

How Autonomous Agents Expand Enterprise Security Exposure

How autonomous AI agents expand enterprise exposure, and what security teams must do to test and defend



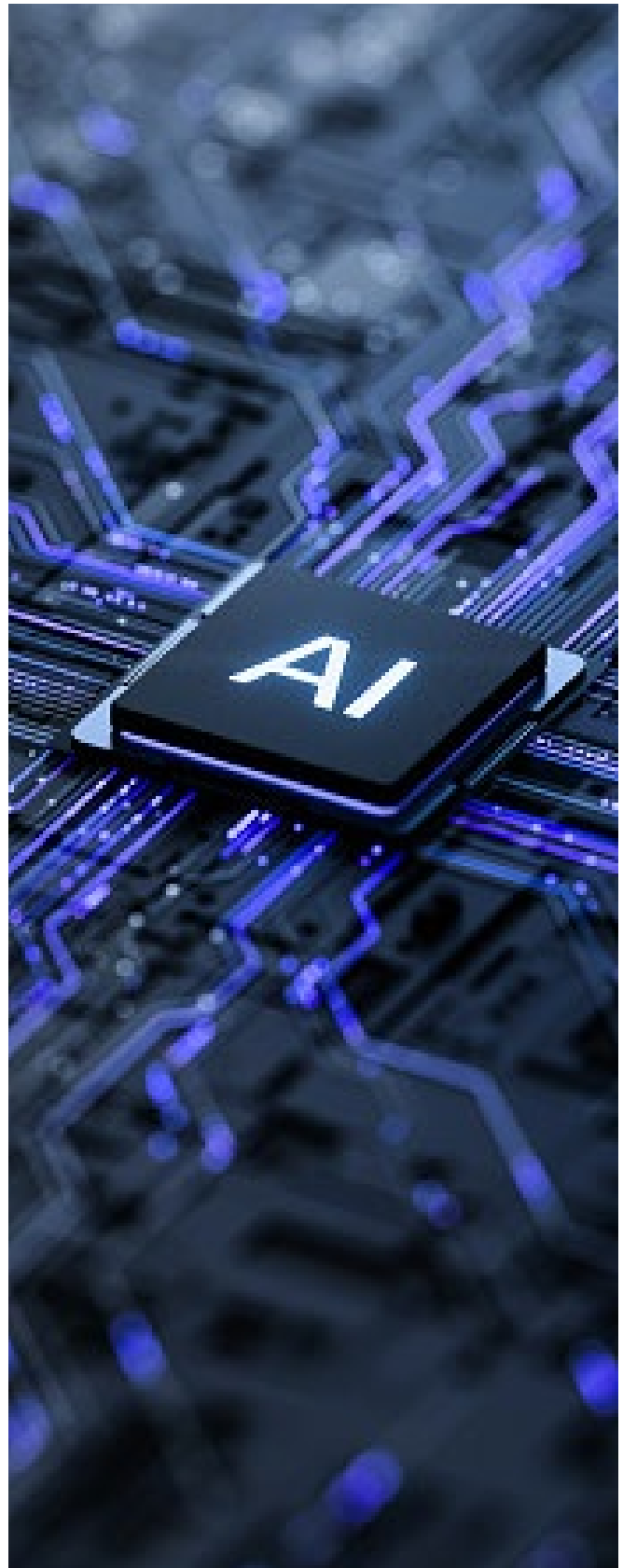
Table of Contents

Executive Summary	3
AI Has Shifted from Answers to Actions	4
How Agentic AI Changes the Attack Surface	5
Layer One: The Prompt and Response Layer	6
Layer Two: The Model Context Layer	7
Layer Three: The Tools and Actions Layer	8
Multi-agent Environments: Where Risk Compounds	9
Mapping Agentic AI Risk to Industry Frameworks	11
Six Requirements Every Agentic AI Security Strategy Needs	12
Securing the Agentic Attack Surface with TrojAI by A10 Networks	13
Conclusion	16
About A10 Networks	17

Executive Summary

AI applications have moved from answering questions to acting. An AI agent reasons over instructions, retrieves context from memory and data, and calls tools that write to systems that can send emails, modify code, and execute financial transactions. Every one of those capabilities opens a new attack path. This results in an attack surface that spans distinct layers that traditional applications and network security controls were never designed to inspect; one that grows with every new agent, model, connector, and tool an organization adopts.

The paper walks through the prompt and response layer, the model context layer, the tools and the actions layer, naming the threats that live at each one and showing how they chain into a single breach. It then covers what multi-agent environments and agent sprawl add to the problem, how the surface maps to OWASP, MITRE ATLAS, and NIST AI Risk management framework, and what a control-set purpose-built for AI has to deliver at both build time and run time. The closing sections describe how TrojAI by A10 Networks addresses each layer through automated red teaming and runtime defense.



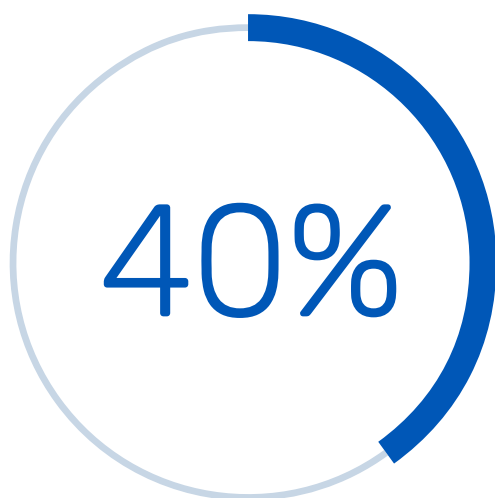
AI Has Shifted from Answers to Actions

Enterprise AI adoption has transitioned through three stages in rapid succession. The first was standalone model interaction, where employees used a chatbot in a browser. The second embedded large language models (LLMs) into enterprise AI applications which led to a support portal or an internal knowledge base article that could be leveraged to answer in natural language. The third stage is giving AI agents autonomy. An agent receives a goal, decides a plan, retrieves the context it needs, and invokes tools to carry the plan out with limited or no human review of each step.

The scale of the shift is significant. Gartner projects that 40 percent of enterprise applications will feature task-specific AI agents by the end of 2026, up from less than 5 percent in 2025. A [Cloud Security Alliance and Aembit survey](#) of more than 200 IT and security professionals conducted in January 2026 found that 85 percent already have AI agents running in production and 73 percent expect agents to be vital to operations within a year.

Governance has not kept pace with deployment. In the same survey 88 percent of organizations said they cannot distinguish AI agent activity from human activity from their logs, 74 percent said agents receive excessive access, 79 percent said agents create pathways that are difficult to monitor, and only 22 percent apply access frameworks to agents very consistently. Ownership is fragmented with security accountable in only 28 percent of organizations, engineering in 21 percent, IT in 19 percent, and identity teams in 9 percent.

Those two facts together define the problem. Agents are already in production, holding broad privilege, and operating largely outside the visibility that security teams rely on for every other class of workload.



Gartner projects that 40 percent of enterprise applications will feature task-specific AI agents by the end of 2026

How Agentic AI Changes the Attack Surface

Three properties separate agentic systems from web applications that security teams already know how to defend.

- **Instructions and data arrive on the same channel**

A traditional application separates executable code from user input. A language model consumes everything as text and has no reliable structural boundary between the instructions the developer wrote and the content it happens to read. A calendar invite, a support ticket, a web page, a code comment, or a payload of a connected tool can all carry instructions that the model will follow or execute. This single property is the root cause of most agentic threats.

- **Behavior is probabilistic rather than deterministic**

Unlike the traditional software, the same input can produce different outputs across runs. A control that blocks an attack phrased one way may completely miss the same attack phrased differently, rendering traditional signature-based defense ineffective. While traditional testing ensures a function returns an expected value, testing an agent requires evaluating behavior across a wide distribution of adversarial inputs, demanding an entirely different engineering discipline and tooling.

- **The blast radius extends through autonomous tool execution**

When a web application is exploited, the damage is controlled by what the application can reach. However, when an agentic AI application is manipulated, the damage propagates to the tools, credentials, datasets, and downstream systems the agent is authorized to use. By chaining autonomous actions, an agent granted access to a CRM, a secure email gateway, and a ticketing system can be weaponized into an untrusted insider carrying all those combined permissions.



"An AI agent compromised through a single line of untrusted text inherits every permission the enterprise granted it."

Layer One: The Prompt and Response Layer

The prompt and response layer is where the AI agent receives instructions and returns output. It is the most visible part of the surface and the most frequently attacked.

Direct and Indirect Prompt Injection

Direct injection is an attack where an attacker types adversarial instructions into an interface the agent reads from and attempts to override system prompts, bypasses built-in guardrails, or extracts configuration. Indirect prompt injection has more severe consequences where an attacker plants instructions within content that the agent will retrieve later, which implies that the attacker does not require access to any interface at all. The [EchoLeak vulnerability disclosed as CVE-2025-32711](#) in Microsoft 365 Copilot showed how far an indirect prompt injection attack can go. An attacker sent the victim an ordinary email. Copilot ingested it as retrieval context with no user click required, followed the embedded instructions, gathered sensitive internal files and exfiltrated them through an automatic image fetch to an allowed domain. The chain defeated the platform prompt injection classifier, the link redaction filter and the content security policy. Reports of prompt injection vulnerabilities rose sharply through the same period, with HackerOne reporting a 540 percent increase in 2025.

Jailbreak and Guardrail Evasion

Jailbreaks push a model outside the behavior its operator intended using role-play framing, encoding, translation, hypothetical scenarios or gradual escalation across many conversational turns. Attacks that unfold over an extended conversation are especially effective against agents, because the agent carries state forward and each individual message looks benign in isolation. A control that evaluates one message at a time will not see the pattern.

Data Leakage in Inputs and Outputs

Agents leak in both directions. Employees paste customer records, credentials, source code, and unreleased financials into prompts. Agents return data the requesting user was never entitled to see, because the model was given broad retrieval access, and no per user authorization check sits between the retrieval and the response. Common categories include personally identifiable information (PII), payment card data, national identifiers, API keys, cloud access tokens, repository tokens, and proprietary source code.

Toxic and Unwanted Content

Output that is harmful, biased, defamatory, or simply off topic creates brand, legal, and regulatory exposure, particularly for agents that face customers or the public. For regulated industries, the same category covers advice the organization is not licensed to give and claims it cannot substantiate.



Layer Two: The Model Context Layer

The model context layer supplies the AI agent with everything it knows beyond its training: retrieval augmented generation (RAG) stores, vector databases, conversation memory, and resources exposed through the Model Context Protocol (MCP). This layer receives far less scrutiny than the prompt layer and is more valuable to an attacker because content placed here is treated by an agent as trusted knowledge.

Memory and Context Poisoning

An attacker who can write to a knowledge base, a repository, a ticket queue or a shared memory store can plant content that changes agent reasoning long after the write. Poisoned context is persistent. It affects every user whose query retrieves it and it does not carry any obvious signals of an active attack. A single poisoned document in a corpus of millions can steer an agent toward a specific action whenever a matching query arrives.

Retrieval as an Untrusted Input Path

Most retrieval pipelines were built for relevance and not for adversarial resistance. Documents are chunked, embedded, and returned to the model without provenance checks, integrity verification, or sanitization. Any system that can write into the corpus including automated ingestion from email, web crawls, partner feeds, and internal ticketing becomes an injection path to every agent that reads it.

Model and Supply Chain Risk

Enterprises source models from public repositories, private fine tunes, and vendor endpoints. They compose agents from open-source frameworks and third-party components. Each source introduces a supply chain question: what was the model trained on, what behavior does it exhibit under adversarial input, who can modify the framework after deployment, and what happens when a dependency is updated. Organizations frequently discover models running in production that weren't registered, tested, or approved by any team.



Layer Three: The Tools and Actions Layer

The tools and actions layer is where an agent stops producing text and starts changing the world. It is the layer with the highest consequence and the newest, least mature controls.

The MCP Expansion

The Model Context Protocol (MCP) has become the standard way to connect agents to external tools, data, and services. Adoption has been extremely fast in the enterprise. Research conducted by Wiz found that MCP was present across 80 percent of cloud environments. Roughly one in six environments were found to be exposing at least one MCP server to the internet. Among those exposed servers, approximately 70 percent returned a complete tool catalog to unauthenticated callers, and about 42 percent returned real data when a tool was invoked anonymously, and roughly 10 percent exposed a sensitive backend system. Nearly all still negotiated a protocol version that predated the authentication capabilities added to the specification.

Independent testing of public MCP implementations has found high rates of conventional software flaws as well, including path traversal, command injection, and server-side request forgery, alongside a steady flow of published CVEs. The picture is a young protocol carrying production privilege in enterprise environments with authentication frequently absent and inventory frequently unknown.

Shadow and Rogue MCP

Developers stand up MCP servers to move quickly, and those servers rarely appear in any enterprise asset inventory. This resulting population of unregistered MCP servers carrying real credentials is known as shadow MCP. The adversarial version of this is known as rogue MCP, where an attacker presents a server that an attacker connects to either through a plausible name in the public registry, a compromised dependency, or a configuration change. Once the agent trusts the server, the attacker controls part of the agent reasoning input and can invoke tools on the agent's behalf.

Tool Poisoning and Tampering

Definitions also change after approval. In a rug pull, a server presents a benign tool during review and swaps the definition for a malicious one once it is trusted, which defeats any control that inspects tools only at registration. Continuous verification of tool and server definitions is the only reliable answer.

The agents read tool definitions as text. An attacker who controls a tool name, description, or parameter schema can hide instructions and inside metadata that a human reviewer would skim past. A description that reads as an ordinary utility can carry a trailing instruction telling the agent to forward user data to an external address on every invocation. The reviewer sees a helpful tool and the agent sees a command.

Tool Abuse and Excessive Agency

Agents are commonly provisioned with the union of every permission any of their tasks might require, and those credentials are often long lived and shared. An agent with database read access, an email tool, and a file transfer capability can be walked through an exfiltration sequence one legitimate call at a time. No single call is anomalous, and the composition is what causes the harm. This is why the survey findings on excessive access matter so much: broad privilege converts a prompt layer compromise into a material incident.

Human Trust Exploitation

Where a human approval step exists, the agent usually controls what a human sees. An agent under attacker influence can summarize an action accurately while omitting the part that matters, and the reviewer approves in seconds because the summary looks routine. Approval workflows that display agent generated descriptions rather than the underlying action provide much weaker assurance than they appear to.

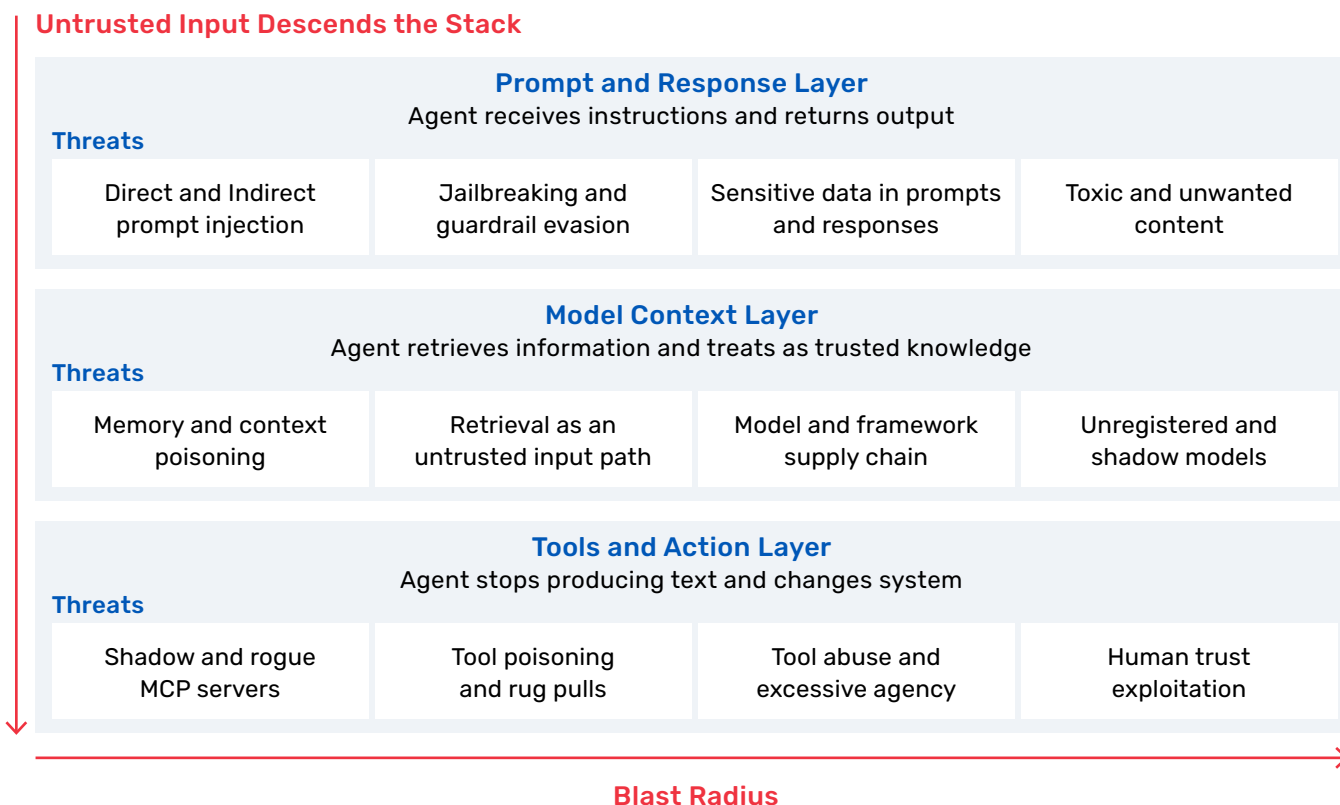


Fig 1. The Agentic AI Attack Surface

Multi-agent Environments: Where Risk Compounds

Few enterprises run a single agent. A typical estate now includes SaaS agents inside business applications, cloud platform agents on services such as AWS Bedrock, Azure AI Foundry, and Vertex AI, desktop and coding agents on employee workstations, and homegrown agents built by internal teams. They interoperate through MCP, agent to agent protocols, APIs, gateways, and shared memory, and they call the same underlying models and enterprise tools. Two properties of this estate turn individual weaknesses into systemic ones.

Agent to Agent Communication as an Attack Path

If MCP is how an agent reaches tools and data, Agent2Agent (A2A) is how an agent reaches other agents. The A2A protocol reached a stable version 1.0 specification under the Linux Foundation, with

more than 150 supporting organizations including AWS, Cisco, Google, IBM, Microsoft, Salesforce, SAP, and ServiceNow, and documented enterprise production use as of April 2026. Adoption still trails MCP by a wide margin, which makes this the part of the surface that is arriving rather than the part already under sustained attack. It is worth understanding before it scales.

The protocol works through “agent cards,” structured metadata that an agent publishes to advertise its identity, capabilities, and endpoints so that other agents can discover it and delegate work to it. That metadata enters the reasoning context of any agent that reads it, which reproduces the tool poisoning problem one level up. Researchers have demonstrated agent card poisoning, where instructions hidden in a card description are treated by the receiving agent as

authoritative planning input. This effectively allows an attacker to hijack the control flow and exfiltrate personal and payment data before the legitimate delegation ever executes. The lesson generalizes: wherever an ecosystem places descriptive metadata in front of a model, that metadata becomes an injection channel.

Delegation also carries trust that was never explicitly granted. An agent that accepts a task from a peer inherits an instruction it did not originate and usually cannot trace to a human. Without authentication and integrity protection on that channel, any agent that can reach another can direct it, and the receiving agent applies its own privilege to the request—introducing a severe “confused deputy” vulnerability where low-privilege agents can trigger high-privilege actions. Signed agent cards in the version 1.0 specification provide cryptographic identity verification and are the baseline control, though signing merely verifies who published a card rather than whether its contents are safe to follow. Inspection of the message content remains necessary.

Cascading Failure and Agent Sprawl

Once agents call other agents, a compromise at one node propagates as trusted input to the next. An injection that begins in a low value workflow, such as a summarization agent reading a public web page, can reach a high value one, such as an agent with write access to a financial system, through a chain of ordinary delegations. Each hop looks legitimate to the agent receiving it, and the enterprise sees a set of successful tasks rather than an attack in progress.

Agent sprawl and lack of inventory constitute the second systemic property. Agents are created by many teams, on many platforms, with no central registration step, so most organizations cannot produce a complete list of the agents operating in their environment, what each is permitted to do, which peers and servers it talks to, and who owns it. Security controls cannot be applied to assets that no one has enumerated, and incident response cannot trace a chain of delegations through agents that were never recorded.

Every Step is a Permitted Operation Chains Can Pass Unnoticed

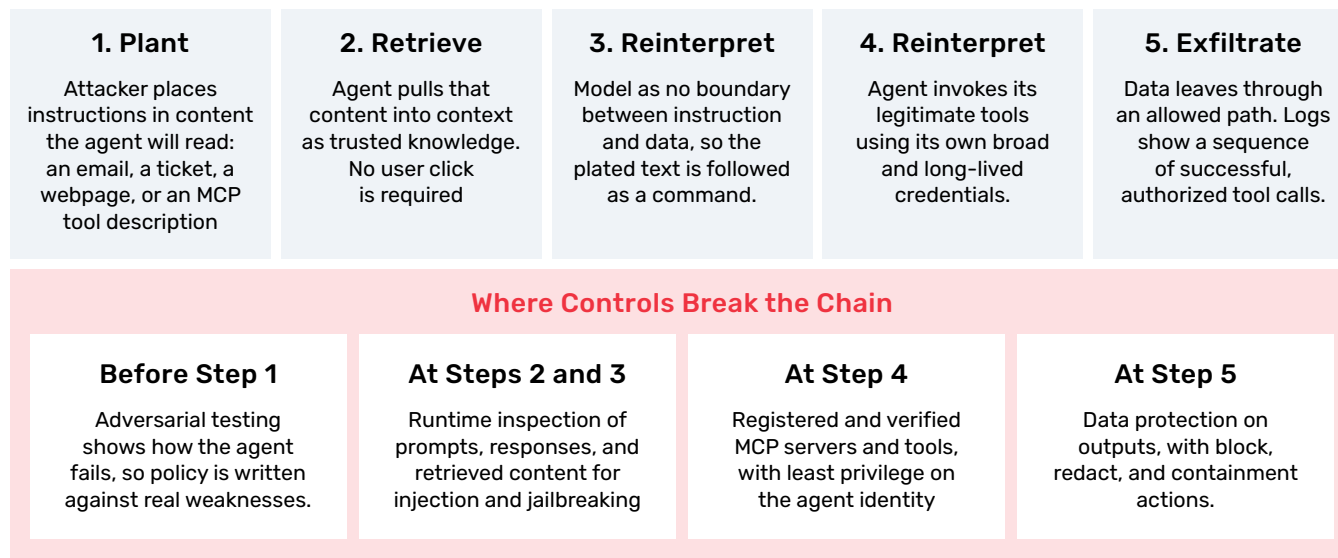


Fig 2. Anatomy of an Agentic Breach

Mapping Agentic AI Risk to Industry Frameworks

Three frameworks give security teams a shared vocabulary for this surface and a defensible basis for reporting.

1. The OWASP Top 10 for Agentic Applications,

published for 2026, is the closest fit to the layer model above. It covers agent goal hijack, tool misuse and exploitation, identity and privilege abuse, agentic supply chain vulnerabilities, unexpected code execution, memory and context poisoning, insecure inter-agent communication, cascading failures, human and agent trust exploitation, and rogue agents. Two of those categories, insecure inter-agent communication and cascading failures, exist specifically because agents now talk to each other over protocols such as A2A, which is a useful signal of where the industry expects the next wave of incidents. The earlier OWASP Top 10 for Large Language Model Applications remains the reference for prompt-layer risks such as prompt injection, sensitive information disclosure, and insecure output handling.

2. MITRE ATLAS provides an adversary tactic and technique matrix for AI systems in the same structure as ATT&CK, which makes it useful for threat modeling, detection engineering, and red team reporting that security operations teams can consume without translation.

3. The NIST AI Risk Management Framework supplies the governance layer, organizing work into govern, map, measure, and manage functions. It is the framework most often referenced by boards, auditors, and regulators, and it is where evidence from testing and runtime monitoring ultimately needs to land.



Mapping findings to these frameworks at the point of discovery, rather than at reporting time, is what makes AI security evidence usable for audit and board reporting.

Six Requirements Every Agentic AI Security Strategy Needs

Existing controls address adjacent problems. Web application firewalls inspect structured requests for known signatures and have no view into semantic manipulation of a model. API gateways authenticate callers and enforce rate limits without evaluating whether the content passing through carries an injection. Data loss prevention tools operate on files and channels rather than on prompts and completions. Secure web gateways see traffic to a domain without seeing the meaning of a conversation. Each remains necessary, and none was built for agentic AI.

A control set built for agentic AI needs six capabilities.

1. Discovery and inventory

A single system of record for models, applications, agents, MCP servers, and tools, including ownership, configuration, and operational status, so that shadow deployments surface rather than accumulate.

2. Adversarial testing before deployment

Systematic red teaming that exercises the model, the application, and the full agent execution path across single turn, extended conversation, and agentic techniques, with results mapped to recognized frameworks.

3. Runtime inspection of prompts and responses

Real-time analysis of inputs and outputs for injection, jailbreaking, sensitive data exposure, and unwanted content, at production latency and production volume.

4. Governance of the context and tool layers

Registration and approval of MCP servers and tools, inspection of tool metadata for hidden instructions, and continuous verification that definitions have not drifted since approval. The same discipline must extend to agent-to-agent channels, covering the peers an agent may accept work from, and the metadata it reads to find them.

5. Enforcement with a graduate response

The ability to block, redact, flag, or route for review according to policy, plus containment controls that can halt an agent session or revoke tool access when something goes wrong.

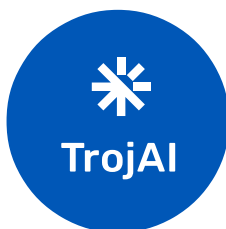
6. Evidence and integration

Findings and events delivered into existing security operations workflows, with framework mapping and audit trails that satisfy compliance and incident response requirements.

Two further approaches are practical rather than functional. Testing and runtime defense should inform each other so that a weakness found in testing becomes a policy in production, and a novel runtime detection becomes a new test case. The platform should deploy where the data is permitted to live, which for regulated enterprises usually means a self-hosted deployment inside their own cloud environment.

Securing the Agentic Attack Surface with TrojAI by A10 Networks

TrojAI by A10 Networks is an AI security platform built for organizations that are deploying AI agents and applications into production. It covers the lifecycle from build time testing through runtime enforcement, and it addresses each of the three layers described in this paper. The platform is fully customizable, deploys as a self-hosted installation in the customer cloud environment of choice or as a cloud service, and supports AWS, Azure, Google Cloud, and OpenShift.



TrojAI by A10 Networks Across the AI Lifecycle

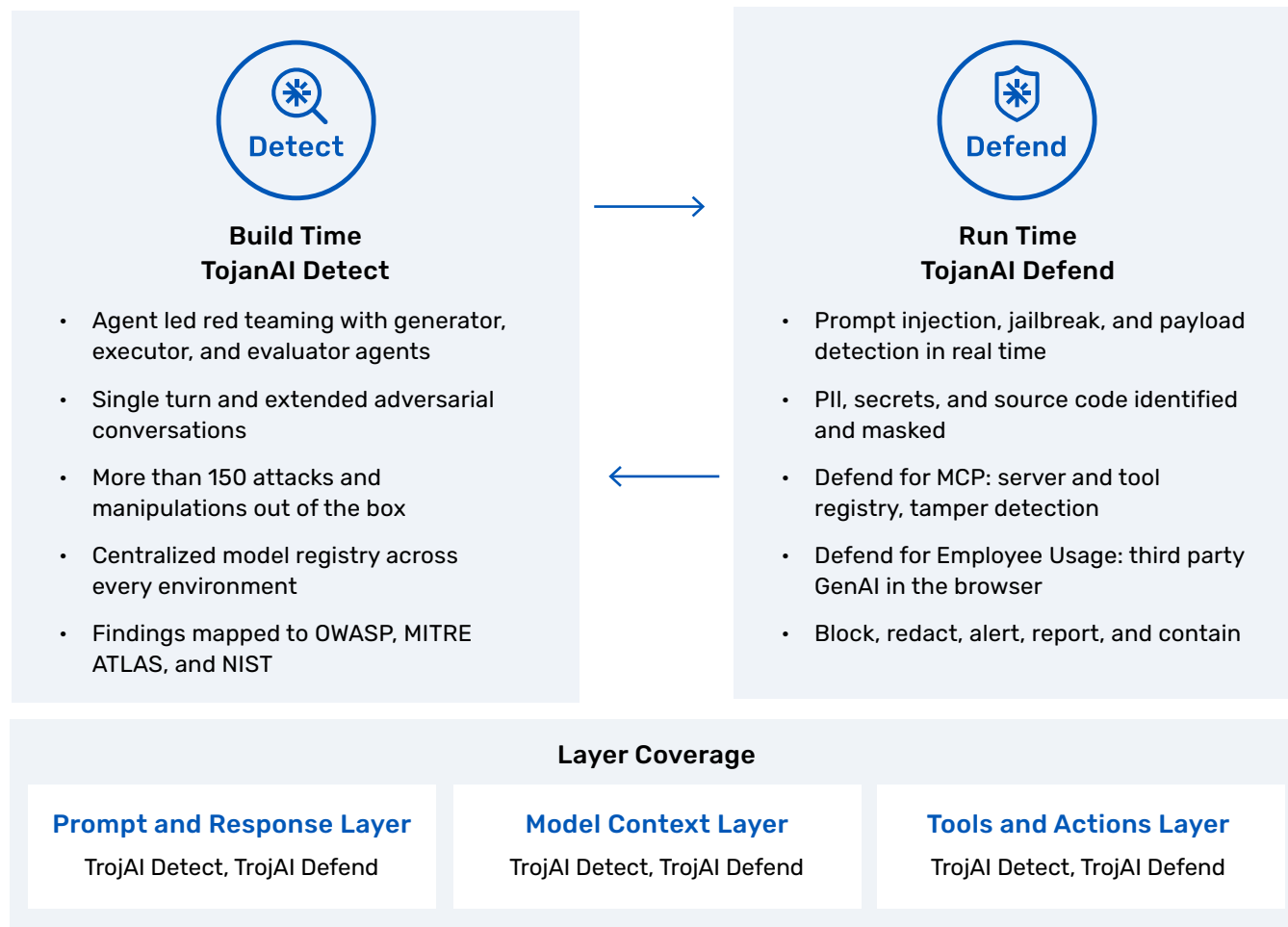


Figure 3. TrojAI by A10 Networks Across the AI Lifecycle

TrojAI Detect: Automated Adversarial Red Teaming

TrojAI Detect finds risks and flaws of AI systems before deployment, providing two automated red-testing modes: custom and agent-led red teaming against AI models, application and agents. The agent-led red teaming uses self-learning adversarial agents. A generator agent crafts adversarial content, an executor agent runs attacks according to a playbook strategy, and an evaluator agent judges the result and feeds what it learned back into the next attempt, so coverage improves successive runs.

The testing engine supports single turn attacks for targeted prompt and response safety and extended adversarial conversations for agents and conversational models. It ships with more than 150 attacks and manipulations out of the box, supports custom attack sets and production datasets, and evaluates behavior with block lists, similarity refusals, and language model content moderation. Red teaming content is generated dynamically using ablated models, with the option to bring your own content. A centralized model of registry tracks every model regardless of where it runs, which addresses the inventory problem directly.

Results map automatically to the OWASP Top 10 for LLMs, MITRE ATLAS, and NIST, producing evidence that supports model approval processes and regulatory reporting. Findings then feed policy creation for runtime protection, helping close the loop between what testing discovered and what production enforces.

TrojAI Defend: Runtime Defense for Agents and Applications

TrojAI Defend is a purpose-built AI application and agent firewall that inspects traffic in real time. It analyzes AI interactions including prompts and responses and blocks threats like prompt injection, jailbreak attempts, malicious payloads, data leakage and toxic contents, and it identifies and

masks sensitive data in both directions to prevent exposure of personally identifiable information, confidential records, intellectual property, secrets, and source code.

The AI-driven detection uses defense-in-depth across several methods: pattern matching, static rule sets, AI and machine learning classifiers, and proprietary LLM moderation. The policy engine supports predefined and custom rule sets and a range of response actions including block, allow, redact, alert, and report. Organizations can extend detection with their own models.

Deployment uses a reverse proxy architecture for straightforward integration; alerts map to the OWASP LLM and MITRE ATLAS frameworks, and events integrate into existing security operations tooling. In one financial services deployment, TrojAI Defend protects tens of thousands of users across hundreds of applications transacting hundreds of thousands of prompts per day, with policies covering toxic language in English and other languages, sensitive data and PII leakage, secrets and API key leakage, custom patterns including invisible Unicode characters, and off topic or non-compliant content.



TrojAI Defend for MCP: Governing the Tools and Actions Layer

TrojAI Defend for MCP applies the same runtime discipline to MCP workflows. It discovers and registers every MCP server and tool, creating the inventory that shadow MCP otherwise prevents, and it approves servers and tools in a registry so that unregistered and rogue servers are blocked. It identifies all MCP traffic to and from each server and eliminates hidden communication paths used for prompt injection and data exfiltration.

It unmask malicious tool descriptions that conceal prompt injections and continuously verifies every server and tool definition, detecting drift or modification in real time. When an unauthorized change appears, the affected tool is isolated and blocked, which stops tampering, poisoning, and rug-to-pull attacks before they spread. Customizable policies inspect and audit every agent interaction, enforce enterprise data handling rules, and maintain the audit trails required for compliance and incident response.

TrojAI Defend for Employee Usage: Third-party GenAI

Much enterprise AI risk arrives through applications the security team does not operate and manage. TrojAI Defend for Employee Usage monitors employee use of popular third-party generative AI applications through a browser extension, blocks and redacts prompts containing sensitive data in real time and provides the visibility and usage policy enforcement needed to demonstrate compliance.

One Vendor Across Every Layer

TrojAI solutions sit within a broader A10 Networks portfolio that spans the infrastructure, application, and AI layers. A10 Thunder ADC provides high-performance application delivery, load balancing, and TLS/SSL encryption offload for AI infrastructure, with always on DDoS mitigation from A10 Defend. ThreatX by A10 Networks delivers web applications and API protection and bot management for application layer. A10 AI Gateway intelligently routes AI traffic with policy enforcement, observability, and AI/LLM usage control.

TrojAI by A10 Networks secures the AI layer. For enterprises already running A10 infrastructure, AI security becomes an extension of a platform they operate today rather than another disconnected tool. TrojAI was born out of early adversarial AI research, has received more than twenty Gartner mentions including the Innovation Guide for Generative AI in Trust, Risk and Security Management and multiple Hype Cycles, and is deployed in large, complex Fortune 500 enterprises and government organizations. The product roadmap continues to follow the agentic surface, with an agent registry for enterprise-wide agent inventory, agent runtime protection covering desktop and custom agents with out-of-the-box policies for the OWASP Top 10 for Agents, and support for agent-to-agent protocols, agent runtime intelligence that provides visibility beyond the prompt into tool calls and memory calls, and containment controls that can halt agent traffic and disable tool access.

Conclusion

The agentic attack surface has three layers. Defending just one of them is not sufficient. Prompt injection at the first layer becomes valuable to an attacker because of what the second and third layers make reachable. Controls must cover prompts and responses, the context an agent retrieves, and the tools it is permitted to call.

The tools and actions layer carries the highest consequence and the least maturity. MCP is present across a majority of cloud environments; exposure to the internet is common, authentication is frequently absent, and tool definitions can change after approval. Discovery, registration, and continuous verification of servers and tools are foundational rather than optional.

The same attack pattern follows the ecosystem into every new protocol. Instructions hidden in tool descriptions and instructions hidden in agent cards are the same attack aimed at a different piece of metadata. Organizations that build controls around a single protocol will rebuild them with each new one, while organizations that treat all model readable metadata as untrusted input will already be covered when the next standard arrives.

Testing and runtime defense are one program. Adversarial testing before deployment tells an organization how its agents fail, and that knowledge is what makes runtime policy accurate. Runtime detections in turn reveal attack patterns that belong in the next round of testing. Organizations that run these as separate efforts get weaker results from both.

Traditional controls remain necessary and are not sufficient on their own. Firewalls, API gateways, and data loss prevention tools were designed for structured traffic and known signatures, and the agentic surface is semantic, probabilistic, and consequential. Security for this layer must be built.

TrojAI by A10 Networks delivers coverage across the AI lifecycle. **TrojAI Detect** red teams models and agents before deployment and maps findings to OWASP, MITRE, and NIST. **TrojAI Defend** inspects and enforces AI interactions in real time to protect and govern agents, applications, models and MCP tools. Together they give security teams the confidence to let the business adopt and scale AI agents, with the testing evidence and runtime control that responsible deployment requires.

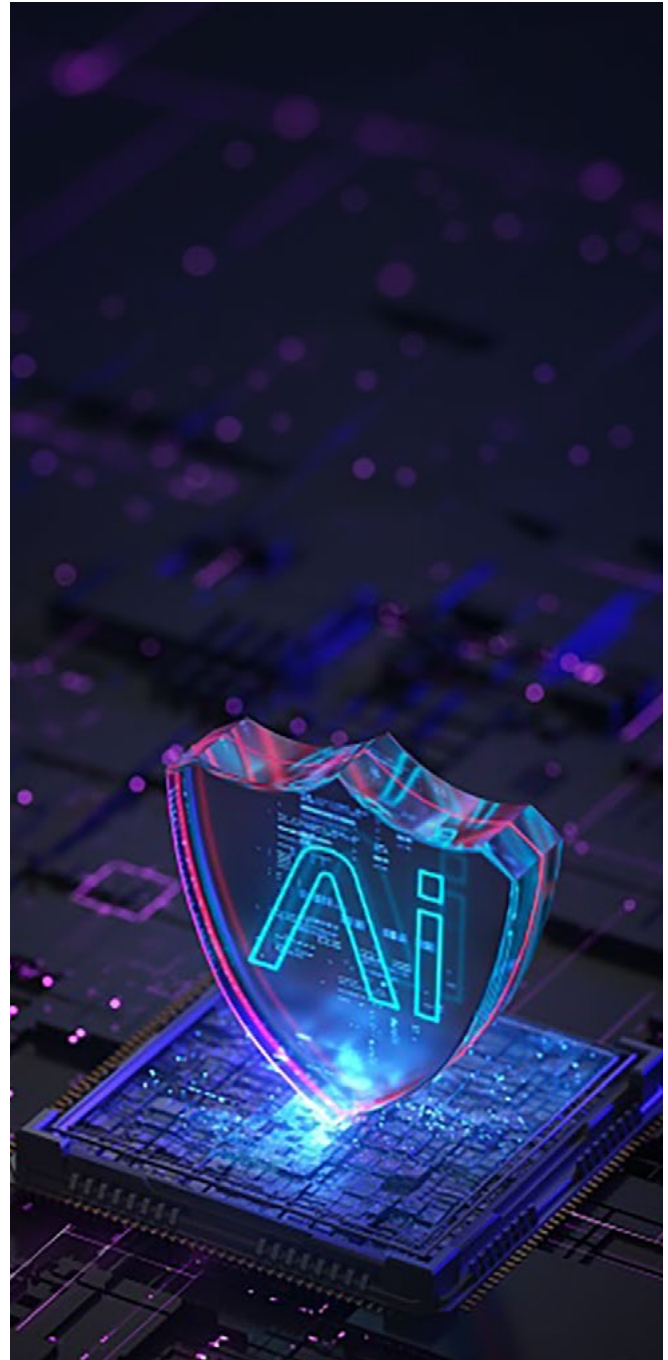
Sources

Gartner. "Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025." 26 Aug. 2025.
Cloud Security Alliance and Aembit. "Identity and Access Gaps in the Age of Autonomous AI." Survey of 228 IT and security professionals, Jan. 2026.
Wiz Research. "The Security Risks Hiding Behind Exposed MCP Servers."
Microsoft. CVE-2025-32711, Microsoft 365 Copilot information disclosure (EchoLeak). 2025.
OWASP. "Top 10 for Agentic Applications." 2026. OWASP. "Top 10 for Large Language Model Applications."
Linux Foundation. "A2A Protocol Surpasses 150 Organizations, Lands in Major Cloud Platforms, and Sees Enterprise Production Use in First Year." 9 Apr. 2026.
Keysight Technologies. "Agent Card Poisoning: A Metadata Injection Vulnerability in the Systems using Google A2A Protocol." 12 Mar. 2026.
MITRE ATLAS. Adversarial Threat Landscape for Artificial Intelligence Systems.
NIST. Artificial Intelligence Risk Management Framework (AI RMF 1.0).

About A10 Networks

A10 Networks provides security and infrastructure solutions for on-premises, hybrid cloud, and edge-cloud environments. Our 7000+ customers span global large enterprises and communications, cloud and web service providers who must provide business-critical applications and networks that are secure, available, and efficient. Founded in 2004, A10 Networks is based in San Jose, Calif. and serves customers globally.

For more information, visit A10Networks.com and follow us at [@A10Networks](https://twitter.com/A10Networks).



About A10

A10Networks.com

Contact Us

A10Networks.com/contact

©2026 A10 Networks, Inc. All rights reserved. A10 Networks, the A10 Logo, A10 Control, A10 Defend, A10 Harmony, Harmony, A10 Thunder, Thunder, ACOS, A10 SSL Insight, SSL Insight, SSLi, vThunder, ThreatX, ThreatX Protect, and TrojAI are trademarks or registered trademarks of A10 Networks, Inc. or its affiliates in the United States and other countries. All other trademarks are property of their respective owners. A10 Networks assumes no responsibility for any inaccuracies in this document. A10 Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice. For the full list of trademarks, visit: A10Networks.com/a10trademarks.