



WHITEPAPER

WHAT'S LURKING IN YOUR AI?

A deep dive into AI Security Posture Management (AISPM)

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



Section I.

FOUNDATIONS OF AISPM

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 (AISPM) comes in.

AISPM 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.

AISPM works alongside tools like SAST, DAST, SCA, and ASPM, adding new lenses that expose risks in models, logic flows, and contextual inputs. It integrates into existing workflows, expanding your visibility into previously invisible or unstable components.

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.

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. AISPM 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.

AISPM 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. They're 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.



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

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

While it's useful to explore how AI can defend us, it's essential to first ensure the AI we deploy is defensible. AISPM brings that defensibility within reach by giving teams the tools, data, and practices needed to build secure, explainable, and governable AI-native applications.

THE AI THREAT LANDSCAPE

What makes AI a unique security challenge?

Traditional applications behave predictably. They follow structured logic, coded explicitly by humans. If a developer writes a rule to validate an email address, that rule executes the same way every time. Security reviews, testing tools, and policy enforcement mechanisms are built on this expectation. AI-native systems operate differently.

Large language models (LLMs) are probabilistic. Their responses vary based on input phrasing, context history, and subtle variations in data. Prompting an AI with identical inputs five times can produce five different outputs. Worse, small changes in the model's weight, training data, or surrounding tools can cause unexpected shifts in behavior, some of which have serious security implications.

Attack surfaces shift rapidly in these environments. In traditional applications, the risk is mostly at the boundaries: exposed APIs, unauthenticated endpoints, and misconfigured infrastructure. In AI applications, the attack surface includes prompts, data pipelines, inference chains, memory states, agent instructions, and more. Each new integration, whether a plugin, tool, or retrieval API, adds potential entry points for misuse.

Much of this behavior is opaque. AI models are often treated as black boxes, with limited visibility into how they arrive at conclusions or decisions. Part of this opaqueness is AI's non-determinism, where it makes different decisions at different times, generating unique output each time. This makes traditional static or dynamic testing unreliable. It also creates gaps in coverage for teams trying to model threats, enforce security policies, or explain behavior.

In AI systems, actions aren't always traceable to clear code paths; they may emerge from inferred relationships between services, tools, or memory states.

This makes enforcement hard: you're building policy on "maybes," not determinism. Logs, traces, and adversarial testing, not static code review, become the only reliable lens.

Finally, data flows are difficult to predict. User prompts can become training inputs. Intermediate results from agents may leak into logs or memory. Plugins connected to the model may access unintended endpoints. These fluid, bidirectional flows defy traditional data classification and tracking tools. As a result, information leakage, model abuse, and unexpected system interactions are increasingly common.

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 begins with understanding the distinct threats that target these systems. Many of these vulnerabilities are tied to the way models are trained, prompted, and deployed, not just the infrastructure around them.

➔ Data Poisoning

Attackers inject malicious examples into a model's training data to alter its behavior during inference. A poisoned dataset might train the model to always respond with a specific string to certain prompts or suppress key outputs in adversarial scenarios. This threat is especially dangerous in open training pipelines or systems that fine-tune on real-world data like product reviews or user content.

➔ Adversarial Input and Model Evasion

These techniques exploit how models interpret input by subtly altering it using misspellings, encoded tokens, or vague phrasing to slip past safety filters and trigger restricted behaviors. Attackers use this method to coax harmful or disallowed responses from otherwise locked-down models.

➔ Model inversion and extraction attacks

Attackers can reconstruct sensitive data used during training by repeatedly querying a model. These attacks may expose personal details, proprietary code, or confidential documents, especially in models trained on private or sensitive datasets, by effectively reversing the model's learned representations.

➔ Membership inference

This tactic allows attackers to determine whether a specific data point was included in a model's training set. In medical or financial data cases, confirming that a person's information was used could violate privacy laws and expose sensitive operational details.

➔ Model theft

Through high-volume querying, attackers can clone a model's functionality and recreate it under a different brand or wrapper. This undermines the original developer's intellectual property and can lead to misuse of the stolen model, including embedding malicious behavior or deploying it without safety controls.

➔ Infrastructure attacks

AI systems are often surrounded by APIs, plugin frameworks, and deployment pipelines, all of which reintroduce classic security risks. If a model output is blindly passed to a tool or endpoint, vulnerabilities like SSRF, XSS, or Command Injection can resurface, turning LLM agents into attack vectors.

➔ Supply chain manipulation

Many AI applications depend on shared resources like open source libraries, pre-trained models, and public datasets. If any component, like a model card on Hugging Face or a plugin dependency, is compromised, it can silently introduce security issues that scale across systems. These assets often go untracked by traditional SBOMs, leaving organizations blind to potential risks.

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.

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 particularly interested in exploiting the strategic advantages of AI. Generative models embedded in commercial and industrial applications represent valuable targets for intelligence gathering and sabotage. Nation-states have the resources to reverse-engineer proprietary models, inject manipulated data into public datasets, or test prompt injection techniques at scale. Their focus often goes beyond stealing data. They aim to degrade trust, manipulate decision-making, or undermine critical infrastructure that now relies on AI-assisted logic.

Cybercriminal groups see generative AI as both a target and a tool. Some attempt to hijack AI agents and reroute them toward malicious actions, like manipulating a customer support chatbot to exfiltrate sensitive information. Others look for indirect wins, like extracting training data that contains credit card numbers, passwords, or confidential business content. There are also monetization schemes emerging around cloning paid APIs or selling jailbreak instructions for popular models.

Corporate competitors, whether operating legally or not, can pose risks through espionage or model cloning. A competitor might attempt to infer your training data, reverse-engineer your model outputs, or seed a poisoning campaign into the same public corpus you're using to fine-tune. This isn't limited to large players. Smaller startups may unwittingly or intentionally pull data, prompts, or architectures from publicly accessible models and wrap them into their own offerings.

Hactivists and ideologically driven attackers have found new ground in generative AI systems. Some look for ways to bypass content moderation or expose biased outputs as a form of protest. Others manipulate models to produce misleading or inflammatory results and then publicize those outputs to drive public distrust. The open nature of AI discourse, combined with the fragility of prompt-based controls, makes these attacks easy to conduct and difficult to contain.

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.



What's at stake?

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

One large language model integrated into a public-facing tool began leaking training data after repeated adversarial prompts. A journalist was able to recover internal Slack messages, employee credentials, and production logs embedded in the model's context window. The root cause? The model had been fine-tuned on a dump of internal chat data with no content filtering.

In another case, an autonomous AI agent connected to a procurement system was tricked into placing a fraudulent order by manipulating the prompt structure and bypassing a plugin validation step. The attacker didn't exploit a buffer overflow or injection flaw, they simply understood how the model parsed goals and took action through integrated APIs.

These failures erode trust quickly. A model that leaks sensitive data or hallucinates in critical environments can damage reputations in minutes. Legal exposure follows closely behind. Regulators are beginning to focus on model accountability, explainability, and safety, especially in healthcare, finance, and government contexts. An AI system that exposes personally identifiable information or makes a biased hiring recommendation can trigger both compliance audits and litigation.

There's also growing concern about physical risks. AI agents connected to IoT systems, manufacturing lines, or autonomous vehicles must be secured against unpredictable behavior. A manipulated prompt that overrides a sensor reading or reorders system logic can cause real-world harm, not just workflow disruption.

Beyond specific incidents, the broader consequence of weak AI security is the loss of control. When organizations don't know what models are running, how they were trained, or what they're connected to, they have no meaningful way to assess exposure or respond effectively. That gap creates space for attackers to exploit, and for decision-makers to lose confidence in their AI investments.

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 AISPM LIFECYCLE PLAYBOOK

From DevSecOps to AI SecOps

DevSecOps transformed how organizations build and secure software. By embedding security into every phase of the development lifecycle, from design to deployment, it shortened feedback loops, improved coverage, and brought developers into the security conversation early. But as applications become AI-native, the assumptions behind that model start to break down.

Traditional DevSecOps pipelines are built around deterministic code. They scan for known vulnerabilities in libraries, analyze static logic for unsafe patterns, and test API behavior through defined routes and schemas. AI development doesn't follow those patterns. It involves datasets, weights, hyperparameters, prompts, embeddings, and memory states. Most of these don't live in the same repositories or build systems as application code. They change independently. They also introduce new risks that security tools weren't designed to detect.

AI SecOps adapts the principles of DevSecOps to fit this new architecture. It embeds security practices not just in source control and CI/CD, but across the entire AI lifecycle, including data collection, fine-tuning, model orchestration, and prompt design. It also shifts the culture: security teams are encouraged to partner with AI engineers, data scientists, and platform teams. The shared goal is to secure the full AI system, not just the wrapper around it.

AI SecOps supports tighter alignment between model behavior and business risk. It requires guardrails that work in probabilistic environments. It favors testing strategies that simulate real threats, not just theoretical ones. And it starts with visibility because you can't defend what you can't map.



Securing the AI/ML lifecycle (AISecOps)

The machine learning lifecycle includes more than training and deployment. Each phase introduces unique security challenges, and every handoff between teams becomes a potential source of exposure. Securing this lifecycle means understanding where risks originate and how to mitigate them early before they embed themselves in production systems.

Data collection

This is often the first point of exposure. If attackers can influence the dataset, especially in open or user-submitted environments, they can inject poisoned examples or subtly bias the model's behavior. Even well-meaning datasets can be problematic if they include unlicensed content, embedded secrets, or sensitive personal information. Secure collection demands strong initial input validation, content filtering, and robust lineage tracking.

Training

This is the phase where models learn and where malicious patterns can be introduced. Poisoned, skewed, or imbalanced data can push models toward misleading or harmful responses. Weak fine-tuning practices can override intended constraints or introduce leakage of system prompts. Solid MLSecOps procedures call for checkpoint validation, reproducibility checks, and red teaming of fine-tuned models before deployment.

Validation

This isn't just about measuring accuracy, it's about identifying misuse potential. Effective validation includes adversarial testing like jailbreak attempts, prompt fuzzing, and stress-testing for unintended behaviors. It must also assess bias, overfitting, and vulnerability to membership inference. Validation datasets should reflect edge cases and potential abuse scenarios, not just happy paths.

Deployment

Going live is more than shipping a model. It means integrating with APIs, tools, memory systems, and user inputs, all of which expand the attack surface. Secure deployment demands access controls, real-time monitoring, version tracking, and detecting and quarantining compromised models. The risk becomes immediate if a model generates unsafe outputs or takes unintended actions. Quarantining contains the damage before it spreads across systems and users.

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.

Securing the data pipeline

Data pipelines power the intelligence of AI systems. They feed models the context they rely on to reason, recommend, or act. But these pipelines also carry some of the highest security risk because the data they process often comes from untrusted, third-party, or user-generated sources.

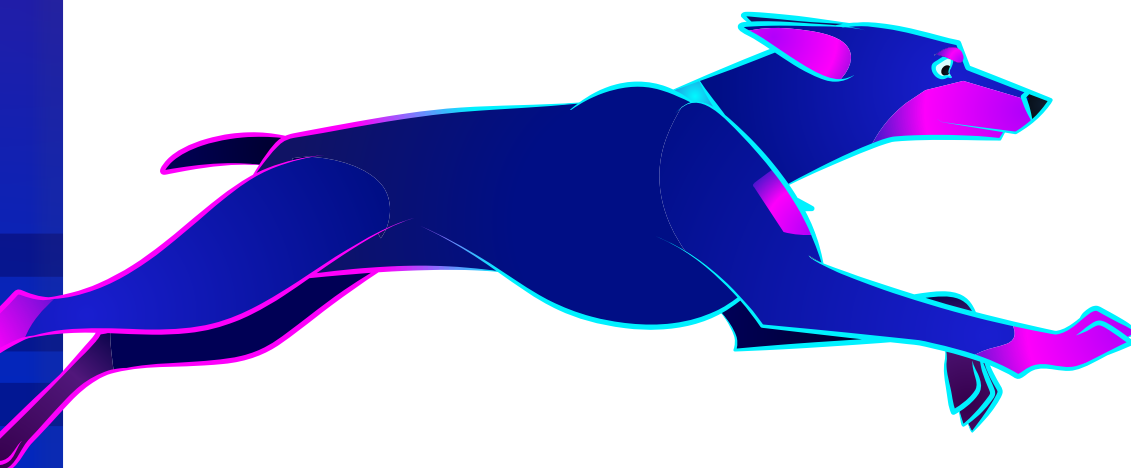
The risks differ between training data and inference data, but both are critical. During training, bad data can alter a model's weights or introduce backdoors. During inference, malicious inputs can trigger unintended behavior, extract hidden information, or influence downstream systems. And because AI systems often blend these phases, fine-tuning on recent interactions, for example, the lines between them blur.

Bias, poisoning, and leakage are all common threats. Bias can creep in when datasets overrepresent or underrepresent specific groups or behaviors. Poisoning introduces malicious data designed to change a model's logic or response patterns.

Leakage happens when models memorize sensitive inputs and later reveal them, either directly or through inversion techniques.

Addressing these risks requires more than data cleaning. It demands data governance at every stage. That means tracking provenance, knowing where the data came from, how it was processed, and who had access to it. This means applying controls that help limit the types of content that can influence model outputs. And it means auditing what's being retained or cached during inference, especially in agent-based systems with memory components.

Without strong governance, AI systems can become unpredictable or unsafe not because of malicious code, but because of data drift, unfiltered inputs, or flawed assumptions baked into the model's context. AISPM brings clarity to this complexity by treating data flows as security-critical assets not afterthoughts.



Securing the model

Once a model is trained, it becomes the core of an AI-native application but the work doesn't stop there. Securing a model requires control over how it's updated, how it behaves, and how it's interpreted.

Fine-tuning is one of the most common practices in adapting foundation models to specific tasks or domains. However, without strict controls, it can introduce new vulnerabilities. Poorly scoped fine-tuning sessions can override system prompts, remove protective behaviors, or embed sensitive training data into inference responses. In some cases, fine-tuned models have become more susceptible to jailbreaks than their base counterparts.

To reduce this risk, it's important for organizations to define policies around which models can be fine-tuned, how tuning datasets are validated, and how changes are versioned and reviewed. Internal checkpoints should be compared to baselines for behavioral drift. Testing should include adversarial input to detect increased susceptibility to prompt manipulation.

Prompt-based guardrails are often the first line of defense during inference. Instructions like "don't generate harmful content" or "only answer in JSON format" are added to the model's system prompt to shape behavior. But these are soft constraints, not hard rules. Without layered safeguards, they can be bypassed through indirect phrasing, multi-step logic, 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.

Model explainability and integrity are also essential, especially in regulated industries. Organizations need to know why a model responded a certain way, whether the output was influenced by recent inputs, and whether the model has been tampered with. This is particularly important for agentic systems that chain reasoning across multiple prompts, tools, or memory contexts.

Explainability can be supported through structured trace logs, deterministic configuration snapshots, and consistent version control. Integrity can be monitored through hash-based validation of weights, audits of configuration changes, and tests for behavioral consistency across environments.

Models may be opaque in how they learn, but their operation shouldn't be. Effective security starts with visibility into both what the model is and what it's doing.

Securing the infrastructure

AI-native systems don't operate in isolation. They run in cloud environments, invoke APIs, store data, and communicate across networks. Every part of that infrastructure plays a role in defining the system's risk surface.

Cloud-native deployments are common, especially when using hosted LLMs or model-as-a-service APIs. These setups are convenient, but they often involve external data handling, logging, and memory retention policies that aren't fully transparent.

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.

Deploying models in on-prem or edge environments offers more control, but also more responsibility. Teams are responsible for managing the full AI stack: model loading, inference servers, tool integrations, and downstream systems. These setups are often less standardized, which can create security gaps, particularly around update mechanisms, observability, and isolation of model components.

Infrastructure security for AI also involves strict identity and access controls. Every component, whether a model API, plugin manager, or data pipeline, should enforce least privilege. This includes separating responsibilities between systems that provide inputs, invoke models, or act on outputs. AI agents should never have unrestricted access to critical services or user data unless explicitly required.

Encryption, audit logging, and runtime policies are also critical. Systems should log every inference call, flag unusual input patterns, and restrict access to model configurations. In edge cases, a compromised prompt could attempt to escalate privileges by triggering tool use or lateral movement across connected APIs. The infrastructure needs to be resilient against those possibilities.

AI applications blur the lines between code, logic, and data. Securing them requires thinking beyond any one layer. Each component, from the model to the container it runs in, plays a part in the system's overall posture. AISPM makes that system observable, enforceable, and aligned with the security controls already in place across the organization.

MONITORING, GOVERNANCE, AND TOOLS

Monitoring for AI

Security doesn't end at deployment. For AI-native applications, the real challenges often begin after the model goes live. Inference behavior changes based on inputs, context, and downstream tool access. That variability demands a different level of monitoring, one that can capture intent, track changes, and alert teams when behavior strays from expectations.

Traditional observability tools offer limited insight into how models operate. They capture metrics like response times or system load but don't expose how the model interpreted a prompt, what decisions it made, or whether the output violated policy. AI-specific monitoring adds that missing layer by logging not just the request and response, but the sequence of steps between them.

Trace logs are the foundation of this visibility. A trace captures the entire interaction path, input prompt, retrieved context, reasoning chain, tool calls, and final output. For agent-based systems, it includes the logic used to plan actions and the intermediate results at each stage. These traces give security teams a way to audit decisions, investigate anomalies, and spot potential misuse patterns over time.

Beyond logs, anomaly detection is key. Since models are probabilistic, behavior can shift without any explicit change to code or infrastructure. Monitoring tools should learn what typical responses look like, then flag outliers. That could include outputs that mention unexpected entities, perform unapproved actions, or respond in an unfamiliar tone or format. These signals help detect jailbreaks, model drift, or unauthorized access to embedded tools and data.

Monitoring should also cover post-deployment protections. Some risks don't emerge until models are exposed to real users. For example, inference inputs might contain prompt injection attempts or trigger edge-case behavior not caught during validation. Runtime defenses like response filters, output sanitizers, and automated remediation rules are needed to contain the impact and protect connected systems.

Effective AISPM 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.

Governance, compliance, and standards

As AI systems gain influence over business operations and user experiences, they also face increasing legal and regulatory scrutiny. Governance is becoming an essential part of deploying AI responsibly at scale.

Security teams are turning to formal frameworks to guide their approach. These frameworks don't offer one-size-fits-all rules, but they do provide a structured way to define expectations, measure risk, and align with evolving global standards.

The NIST AI Risk Management Framework (AI RMF) is one of the most comprehensive efforts to date. It outlines a set of functions that govern, map, measure, and manage, which help organizations integrate risk mitigation across the AI lifecycle. While NIST's framework is voluntary, it's influencing both public sector procurement and enterprise-level governance strategies.

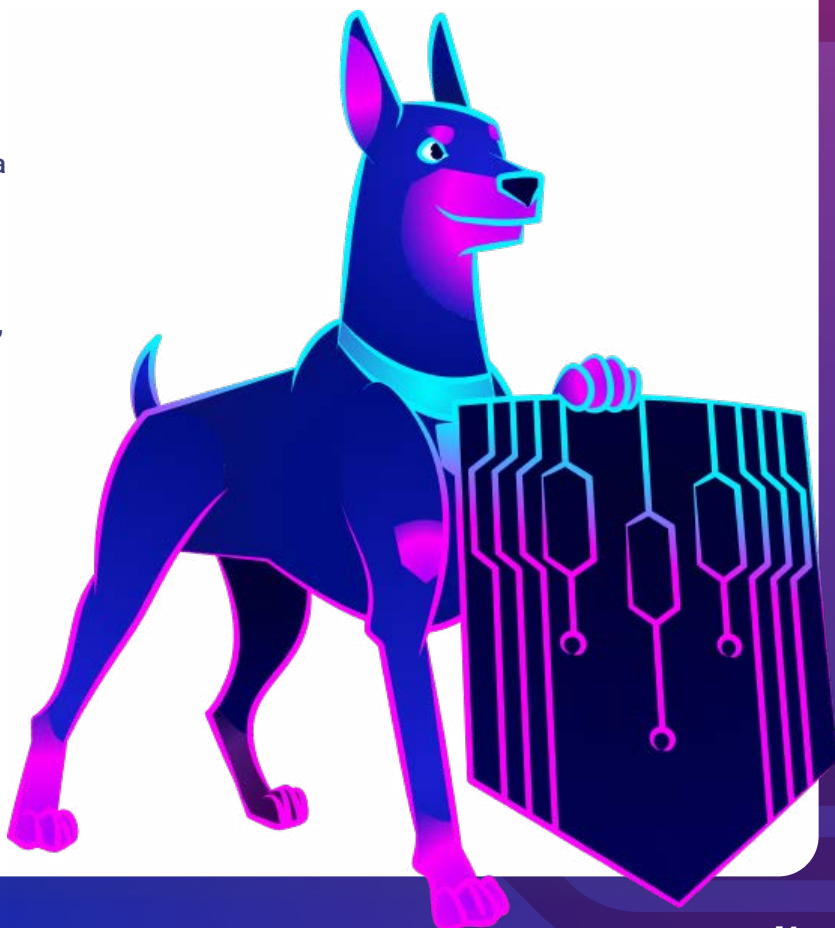
On the technical side, the OWASP LLM Top 10 offers a practical lens for security professionals.

It identifies the most common and impactful vulnerabilities associated with large language models, including prompt injection, training data poisoning, and insecure output handling. The list is updated as attack techniques evolve, making it a living reference for red teams and engineering leads alike.

Existing standards are also being extended to account for AI-specific concerns. ISO 27001 and 27701, which focus on information security and privacy management, are now being mapped to AI system design. These extensions emphasize control over model training data, transparency into model logic, and secure handling of personal information during inference.

Regulatory bodies are codifying these expectations into law. The EU AI Act, one of the most developed legislative efforts so far, classifies AI applications by risk level and assigns requirements accordingly. High-risk applications like those in healthcare, law enforcement, or hiring must meet strict criteria for documentation, traceability, robustness, and human oversight. Other countries are moving in similar directions, with draft laws or policy guidance already in progress in the U.S., U.K., Canada, and Singapore.

Staying compliant isn't just a box to check.
It's a way to align security efforts with broader business risk and stakeholder trust.



Section V.

TOOLS, TECH, AND TACTICAL APPROACHES

The AISPM toolbox

AI Security Posture Management is not a single feature or dashboard. It's a capability set built from specialized tools designed to bring structure, visibility, and defense to systems that operate outside traditional software norms. These tools don't replace existing AppSec workflows. They extend them, applying tested principles to new surfaces like models, prompts, datasets, and autonomous agents.

The first category to gain traction is AI vulnerability scanners. These tools examine model behavior, prompt responses, and configuration settings to identify risks such as prompt injection, model inversion, or insecure tool execution. Unlike static scanners for code, these systems must test behavior dynamically, probing how a model responds to edge-case inputs, adversarial sequences, or subtle jailbreaks.

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.

A well-structured AI-BOM gives AppSec and governance teams visibility into what AI assets they're running, where those assets came from, and what risks they may carry. This is especially critical for open source models and community-driven plugin ecosystems, where third-party risk is often hidden in dependencies.

Model tracing and observability tools create transparency into how LLMs and agents make decisions. These tools capture input/output sequences, reasoning steps, and contextual memory during runtime. They help security teams debug anomalies, track usage, and verify whether output stayed within expected guardrails. In agentic systems, tracing becomes even more important, allowing teams to audit how the model selected tools, handled intermediate results, and interpreted prompts.

Finally, data scanning tools help validate the inputs and training materials that power AI behavior. They flag personally identifiable information (PII), licensing violations, and indicators of bias or poisoning. Data scanners work best when integrated early in the pipeline before data is fine-tuned into a model or fed into context windows during inference.

While the tool categories above serve different purposes, the strongest AISPM platforms bring them together with common features: shift-left enforcement, continuous monitoring, secure policy application,

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 don't operate that way. They interpret prompts in real time, generate responses probabilistically, and change behavior based on fine-tuning, memory, and tool access.

Red teaming for AI solves this by simulating real attacks. Instead of probing ports or injecting SQL, red teams for LLMs craft structured prompts designed to bypass filters, leak information, or trigger unintended actions. These scenarios aren't speculative. Successful attacks have involved agents booking \$0 transactions, chatbots disclosing sensitive logs, and models refusing to follow safety instructions due to subtle prompt phrasing.

Red teaming starts with fuzzing, which generates variations of prompts to test the boundaries of model behavior. These prompts might introduce oblique language, foreign characters, or chain-of-thought logic designed to confuse guardrails. Fuzzing is especially effective when paired with structured attack libraries, which encode known jailbreaks, evasions, and manipulation tactics.

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 stability of the exploit.

This simulation-based approach is critical for identifying not only direct vulnerabilities but also the potential blast radius of a successful attack. If a model leaks data, how sensitive is it? If an agent runs a command, what can it access? Red teaming connects behavior to consequence and helps prioritize remediation based on business impact.

AI red teaming doesn't end with a report. Findings should feed into training updates, policy adjustments, and improved model selection. As models evolve, so must the tests. Effective teams automate key parts of this work, incorporate findings into CI/CD security gates, and repeat tests as prompts, models, or tools change.

Red teaming isn't just a fallback. It's a key practice for understanding risk. It gives organizations the clearest window into how AI might behave under pressure, and the best chance to fix issues before they appear in production.

Embracing visibility over vagueness

AI-native applications are redefining what it means to build, deploy, and secure software. They're dynamic, unpredictable, and composed of components that don't exist in traditional codebases models, prompts, embeddings, memory chains, agents, and plugins. These elements don't just add complexity. They create entirely new categories of risk.

Securing them requires a shift in mindset. It's not enough to bolt on traditional tools and hope they catch what matters. Organizations need visibility into every part of their AI systems, how they're built, what they contain, how they behave, and where they might be exposed. That visibility is the foundation of AI Security Posture Management.

AISPM isn't about slowing down innovation. It's about making sure innovation doesn't outpace safety. It provides a structured, testable, and scalable way to manage AI risk, one that supports how teams already work and adapts to the unique nature of LLM-driven systems.

Whether you're just starting to explore generative AI or already deploying models in production, the same principles apply: know what you're running, track how it behaves, and test it like an attacker would. Secure systems aren't just well-built, they're well-observed, well-governed, and well-maintained.

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.

