www.pipelinepub.com
Volume 22, Issue 10

Runtime Security for AI-Driven Networks

By: Vishnu Gatla

Runtime has become the place where the real risk of AI-driven networks appears. Strategy documents, policy frameworks, and architecture diagrams are useful, but they do not decide whether an autonomous workflow should call an application programming interface, change a route, open a ticket, retrieve customer data, or trigger a production action. Those decisions happen in motion, inside live systems, under operational pressure. For telecom providers and enterprise infrastructure leaders, the security question is shifting from whether AI can improve operations to whether the runtime environment can prove what AI-assisted systems are doing, why they are doing it, and how quickly unsafe behavior can be stopped.

This shift matters because AI is entering networks through ordinary operational paths. It is not always introduced as a large standalone platform with a new security boundary. It can appear inside service assurance workflows, application delivery operations, incident triage, testing pipelines, customer support tools, cloud automation, API gateways, and data analysis processes. In many cases, the AI component is not directly exposed to the internet. That can create a false sense of safety. The higher risk is that the system inherits trusted access to tools, data and workflows that already exist.

Traditional network security models were built around human users, defined applications and traffic patterns that changed at a manageable pace. Those controls still matter, but they are incomplete when automation can reason across systems and act at machine speed. A human analyst may review an alert, check context, create a change request and wait for approval. An AI-assisted workflow may compress those steps into seconds. That speed can reduce operational friction, but it also reduces the time available to detect mistakes, abuse, excessive access, or unintended escalation.

The runtime problem is not simply that AI systems might make incorrect recommendations. The more serious concern is that a recommendation becomes an action without enough verification. A model that summarizes an outage is one level of risk. A model that can query customer-impacting APIs, modify configurations, adjust routing, suppress alerts, or initiate remediation is another. Once tool access is granted, the boundary between assistance and operation becomes thin. Security teams

need controls that understand identity, intent, authorization, behavior and recovery at the moment an action is attempted.

API security is central to this issue. Modern digital infrastructure depends on APIs for provisioning, telemetry, orchestration, billing, identity, service management and customer experience. AI-driven workflows will use the same APIs because they are the most efficient way to interact with production systems. That means API authorization can no longer be treated only as an application development concern. It becomes a runtime governance control for network and service operations.

The OWASP API Security Top 10 highlights risks such as broken object-level authorization, broken authentication, unrestricted resource consumption, and unsafe consumption of APIs. These risks become more consequential when the caller is automated and persistent. An API that was safe enough for a narrow human workflow may be unsafe when invoked repeatedly by an agent with broad access. Rate limits, schema validation, and authentication are necessary, but they are not sufficient. Runtime controls also need to verify whether the caller is allowed to perform the specific action on the specific object in the specific context.

Identity also needs to be reconsidered. Many enterprises have spent years improving user identity, multifactor authentication and privileged access management for human administrators. AI-driven networks require the same level of discipline for nonhuman actors. Each automated workflow should have a distinct identity, a defined purpose, scoped permissions and a clear owner. Shared service accounts, broad tokens and long-lived credentials are weak foundations for autonomous operations. They make it difficult to prove who initiated an action, what system authorized it, and whether the behavior was consistent with policy.

The next layer is intent binding. Runtime systems should not only ask whether a credential is valid. They should ask whether the requested action aligns with an approved task. For example, a workflow investigating packet loss may need read access to telemetry, topology, and recent change records. It should not automatically receive the ability to export sensitive data, disable monitoring, or modify a production configuration. Separating observation from execution is one of the simplest ways to reduce blast radius.

This is where approval gates remain important. The goal is not to slow every AI-assisted process until it becomes useless. The goal is to classify actions by risk. Low-risk queries and enrichment steps can often proceed automatically. Medium-risk recommendations may require human review. High-risk changes should require explicit approval, strong logging, and a rollback plan. In mature environments, this classification should be policy-driven rather than improvised by each team. Runtime visibility must also become richer. Conventional logs may show that an API call occurred, but not whether it was part of an approved playbook, which prompt or task led to it, which data was available to the system, and which policy decision allowed it. For AI-driven operations, auditability requires a chain of evidence. Teams need to connect the initiating request, model-assisted reasoning, tool call, authorization decision, system response, and operational outcome. Without that chain, incident response becomes guesswork.

Detection must evolve in the same direction. Signature-based controls and static allow lists are useful, but AI-driven traffic often looks legitimate because it uses valid credentials and approved interfaces. The suspicious signal may be behavioral. A workflow may call an unusual sequence of APIs, access a new class of object, repeat a request at abnormal frequency, combine data from unrelated systems, or attempt an action outside its normal time window. Runtime security should watch for these deviations and route them to the right control point before they become incidents.

Bot and automation detection also need a more balanced role. Not all automation is hostile. Telecom

and enterprise environments depend on automated testing, service assurance, provisioning, and customer-facing digital workflows. Blocking automation broadly would harm reliability and business operations. The better approach is to distinguish known, owned and policy-compliant automation from unknown or abusive automation. That requires inventory, behavior baselines, telemetry correlation and continuous review of what automated systems are allowed to do.

Operational resilience is the other half of runtime security

A control that only blocks unsafe activity is helpful, but production environments also need safe recovery. If an AI-assisted workflow attempts an unauthorized action, the system should preserve evidence, stop the unsafe step, notify the right owner, and continue with a safer path when possible. If a change is approved but produces unexpected effects, teams need rollback procedures that are tested before they are needed. In AI-driven networks, resilience means being able to contain both malicious abuse and well-intentioned automation errors.

The 2026 Data Breach Investigations Report notes that software vulnerabilities have become a leading breach entry point, and that generative AI is influencing multiple attack techniques. The lesson for infrastructure leaders is not that AI should be avoided. The lesson is that software, APIs, and automation have become central to the attack surface. Runtime security must therefore be designed around the systems that actually execute actions, not only around the humans who initiate work.

Governance frameworks can help, but only when they are translated into operational controls. The NIST AI Risk Management Framework emphasizes managing AI risk across design, development, use, and evaluation. CISA's Secure by Design guidance reinforces the need to shift security responsibility earlier into system design and to build safer defaults. For network and service operations, those ideas should translate into practical questions. What can this AI-assisted workflow access? Which actions are read-only? Which actions require approval? What evidence is retained? Who owns the control? What happens when the system is wrong?

There is also a business balance to maintain. Overly restrictive controls can prevent useful automation and push teams toward shadow tools. Overly permissive controls can create invisible operational risk. The right model is neither blind trust nor blanket denial. It is governed autonomy. AI-assisted systems should be allowed to improve speed, scale and decision support, but only within boundaries that are visible, testable and reversible.

For telecom and enterprise leaders, the practical starting point is to treat AI-driven runtime activity as a production control plane issue. That means creating an inventory of automated actors, mapping their API permissions, separating read access from change authority, enforcing least privilege, logging decisions as well as actions, and testing failure modes. These steps are not glamorous, but they are the difference between automation that can be governed and automation that becomes another unmanaged dependency.

AI-driven networks will not be secured by policy statements alone. They will be secured by runtime controls that can see, decide, limit, log, and recover while systems are operating. The organizations that get this right will be able to use automation more confidently because they will understand where authority begins, where it ends, and how to intervene when the system crosses a line.