snyk

WHITE PAPER

What's Lurking in your AI?

A deep dive into AI Security Posture Management (AI-SPM)

Why securing your AI systems, models, prompts, and agents is now just as critical as securing your code.

SECTION I.

Foundations of AI-SPM

What is AI Security Posture Management?

Application Security Posture Management (ASPM) brought order to chaos. It helped teams manage scattered test results, spot security blind spots, and unblock remediation bottlenecks across increasingly complex application environments. It works well, especially when applications behave predictably.

But AI changed the game.

AI-native systems don't follow the same tidy rules. They're built on probabilistic engines trained on oceans of data. They evolve. They generate. Sometimes, they even decide. Their components, prompts, models, embeddings, and agents shift in real time. They don't leave consistent audit trails. They reshaped your risk surface before last week's scan results were even reviewed.

This is where AI Security Posture Management (AI-SPM) comes in.

AI-SPM isn't a replacement for ASPM. It's a specialized discipline designed to tackle the unpredictable nature of AI. It doesn't just look at misconfigurations or vulnerable APIs. It analyzes model behavior, prompt manipulation, plugin use, fine-tuning drift, and the unexpected choices made by autonomous systems.

AI-SPM works alongside tools such as SAST, DAST, SCA, and ASPM, adding new lenses that expose risks across models, logic flows, and contextual inputs. It integrates into existing workflows, expanding your visibility into previously invisible or unstable components.

What AI-SPM changes isn't just where teams look for risk, but how they manage it. In AI-native systems, posture can't be established solely through periodic scans or manual review. Risk emerges continuously as models evolve, prompts change, agents gain new capabilities, and integrations expand.

Because securing AI isn't just about adding new tools. It's about redefining what "secure" means when your application can reason, generate, and act.

Why now?

Until recently, AI security was easy to ignore. Most use cases were experimental or isolated. Today, AI is powering customer-facing features, operational workflows, and entire lines of business. Organizations are launching LLM-driven chat interfaces, AI copilots, internal search engines, and agentic decision-making systems, all of which handle sensitive data, make critical decisions, and run autonomously across connected systems.

This shift happened much faster than security teams could reasonably prepare for. Guardrails that work well for traditional applications don't apply to non-deterministic systems. A single model update, plugin install, or prompt tweak can radically alter behavior. Worse, many of these risks aren't visible in source code or CI/CD pipelines. They live in model weights, inference chains, embedded tools, and unpredictable user inputs.

As a result, many organizations don't know which AI components are in use, how they interact, or how changes in one part of the system affect overall risk. Without continuous visibility and control, posture degrades faster than teams can respond.

Security teams are also facing increasing external pressure. Regulations such as the EU AI Act and NIST's AI Risk Management Framework are moving from guidance to enforcement. Audits and incident investigations are beginning to ask questions that traditional ASPM tools can't answer: What models are in use? Were any fine-tuned using sensitive data? How does the system prevent prompt injection or data leakage at runtime?

Meanwhile, business leaders are making high-stakes bets on AI to drive growth, reduce costs, and differentiate from competitors. When an AI failure happens, whether due to a model hijack, data exfiltration, or hallucinated decision, it's not just an engineering issue. It's a brand, compliance, and operational risk. AI-SPM helps organizations stay ahead of these risks by introducing systematic visibility and governance into AI development, without slowing down innovation.

Two sides of the coin: Security for AI vs. AI for security

There's a growing interest in using AI to defend systems, whether through anomaly detection, intelligent triage, or automated remediation. These efforts matter. But they're only one side of the story.

AI-SPM focuses on the other side — securing AI itself.

It's one thing to use machine learning to enhance a SOC. It's another to trust LLMs to write code, access sensitive data, make decisions, and act through plugins. These systems need to be treated as security-critical from day one. That means securing their inputs (prompts), outputs (responses), logic (models and embeddings), and integrations (plugins, APIs, and data flows).

Attackers are already adapting to this new terrain. Prompt injection, model inversion, data poisoning, and supply chain manipulation are no longer theoretical. These are real risks that can lead to data exfiltration, access control failures, and unexpected behavior. And because AI systems often retrain or fine-tune based on recent data, even small manipulations can create cascading vulnerabilities.

```
AI-SPM doesn't just identify these issues.
It helps teams map their AI assets, assess
exposure, enforce guardrails, and simulate
attacks before they happen.
```

AI-SPM must operate at the same speed and scale as the systems it protects. Manual reviews, disconnected tools, and human-driven workflows can't keep pace with AI-native environments where risk changes continuously and across systems.

It blends familiar concepts like threat modeling and policy enforcement with new practices like red teaming LLMs and benchmarking model behavior under adversarial pressure.

By treating AI security as a continuously executed capability rather than a one-time assessment, AI-SPM creates a foundation for defensible AI. While it's useful to explore how AI can defend us, it's essential to first ensure the AI we deploy is defensible. AI-SPM brings that defensibility within reach by giving teams the structure, visibility, and execution model needed to build secure, explainable, and governable AI-native applications.

SECTION II.

What makes AI a unique security challenge?

Traditional applications behave predictably. They follow structured logic explicitly coded by humans. When a developer writes a rule, it executes the same way every time. Security reviews, testing tools, and policy enforcement are built on that assumption.

```
AI-native systems break it.
```

Large language models are probabilistic. Their behavior changes based on phrasing, context, training data, and surrounding tools. Identical inputs can produce different outputs, and small changes to weights, prompts, or integrations can shift behavior in unexpected ways with real security impact.

As a result, risk is no longer tied to fixed execution paths. It emerges from how models interpret context, chain reasoning, and interact with other systems at runtime.

The attack surface expands accordingly. Instead of being limited to APIs or infrastructure, AI systems expose prompts, data pipelines, inference chains, memory states, and agent instructions. Each new integration adds potential entry points, and in agentic systems, those entry points interact. A single prompt can trigger a chain of actions across tools, data sources, and services, creating attack paths that are difficult to predict or constrain.

Much of this behavior is opaque. Models are often treated as black boxes, and non-determinism makes traditional static and dynamic testing unreliable.

> In AI systems, actions aren't always traceable to clear code paths; they may emerge from inferred relationships between services, tools, or memory states, leaving gaps in visibility, policy enforcement, and threat modeling.

This is why AI security failures rarely stem from a single vulnerability. They arise from how multiple components interact under specific conditions.

Enforcement becomes harder when policy is built on probability rather than determinism. Logs, traces, and adversarial testing become more reliable than static review, but point-in-time testing and isolated controls still struggle to capture full system behavior.

Data flow adds another layer of complexity. Prompts may become training inputs. Agent outputs may persist in logs or memory. Plugins may access unintended endpoints. These fluid, bidirectional flows break traditional data classification models and allow risk to propagate long after an initial change.

Taken together, these characteristics make AI not just another attack surface, but a fundamentally different type of system to secure.

Threats and vulnerabilities to know

Securing AI starts with understanding the threats that target these systems. Many vulnerabilities stem from how models are trained, prompted, and deployed, not just from the surrounding infrastructure.

Unlike traditional application flaws, AI threats rarely exist in isolation. They emerge from interactions between models, data, prompts, agents, and tools across the lifecycle. Issues that seem low risk on their own can escalate when combined with other system behaviors.

Some threats appear during training, others at inference or runtime, and many originate in supply chain dependencies or agent orchestration. Defending against them requires understanding how risk accumulates and propagates across the system, not just identifying individual weaknesses.

Data Poisoning

Attackers inject malicious examples into training data to influence a model's behavior at inference. Poisoned datasets can suppress outputs, distort responses to specific prompts, or introduce unpredictable behavior, particularly in open pipelines or systems that fine-tune on user-generated data.

Although the effects may be subtle at first, poisoned data often shapes behavior that only surfaces later, when combined with specific prompts, agents, or integrations, making it a frequent foundation for downstream attacks.

Adversarial Input and Model Evasion

These attacks exploit how models interpret input by using misspellings, encoded tokens, or indirect phrasing to bypass safety controls and trigger restricted behavior. Attackers use iterative probing to refine prompts based on observed responses.

When models are connected to tools or agents, successful evasion can escalate quickly. What begins as an unsafe response can turn into unintended actions across connected systems.

Model inversion and extraction attacks

Through repeated querying, attackers can reconstruct sensitive data used during training. These attacks may expose personal information, proprietary code, or confidential documents, particularly in models trained on private datasets.

Because inversion and extraction rely on sustained, low-noise interaction, they are difficult to detect without behavioral monitoring and usage context.

Membership inference

Membership inference allows attackers to determine whether a specific data point was included in a model's training set. In regulated contexts such as healthcare or finance, confirming the presence of an individual's data can violate privacy laws and expose sensitive operational details.

Even partial confirmation can carry significant regulatory and reputational consequences.

Model theft

By issuing high-volume queries, attackers can approximate a model's functionality and recreate it under a different brand or wrapper.

This undermines intellectual property and enables misuse, including deploying the cloned model without safety controls or embedding malicious behavior.

Because these attacks often resemble legitimate use, they frequently bypass rate limiting and perimeter-based defenses.

Infrastructure attacks

AI systems rely on APIs, plugin frameworks, and deployment pipelines that reintroduce familiar risks. If model output is blindly passed to tools or endpoints, vulnerabilities such as SSRF, XSS, or command injection can resurface, turning LLMs and agents into attack vectors.

In agentic systems, a single unsafe output can trigger automated actions across multiple services, significantly expanding the blast radius.

Supply chain manipulation

AI applications depend heavily on shared resources, including open source libraries, pre-trained models, plugins, and public datasets. If any component is compromised, it can introduce security issues that scale across systems. These assets are often invisible to traditional SBOMs.

Unlike conventional dependencies, AI supply chain components shape behavior rather than just execution. They persist across retraining and fine-tuning cycles, making compromise harder to isolate and remediate.

Because these components are frequently shared and updated across teams, a single upstream issue can propagate widely before it is detected. Without continuous visibility into AI-specific dependencies and their usage, organizations are left reacting after risk has already spread.

These aren't speculative issues. Red teaming exercises and public incident reports have shown how quickly these attack patterns are moving from research to reality. Securing AI applications requires recognizing that threats aren't limited to code. They're embedded in data, learned behavior, and the way models reason through complex tasks.

In practice, this means attackers aren't exploiting a single flaw. They're probing how AI systems behave under pressure and how small weaknesses can be chained into larger failures.

Who's behind these attacks?

The threat actors targeting AI-native systems are as diverse as the technologies they exploit. From highly organized operations to lone actors, motivations range from geopolitical leverage to simple disruption or economic gain.

Nation-state actors are drawn to AI's strategic value. Generative models embedded in commercial and industrial systems are attractive targets for intelligence gathering, manipulation, and sabotage. These actors have the resources to reverse-engineer models, poison public datasets, or test prompt-based attacks at scale. Their objective often goes beyond data theft. They aim to influence decisions, degrade trust, or quietly undermine systems that increasingly rely on AI-assisted logic.

In these cases, success is measured in persistence and impact over time, not immediate exploitation.

Cybercriminal groups view AI as both a target and an opportunity. Some attempt to hijack agents or manipulate chatbots to exfiltrate sensitive information. Others focus on indirect monetization, such as extracting training data, cloning paid APIs, or selling jailbreak techniques. As AI systems gain access to tools and workflows, these attacks increasingly shift from content abuse to operational abuse, where compromise leads directly to financial loss.

Corporate competitors can also pose a risk, whether intentionally or through negligent reuse. Attempts to infer training data, replicate model behavior, or seed poisoned data into shared corporate resources blur the line between competitive intelligence and security incidents. Attribution is often difficult, and the impact can persist unnoticed across products and teams.

Hacktivists and ideologically driven attackers target AI systems for visibility. By bypassing moderation controls or provoking biased or inflammatory outputs, they aim to expose perceived failures and erode public trust. Because these attacks are designed for amplification rather than stealth, reputational damage can occur even without a traditional breach.

> Across all these actors, one pattern holds: Attackers aren't only adapting to AI, they're shaping how these systems evolve. And unless development teams anticipate their methods, the threat surface will continue to expand unchecked.

Defending against them requires more than spotting individual attacks. It requires understanding how different threat behaviors surface across models, agents, data, and integrations over time.

What's at stake?

The security risks tied to AI-native applications are no longer theoretical. Real-world incidents already show how poorly controlled AI behavior can translate into material damage.

In one case, a large language model embedded in a public-facing tool began leaking training data after repeated adversarial prompts. A journalist recovered internal Slack messages, employee credentials, and production logs from the model's context window. The cause was not a single error, but a systemic failure: internal data entered training unchecked, model behavior was never tested under adversarial pressure, and the leakage went unnoticed until it became public.

In another incident, an autonomous AI agent connected to a procurement system was manipulated into placing a fraudulent order by exploiting how it interpreted goals and interacted with integrated APIs. No traditional vulnerability was exploited. The attacker simply understood how the model reasoned and acted.

As AI systems gain autonomy and access to real-world workflows, subtle manipulation can translate directly into financial or operational impact without triggering traditional security alarms.

These failures erode trust quickly. Models that leak sensitive data, hallucinate in critical environments, or behave unpredictably can damage reputations in minutes. Legal and regulatory exposure follows closely behind. In sectors such as healthcare, finance, and government, organizations are increasingly expected to demonstrate not only that controls exist, but that they are enforced continuously and effectively.

The risk extends beyond digital systems. AI agents connected to physical environments, including manufacturing lines, IoT systems, or autonomous vehicles, raise the stakes further. When AI decisions cross into the physical world, the margin for error narrows dramatically.

Beyond any single incident, the broader consequence of weak AI security is loss of control. When organizations don't know what models are running, how they were trained, or what they're connected to, they cannot accurately assess exposure or respond with confidence. That loss accumulates over time as models change, agents gain capabilities, and integrations expand faster than governance can keep up.

> AI systems are no longer side experiments. They're business-critical, connected, and exposed. Without visibility, testing, and governance, the risks don't stay theoretical for long.

The AI-SPM lifecycle playbook

From DevSecOps to AISecOps

DevSecOps transformed how organizations build and secure software by embedding security across the development lifecycle. It shortened feedback loops, improved coverage, and brought developers into the security process early. But as applications become AI-native, the assumptions behind that model begin to break down.

Traditional DevSecOps pipelines are built for deterministic code. They scan libraries for known vulnerabilities, analyze static logic, and test APIs through defined routes and schemas. AI development does not follow those patterns. It introduces datasets, weights, prompts, embeddings, and memory states that often live outside standard repositories and change independently of application code. These elements introduce risks that most security tools were never designed to detect.

Applied without adaptation, DevSecOps can create a false sense of coverage. Teams may secure the surrounding application while the AI system itself remains largely unexamined.

AISecOps adapts DevSecOps principles to this new architecture. It extends security beyond source control and CI/CD to cover the full AI lifecycle, including data collection, fine-tuning, model orchestration, and prompt design. It also requires closer collaboration between security teams, AI engineers, data scientists, and platform teams, with the shared goal of securing the entire system, not just the application wrapper.

Unlike traditional DevSecOps, AISecOps cannot rely on manual processes or periodic checks. AI systems evolve continuously, and security practices must evolve with them.

In practice, AISecOps focuses on aligning model behavior with business risk. It emphasizes guardrails that function in probabilistic environments, testing that simulates real-world attacks, and continuous visibility across models, data, and agents. Security shifts from static assessment to ongoing execution, and from isolated controls to coordinated enforcement across the AI lifecycle.

Securing the AI/ML lifecycle (AISecOps)

The machine learning lifecycle extends far beyond training and deployment. Each phase introduces distinct security risks, and every handoff between teams creates new opportunities for exposure. Securing this lifecycle means identifying where risk enters and addressing it early, before it becomes embedded in production systems.

In AI-native environments, lifecycle stages rarely operate in isolation. Data, models, and configurations move continuously between phases, allowing issues introduced early to persist or reappear later in unexpected ways.

Data collection

Data collection is often the first point of exposure. If attackers can influence datasets, especially in open or user-submitted environments, they can poison inputs or subtly bias model behavior. Even trusted datasets can introduce risk when they contain unlicensed material, embedded secrets, or sensitive personal information.

Because data is frequently reused, shared, or incrementally updated, weaknesses introduced at this stage can propagate across multiple models and downstream systems if they are not detected early.

Training

Training is where models internalize patterns and where risk can become embedded. Poisoned, skewed, or imbalanced data can push models toward misleading or harmful behavior. Poorly governed fine-tuning can override safeguards or leak system prompts into inference responses.

Training is rarely a one-time event. Continuous fine-tuning and retraining introduce drift, making it essential to monitor behavioral changes and increased susceptibility to abuse over time.

Validation

Validation goes beyond accuracy. It is about understanding misuse potential. Effective validation includes adversarial testing such as jailbreak attempts, prompt fuzzing, and stress testing for unintended behavior. It must also assess bias, overfitting, and membership inference exposure.

When treated as a final gate, validation quickly loses value. In evolving AI systems, validation must be repeatable and responsive to changes in data, prompts, models, and integrations.

Deployment

Deployment is more than releasing a model. It involves integrating APIs, tools, memory systems, and user inputs, all of which expand the attack surface. Secure deployment requires access controls, version tracking, real-time monitoring, and the ability to isolate or disable compromised models before damage spreads.

In AI-native systems, deployment is not an endpoint. Models, prompts, tools, and integrations continue to change in production, requiring controls that remain effective as behavior evolves.

MLSecOps makes these practices repeatable and accountable. It gives security a structured way to collaborate with model developers and ensures AI systems carry the same level of risk rigor as traditional code. That rigor depends on continuous enforcement, not one-time approvals, especially as deployed systems gain autonomy and access to downstream services.

Securing the data pipeline

Data pipelines power how AI systems reason, recommend, and act. They provide the context models rely on during training and inference, but they also represent one of the highest-risk components of the AI lifecycle. Much of this data originates from untrusted, third-party, or user-generated sources.

Training and inference introduce different risks, but both are critical. During training, compromised data can alter model weights or introduce hidden backdoors. During inference, malicious inputs can trigger unintended behavior, extract sensitive information, or influence downstream systems. Because AI systems often blur these phases, such as through fine-tuning on recent interactions, issues introduced at inference can persist and resurface later in model behavior.

Common threats include bias, poisoning, and leakage. Bias emerges when datasets overrepresent or underrepresent certain groups or behaviors. Poisoning introduces malicious inputs designed to change how a model responds. Leakage happens when models memorize sensitive inputs and later reveal them, either directly or through inversion techniques.

Managing these risks requires more than data cleaning. It demands governance across the entire pipeline. Organizations need visibility into data provenance, how data is processed, and what information is allowed to influence model outputs. They must also audit what is retained or cached during inference, particularly in agent-based systems that rely on memory.

Without continuous oversight, data pipelines can quietly reintroduce risk long after initial validation. AI systems fail not only because of malicious code, but because of data drift, unfiltered inputs, or flawed assumptions embedded in context. AI-SPM addresses this by treating data flows as security-critical assets, not afterthoughts.

Securing the model

Once trained, a model becomes the core of an AI-native application, but securing it is an ongoing effort. Models evolve through fine-tuning, configuration changes, and interaction with data and tools, all of which can shift their risk profile over time.

Fine-tuning is a common way to adapt foundation models to specific tasks, but without strict controls, it can introduce new vulnerabilities. Poorly governed tuning can override system prompts, weaken safeguards, or embed sensitive training data into inference responses. In some cases, fine-tuned models become more susceptible to jailbreaks than their base counterparts.

Mitigating this risk requires clear policies around which models can be fine-tuned, how tuning datasets are validated, and how changes are versioned and reviewed. Behavioral checkpoints should be measured against baselines, and testing must include adversarial input to detect increased susceptibility to manipulation. Because tuning often occurs incrementally, drift tends to emerge gradually and requires continuous comparison across versions and environments.

Prompt-based guardrails are often the first line of defense during inference, but they are soft constraints. Instructions embedded in system prompts can be bypassed through indirect phrasing, multi-step reasoning, or prompt injection. Guardrails should not rely on wording alone. They work best when combined with monitoring systems that detect behavior anomalies, validation filters that scan outputs, and red teaming libraries that probe for weaknesses. These mechanisms transform intent into enforceable control.

As models gain access to tools, memory, and downstream systems, guardrails must constrain both actions and responses.

Explainability and integrity are equally critical, particularly in regulated environments. Organizations need to understand why a model behaved a certain way, whether outputs were influenced by recent inputs, and whether configurations or weights have been altered. Trace logs, versioned configurations, and behavioral consistency checks provide the visibility needed to support trust and accountability.

Models may be opaque in how they learn, but their operation should not be. Effective security depends on visibility into both what a model is and how it behaves over time.

Securing the infrastructure

AI-native systems don't operate in isolation. They run in cloud environments, invoke APIs, store data, and communicate across networks. Infrastructure is not just a delivery layer. It directly shapes how models access data, invoke tools, and take action, making it inseparable from overall security posture.

Cloud-native deployments are common, particularly when using hosted LLMs or model-as-a-service APIs. While convenient, these environments often introduce opaque data handling, logging, and retention practices. Teams need to ask what's being stored, where, and by whom. Improper handling of inference data, especially inputs containing personal or proprietary information, can result in compliance failures or unintentional data exposure.

Limited transparency into external controls makes it difficult to assess risk or demonstrate compliance as models interact with sensitive information at runtime.

On-prem and edge deployments offer greater control but also greater responsibility. Teams must secure the full AI stack, including model loading, inference servers, tool integrations, and downstream services. These environments are often less standardized, creating gaps in update management, observability, and isolation. Inconsistent configuration or delayed updates can leave models exposed long after issues are identified.

Strong identity and access controls are critical across all deployment models. Every component, from model APIs to plugin managers and data pipelines, should enforce least privilege. Responsibilities must be clearly separated between systems that supply inputs, invoke models, or act on outputs. As agents gain autonomy, overly broad permissions can quickly turn minor manipulation into system-wide compromise.

Encryption, audit logging, and runtime enforcement complete the picture. Inference activity should be logged, unusual patterns flagged, and access to configurations tightly restricted. A compromised prompt should not be able to escalate privileges or move laterally across connected services. Resilience depends on detecting and containing misuse as it happens, not after the fact.

AI applications blur the boundaries between code, logic, and data. Securing them requires a system-wide view, where each component contributes to the overall posture. AI-SPM enables that approach by making infrastructure observable, enforceable, and aligned with the organization's existing security controls.

SECTION IV.

Monitoring, Governance, and Tools

Monitoring for AI

Security doesn't end at deployment. In AI-native applications, many of the hardest problems emerge after models go live. Inference behavior shifts based on inputs, context, and tool access, requiring monitoring that captures intent, tracks change, and flags behavior that falls outside expectations.

In these environments, monitoring becomes a primary control for understanding how AI systems behave in the real world.

Traditional observability tools offer limited insight. They measure response times and system load but don't reveal how a model interprets a prompt, what decisions it makes, or whether its outputs violate policy. AI-specific monitoring fills this gap by capturing not only requests and responses, but the steps between them.

Trace logs are the foundation of this visibility. They record the full interaction path, including prompts, retrieved context, reasoning steps, tool calls, and outputs. In agent-based systems, traces expose how actions were planned and executed, enabling auditing, investigation, and detection of misuse patterns. Without this context, teams are left reacting to symptoms rather than understanding root causes.

Anomaly detection adds another critical layer. Because models are probabilistic, behavior can drift without any explicit change to code or infrastructure. Monitoring tools must learn what normal behavior looks like and flag deviations, such as unexpected actions, unfamiliar output patterns, or unapproved tool usage. Early detection is especially important in agentic systems, where a single abnormal decision can cascade across services.

Monitoring must also support runtime protection. Some risks only surface in real-world use, such as prompt injection attempts or edge cases missed during validation. Runtime defenses such as output filtering, response sanitization, and automated containment help limit impact and protect connected systems.

When monitoring is paired with enforcement, it becomes a control mechanism rather than a passive observation.

> Effective AI-SPM platforms integrate these capabilities: trace logging, behavior monitoring, threat detection, and response automation into a single feedback loop. This makes it possible to catch issues early, learn from them, and adapt controls without halting innovation.

That feedback loop turns telemetry into action, enabling teams to contain risk and maintain security posture as AI systems evolve.

Governance, compliance, and standards

As AI systems take on greater influence over business operations and user experiences, legal and regulatory scrutiny is increasing. Governance is no longer optional. It is a core requirement for deploying AI responsibly at scale.

For security teams, governance means more than defining policies. It means demonstrating that controls are continuously and effectively enforced.

Frameworks play a central role in this shift. While they don't offer prescriptive rules, they provide structured ways to define expectations, measure risk, and align practices with evolving standards. Their real value lies in translating high-level principles into repeatable, auditable actions across the AI lifecycle.

The NIST AI Risk Management Framework (AI RMF) is one of the most influential examples. Its functions, govern, map, measure, and manage, provide a foundation for integrating risk management into AI development and operations. Although voluntary, the framework increasingly shapes enterprise governance programs and public sector procurement requirements.

On the technical side, the OWASP LLM Top 10 offers a practical lens for security professionals.

It highlights common and high-impact failure modes such as prompt injection, training data poisoning, and insecure output handling. Because it evolves alongside attacker techniques, it serves as a living guide for testing, monitoring, and mitigation.

Existing security and privacy standards are also being extended to address AI-specific risks. ISO 27001 and 27701 are being mapped to AI system design, with an emphasis on data governance, transparency into model behavior, and secure handling of personal information during inference. These efforts reinforce the expectation that AI systems be treated as first-class assets within established security programs.

Regulators are codifying these expectations into law. The EU AI Act classifies AI systems by risk and imposes stricter requirements on high-risk use cases, such as healthcare, hiring, and law enforcement. Similar initiatives are underway in the U.S., U.K., Canada, and Singapore. Across jurisdictions, the common thread is accountability: organizations must be able to show how AI systems are governed, monitored, and controlled throughout their lifecycle.

Compliance is not just a box to check. It is a way to align security practices with business risk and stakeholder trust. AI-SPM supports that alignment by turning governance from a static obligation into an operational capability that enables safe, scalable AI adoption.

SECTION V.

Tools, Tech, and Tactical Approaches

The AI-SPM toolbox

AI Security Posture Management is not a single feature or dashboard. It is a set of capabilities built from specialized tools that bring structure, visibility, and defense to systems operating outside traditional software norms. These tools extend existing AppSec workflows, applying proven security principles to new surfaces such as models, prompts, datasets, and autonomous agents.

What matters most is not individual tools, but how their capabilities are coordinated across the AI lifecycle.

AI vulnerability scanners are often the first tools teams adopt. These scanners analyze model behavior, prompt handling, and configuration to identify risks such as prompt injection, model inversion, or insecure tool execution. Unlike static code scanners, they test behavior dynamically, probing how models respond to adversarial inputs, edge cases, and jailbreak techniques. On their own, however, scan results provide limited value unless they are tied to deployment context, model versions, and potential impact.

Another key capability is the AI Software Bill of Materials (AI SBOM or AI-BOM). Much like a traditional SBOM, the AI-BOM catalogs what's inside an AI system's models, datasets, embeddings, plugins, and toolchain components. This visibility helps teams understand which AI assets are in use, where they originated, and the risks they pose. It is essential in open source and community-driven ecosystems, where third-party risk is often hidden within dependencies. Without continuous updates and linkage to runtime behavior, these inventories quickly become outdated.

Model tracing and observability tools add runtime transparency. They capture prompts, retrieved context, reasoning steps, tool calls, and outputs, enabling teams to investigate anomalies, verify guardrails, and audit agent behavior. In agentic systems, tracing is essential for understanding how decisions translate into actions across connected tools and services.

Data scanning tools complete the picture by validating training and inference inputs. They identify sensitive information, licensing issues, and signs of bias or poisoning. These scanners are most effective when integrated early in the pipeline and when findings feed directly into enforcement and policy decisions rather than isolated reports.

Individually, these tools address different risks. Together, they form the foundation of AI-SPM when combined with shift-left enforcement, continuous monitoring, and consistent policy application across the AI lifecycle.

Red teaming for AI: Your best ally

Security testing for AI-native applications can't rely on legacy approaches. Traditional penetration testing assumes static code paths, known interfaces, and repeatable logic. Large language models and AI agents behave differently. They interpret prompts in real time, respond probabilistically, and change behavior based on fine-tuning, memory, and tool access.

Because risk emerges through interaction and adaptation, snapshot testing falls short.

Red teaming addresses this gap by simulating real attacks. Instead of probing ports or injecting SQL, AI red teams craft structured prompts designed to bypass safeguards, leak information, or trigger unintended actions. These attacks are not theoretical. Real incidents have involved agents executing fraudulent transactions, chatbots exposing sensitive logs, and models ignoring safety controls due to subtle prompt manipulation.

These failures occur because attackers exploit how models reason and act, not because of traditional software flaws.

Red teaming often begins with fuzzing, which generates prompt variations to probe behavioral boundaries. Indirect phrasing, encoded tokens, or multi-step reasoning can expose weaknesses that static testing cannot, especially when paired with structured attack libraries that capture known jailbreak and evasion techniques.

Tools like LLM red teaming go a step further. They execute these prompts across models, trace responses, and evaluate the likelihood of success. Tests can be configured

by goal, such as data exfiltration, content bypass, or injection into downstream systems, and run repeatedly to evaluate the stability of the exploit.

This repeatability is essential for detecting regression, drift, and the reintroduction of previously mitigated risks.

Beyond identifying vulnerabilities, red teaming clarifies impact. It answers practical questions about what data can be accessed, what actions can be triggered, and how far an exploit can propagate. By linking behavior to consequence, teams can prioritize remediation based on real business risk.

AI red teaming is not a one-time exercise. Findings should feed directly into training updates, policy changes, and model selection. As AI systems evolve, tests must evolve with them. When integrated into development and deployment workflows, red teaming becomes a continuous control rather than a periodic assessment.

Red teaming is not a fallback. It is one of the most effective ways to understand how AI systems behave under pressure and to address risks before they reach production.

Embracing visibility over vagueness

AI-native applications are redefining how software is built and secured. They are dynamic, adaptive, and composed of elements that don't exist in traditional codebases: models, prompts, embeddings, memory chains, agents, and plugins. These components don't just add complexity. They introduce new categories of risk.

That risk doesn't come from any single element, but from how these systems behave and evolve over time.

Securing AI requires a shift in mindset. Traditional tools layered on top of AI systems can't keep up with components that change continuously and reason in real time. Organizations need visibility into what their AI systems contain, how they behave, and where they connect. That visibility is the foundation of AI Security Posture Management.

Without it, security teams are left making assumptions about systems that change faster than manual review or documentation can track.

AI-SPM is not about slowing innovation. It provides a structured, scalable way to manage AI risk while supporting existing workflows and adapting to the non-deterministic nature of LLM-driven systems. Most importantly, it shifts security from a reactive function to a continuously executed capability.

Whether you're experimenting with generative AI or running models in production, the principle is the same: know what you're running, observe how it behaves, and test it as an attacker would. Secure systems aren't just well built. They're well observed, governed, and maintained.

Visibility enables all three.

Perfect answers aren't the goal, but guessing shouldn't be either.

Start with visibility. The rest follows from there.

Book a demo with Snyk to secure your AI systems.