underlying gap the Architectural Divide: the widening distance between the speed at which organizations deploy cloud workloads and the ability of their security architecture to enforce consistent policy across them. It manifests as three operational gaps that compound one another. And it demands a response that is architectural, not incremental.

The industry has entered a new era. We call it the Containment Era. Detection remains necessary. It is no longer sufficient. The primary metric is no longer how fast you find the attacker. It is how far the attacker can reach when your detection misses. That is a blast radius question. And blast radius is determined by architecture.

The Pattern That Built for Six Years

The Cascade did not emerge from nothing. It was the culmination of an attack pattern that has been building since 2020, each incident moving deeper into the foundations of trust itself.

SolarWinds (2020) proved the concept: a nation-state actor rode a legitimate software update into eighteen thousand organizations. The attack executed as trusted code, delivered through a trusted channel, signed with a trusted certificate. Detection had no signal to find. Log4j (2021) expanded the surface: a vulnerability in a ubiquitous open-source dependency exposed virtually every Java application on earth. The trust surface was not just software updates. It was every dependency your workloads inherit. 3CX (2023) showed cascading compromise: a supply chain attack on a VoIP client originated from a prior supply chain compromise of a financial platform. For the first time, one compromised vendor infected another. XZ Utils (2024) proved the adversary manufactures trust itself: a multi-year social engineering campaign to gain maintainer access to a foundational compression library used in SSH authentication across virtually every Linux distribution. Caught by accident, not by detection.

And then, in Q1 2026, the pattern that had been building across four major incidents became the default operating model for multiple threat actor categories simultaneously.

The Cascade and the Fork

Something changed in March 2026. Not the tactics. Not the tools. The structure.

The Cascade was not a single breach. It was an industrialized credential harvesting operation that compromised five major software ecosystems in twelve days, formed monetization partnerships with ransomware groups, mobilized over 300,000 dark web actors as an affiliate distribution layer, and began deploying ransomware against affected organizations. Five ecosystems in twelve days is not the work of a small cell. It is the output of a value chain operating with the throughput of a Fortune 500 supply chain. Simultaneously, North Korea's UNC1069 compromised the Axios npm package with a cross-platform remote access trojan. Salt Typhoon expanded operations to 200+ companies across 80+ countries. Storm-0501 pivoted from endpoint ransomware to cloud-native attack chains. Four independent operations. Similar tradecraft. The same window.

For a growing class of attacks, the detection era's foundational assumption no longer holds: that malicious activity looks different from legitimate activity. When the attack IS the trusted activity, detection cannot distinguish signal from noise.

The credential harvesting executed in a single run. The output was encrypted before touching the network. The exfiltration was an HTTPS POST indistinguishable from legitimate API traffic. For this class of attack, the entire detection apparatus was positioned in the wrong layer. Not because those tools failed at what they do. Detection continues to catch the majority of threats enterprises face: phishing, stolen credentials, misconfigured resources, exploitation of known vulnerabilities. But the Cascade represented something structurally different. The adversary moved to a layer those tools were never designed to govern.

The right question is no longer: Is something bad happening? The right question is: If something bad is already running, what can it reach, and what can it send? That question has an architectural answer, not a detection answer. And the architecture that answers it is the communication topology of the workload environment.

We call this inflection point the Fork. Not because one path is wrong and the other is right. Both paths include detection. The Fork is about what else you build. The organization that invests only in detection is betting that every future attack will be detectable. The pattern from SolarWinds to The Cascade proves that bet is structurally unsound.

The Toxic Combination

The Cascade and the Fork are visible. The forces that produced them are structural. Three of them are converging, and each one accelerates the others.

Attackers are industrializing

The Cascade is the visible signature of a criminal economy that has matured to industrial scale. Specialization of roles. Affiliate networks. Mutual-benefit contracts between initial access brokers, credential harvesters, ransomware operators, and money launderers. TeamPCP did not just compromise five ecosystems in twelve days. It demonstrated a value chain that can be operated at the throughput of legitimate industry, with 300,000 affiliates as a distribution layer and ransomware groups as the monetization engine. The pattern that has been building since SolarWinds is no longer an emerging tradecraft. It is the default operating model for multiple sophisticated threat actor categories simultaneously.

AI puts frontier offensive capability into millions of hands

In April 2026, Anthropic announced Project Glasswing, a $100 million emergency coalition with twelve major technology companies. The coalition was formed because Anthropic's Claude Mythos model, during controlled safety testing, autonomously discovered thousands of exploitable zero-day vulnerabilities across every major operating system and browser. Mythos weaponized a 17-year-old FreeBSD kernel vulnerability in approximately four hours with no human guidance. The same week, China's Z.ai released GLM-5.1, a 754 billion parameter model with comparable capability, under MIT license with no safety constraints.

Mythos is one data point on a curve, not the curve itself. The structural truth is that offensive capability is an emergent property of general AI capability improvement. Every frontier model going forward will have it. Every open-weight model will inherit it within months. Open weights cannot be recalled.

What this changes is not just capability. It is cost. Six months of skilled human labor becomes a few hours of compute. In any system where attacker cost collapses, attack volume scales nonlinearly. A single attacker now has the leverage to run hundreds of campaigns in parallel. The population of actors capable of exploiting zero-days is expanding by orders of magnitude, and the cost per exploit is collapsing toward consumer compute. AI does not create a new attack surface. It guarantees the existing one will be tested by far more hands, far more cheaply, far faster.

Cloud is insecure by default, and that is by design

The major cloud providers ship permissive defaults because friction kills developer velocity, and developers are their primary buyer. AWS security groups allow all outbound traffic. GCP has an implied allow-all egress rule, invisible in the firewall rules list. Azure NSGs default to AllowVNetInBound. AKS, EKS, and GKE all default to unrestricted pod-to-pod communication across the entire cluster. Gartner estimates that only 5 to 20% of enterprises have implemented microsegmentation in any form.

The shared responsibility model placed interior security on the customer. The customer did not accept that responsibility at scale, because the tooling to enforce it did not exist in a form that matched cloud's shape. The result is a tangled web of interconnected workloads, most of them with direct internet egress and zero inspection. Most enterprises cannot map the communication graph of their own cloud, let alone govern it.

This is the leg of the stool the defender controls. You cannot stop attackers from industrializing. You cannot recall AI capabilities. You can change your cloud's default posture. Containment is how.

Why each leg makes the others worse

The three forces are not additive. They are multiplicative. AI lowers attacker cost, which lets more actors enter, which is the precondition for industrialization. Industrialization specializes in cloud-native tradecraft, which targets the insecure-by-default surface. The insecure-by-default surface is so large that even untargeted attacks at AI economics find paths. Each leg accelerates the others. That is what makes it toxic.

The Architectural Divide

The toxic combination exposed a structural problem that predates March 2026. The threat model has been outgrowing the architecture for a decade.

The Architectural Divide is the growing, structural gap between the rate at which organizations deploy cloud workloads and the ability of their security architecture to enforce consistent, real-time policy across them.

Over the past decade, two vectors have been diverging at an accelerating rate. Workload proliferation has multiplied the number of workload types from virtual machines to containers, serverless functions, managed services, and now reasoning AI agents. Infrastructure fragmentation has expanded environments across multi-cloud, hybrid, edge, and sovereign deployments. The result is not additive complexity. It is multiplicative. And the security architectures most enterprises rely on were designed before either vector began to move.

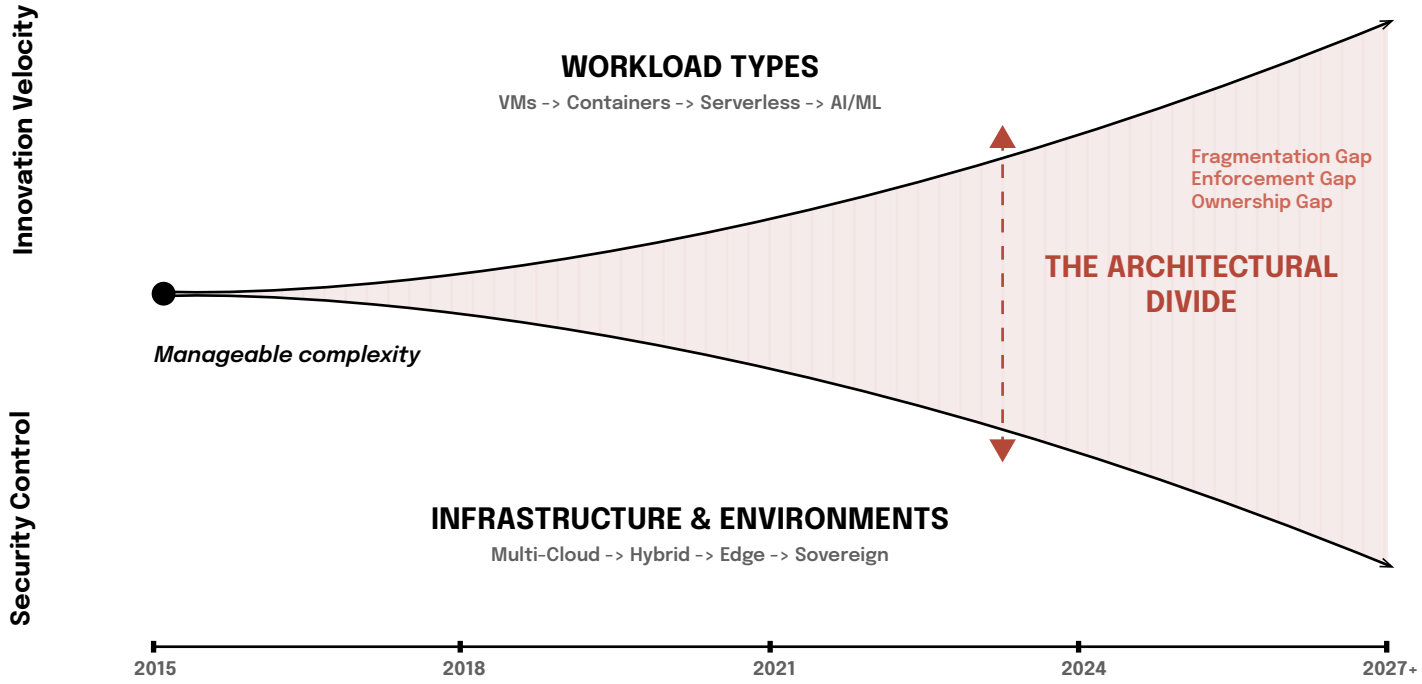

Figure 1: The Diverging Vectors – The Architectural Divide widens as workload innovation and infrastructure fragmentation accelerate on opposing trajectories.

It is not additive complexity. It is multiplicative.
And the multiplier is growing on both axes simultaneously.

Each cloud provider offers its own networking constructs, its own security controls, its own identity and access management. The same security intent must be expressed differently in every environment. When policy must be manually re-expressed in every environment, inconsistency is not a risk. It is a certainty.

The Fragmentation Gap

The average enterprise now manages 76 distinct security tools. Each was acquired to solve a real problem. Collectively, they create an environment where no single team has a unified view of security posture. The growing volume of assets deployed by development teams, business units, and employees creates an expanding attack surface that agent-based security models cannot cover. You cannot secure what you cannot see, and you cannot install an agent on what you do not know exists.

The Runtime Enforcement Gap

The cloud security industry has made enormous progress in visibility. Posture management tools scan environments. Vulnerability scanners catalog exposures. But there is a critical difference between seeing a risk and stopping a threat. Most visibility tools operate in an advisory capacity. They alert. They report. They create tickets. What they do not do is sit in the data path and enforce policy in real time.

You cannot stop what you cannot enforce. Visibility without inline enforcement is a monitoring strategy, not a security architecture.

The CrowdStrike 2026 Global Threat Report found that the average eCrime breakout time is now 29 minutes, with the fastest observed at 27 seconds. Meanwhile, organizations take an average of 241 days to identify and contain a breach. That asymmetry is the Runtime Enforcement Gap expressed as operational risk. And it is compounded by a credential vector that detection cannot reach. 82% of intrusions in 2026 ride valid credentials through legitimate channels, producing no anomalous signal for any tool to find.

The Ownership Gap

In the cloud operating model, the teams that deploy infrastructure are often not the teams that secure it. Platform engineering builds the landing zones. Application teams deploy workloads. Security teams define policy. Cloud operations manages the network. When an incident occurs, the first thirty minutes are spent determining which team owns the affected workload, which tool has the relevant telemetry, and which console to use to respond.

Why Chokepoint Security Fails

The dominant response to the Architectural Divide has been to extend what worked in the data center: deploy an inspection appliance in a transit VPC, route traffic through it, and inspect at the chokepoint. This is Chokepoint Security. It is a device in a traffic path that governs what happens to route through it.

The problem is that cloud networking does not work like data center networking. In cloud, workloads reach the internet through multiple paths that never traverse the centralized inspection point.

Kubernetes pod egress

A compromised pod reaches the internet through the node's NAT gateway. It never traverses the transit inspection point. Exfiltration completes.

Serverless functions

Lambda, Cloud Functions, Azure Functions egress through the provider's native NAT. They do not traverse customer inspection infrastructure.

East-west traffic between VPCs

Traffic moves directly between spokes via VPC peering or transit gateway. Lateral movement between cloud segments travels a path the centralized inspection point does not see.

Hidden communication paths

PrivateLink endpoints, VPC peering to island VPCs, and load balancers with public IPs all create outbound data paths that bypass the transit inspection point entirely. These are standard cloud networking constructs.

A chokepoint governs the traffic that routes through it. Communication Governance means every workload, in every VPC, on every path to the internet, is subject to policy. One is a device. The other is an architecture.

The operational burden of the chokepoint model is not theoretical. The industry's leading next-generation firewall vendor publishes a reference architecture for securing a single cloud provider–one cloud, one set of workloads. The document runs to more than sixty pages. It requires five to seven distinct products, each with its own configuration surface, licensing model, and failure domain, all converging on a single centralized inspection point. This is not an implementation shortcut or a poorly designed deployment. It is the vendor's own best practice–the canonical answer for how to make a chokepoint work in cloud. The complexity is structural. It is the necessary consequence of forcing distributed workloads through a centralized enforcement model. And it governs only the traffic that routes through it. Every path documented in the sections above–pod egress, serverless functions, east-west peering, hidden communication paths–remains ungoverned.

The Three Eras of Cybersecurity

The history of cybersecurity is the history of where you place the enforcement point. Each era answered that question differently. Each era's answer held until the traffic pattern changed underneath it.

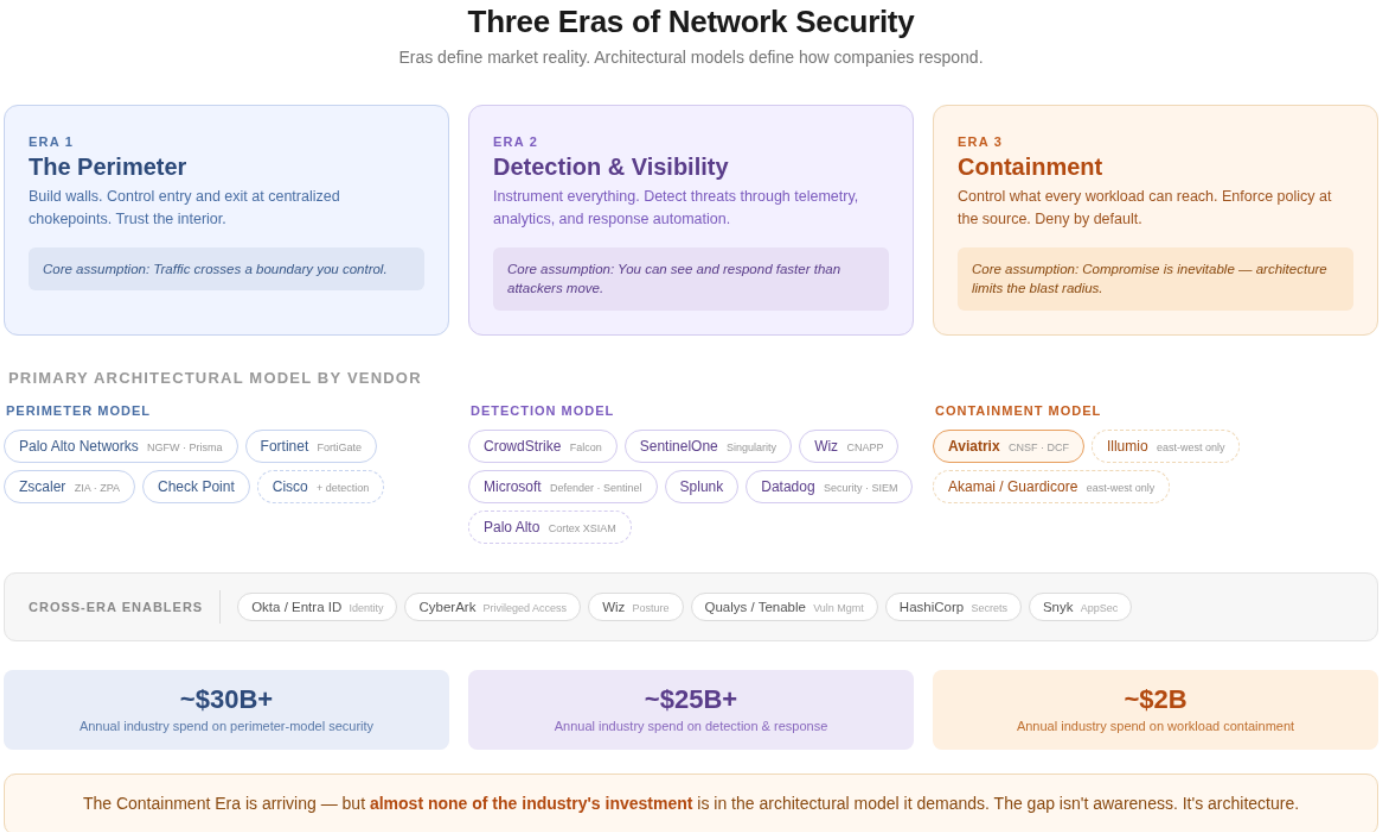

Figure 2: The Three Eras of Cybersecurity – Eras define market reality. Architectural models define how companies respond.

Era 1: The Perimeter Era (1995–2010)

The enforcement point was the network edge. The operating assumption: the threat comes from outside. It held for fifteen years because the computing model it protected was stationary. Cloud computing dissolved the perimeter. You cannot enforce at a boundary that no longer exists.

Era 2: The Detection Era (2010–2026)

The enforcement point moved inside. Since you could no longer keep threats out, the thesis became: find them fast once they are in. SIEM, EDR, SOAR, XDR. It worked because attacks looked different from legitimate activity. What is ending it: a growing class of attacks that move through trust, not around defenses. 82% of detections are now malware-free. The average breakout time is 29 minutes. The fastest was 27 seconds. AI collapses the detection-response cycle below human response thresholds.

Era 3: The Containment Era (2026–)

The enforcement point moves to every workload. Since you can no longer reliably detect compromised code before it executes, the thesis becomes: limit what it can do when it runs. Communication Governance that ensures no workload can reach anything it was not explicitly permitted to reach. The operating assumption: something in your environment is compromised right now. Detection may never flag it. The question is not whether you detect it. The question is what the blast radius is when it runs.

Containment does not replace detection. It is the precondition that makes detection effective. By bounding the blast radius first, containment narrows the search space so that detection tools can identify and eliminate the threat within a governed space.

What Containment Architecture Requires

The Architectural Divide tells us what the consolidation architecture must deliver. These are architectural requirements dictated by the structure of the problem. Paper 2 in this series details how the Cloud Native Security Fabric meets each one.

Paper 2 in this series provides the formal definition of containment and the five testable properties–path-complete, identity-aware at L7, detection-independent, compute-model agnostic, and universally propagated–that any containment architecture must demonstrate. That definition anchors every architectural requirement that follows.

Embedded in the cloud fabric.
Enforcement must operate within the cloud network itself, not bolted on. Policy must live where traffic flows.

Real-time inline enforcement.
The architecture must sit in the data path and enforce policy in real time, before lateral movement occurs, before data leaves the environment.

Unified policy across all constructs.
A single security intent must be expressible once and enforced everywhere, across cloud providers, across workload types, across network segments.

Cloud-native and workload-agnostic.
The architecture cannot depend on installing agents. It must secure virtual machines, containers, serverless functions, and managed services with equal effectiveness.

Communication Governance as the primary metric.
The organizing question is not did we detect the attack but for every workload in our environment, what can it communicate with, and what is it permitted to send to the internet?

Detection tells you what happened. Containment limits what could happen. Both matter. Only one governs blast radius.

Architecture in Practice: The LiteLLM Proof Point

The requirements above are not theoretical. One Fortune Global 500 company–$46 billion in revenue, 580,000 employees, operations across 30 countries–proved them in production during the LiteLLM supply chain compromise described in this paper.

When the TeamPCP campaign was identified, their security team flagged four command-and-control IP addresses. An infrastructure engineer added them to the Aviatrix Distributed Cloud Firewall Global IP Blocklist. 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, DENY logs confirmed: compromised pods were attempting to reach the C2 endpoints, and every attempt was blocked. Zero credentials exfiltrated.

The detail that matters: the engineer did not know what those IP addresses were connected to at the time. He applied indicators to a framework already in place. The architecture handled the rest. And critically, this block could not have happened at a centralized inspection point. LiteLLM runs as a Kubernetes workload. When a compromised pod exfiltrates credentials, the traffic routes through the node's NAT gateway–a path the chokepoint never sees. Paper 3 in this series documents the full case study.

Why AI Workloads Are the Bullseye

The toxic combination produces a bullseye. The 2026 breach data is unambiguous about where it is.

LiteLLM, the AI gateway proxy used in 36% of enterprise cloud environments, was the headline of The Cascade. 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. The Vercel breach moved through Context.AI. 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.

This is not coincidence. It is structure. AI workloads concentrate risk for three reasons. They are ephemeral, defeating agent-based defense. They are highly privileged, because they need broad cross-service access to do their jobs. They are rapidly shipped, because every enterprise is racing an AI roadmap and security maturity has not caught up. The world is over-rotated on AI. The greatest concentration of risk is also AI. Paper 3 in this series quantifies why.

The Category Landscape

The Containment Era does not render existing security categories obsolete. Each delivers genuine value. The question each organization must answer is whether its architecture includes the category that governs blast radius.

Category	Primary Model	Job	What It Does Not Govern
Hybrid Mesh Firewall	Perimeter	Enforce at network edge	Traffic paths that bypass the inspection point
Cloud-Delivered Perimeter (SASE / SSE)	Cloud-Relocated Perimeter	Secure user-to-application access	Machine-to-machine communication
CNAPP / Posture Management	Detection	Detect misconfigurations and vulnerabilities	Runtime enforcement
Endpoint / XDR	Detection	Detect and respond on endpoints	Cloud-native workloads where agents cannot run
Identity / CIEM	Detection / Governance	Govern identity lifecycle and privilege	Communication paths between workloads
Containment Platform	**Containment**	**Govern blast radius at every workload**	**Requires deployment at workload layer; complements detection and posture**

One category conspicuously absent from this table deserves attention: AI Workload Security. AI systems, including agentic frameworks, MCP-connected tools, and inference endpoints, represent the fastest-growing and least-secured workload category in the enterprise. This is not an emerging category that requires a new product. It is a workload class that requires containment architecture. Paper 3 in this series quantifies why.

Containment is the only category that holds equally against all three legs of the toxic combination. It does not care whether the workload was compromised through an industrialized supply chain attack, an AI-discovered zero-day, or a phished credential. It cares what the workload can reach.

The Question Before You

The Architectural Divide is not a future risk. It is a present reality that grows wider with every new cloud deployment, every new workload type, and every new environment added to the enterprise portfolio.

Three questions can clarify where you stand:

Can you express a single security policy once and have it enforced consistently across every cloud environment, workload type, and network segment you operate?

When a workload is compromised, can your architecture prevent lateral movement in real time, before the alert reaches a human console?

If a valid credential is used against one of your AI workloads at 3:00 AM on a Sunday, what, architecturally, prevents it from reaching the ledger?

If the answer to any of these is no, the Architectural Divide is present in your organization. That is not an indictment. It is the reality of nearly every enterprise operating at scale in the cloud today. But it is a reality that demands a deliberate architectural response.

The Containment Era has begun.

This is Paper 1 of 4 in the Containment Era series.

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

Paper 3: 144 to 1 – Why Every Workload in Your Cloud Is Already Exposed.

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.