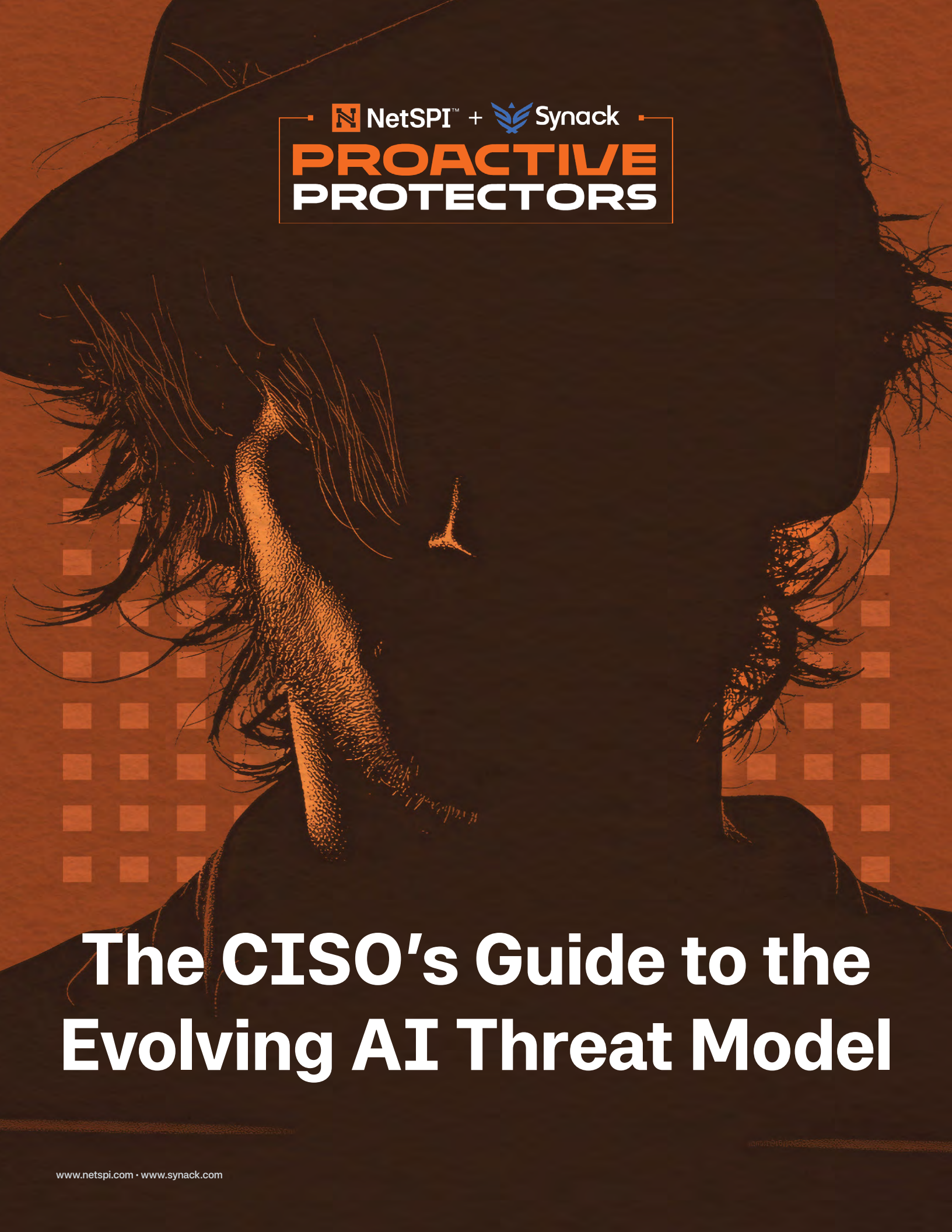


The CISO's Guide to the Evolving AI Threat Model

www.netspi.com · www.synack.com

NetSPI™ + Synack

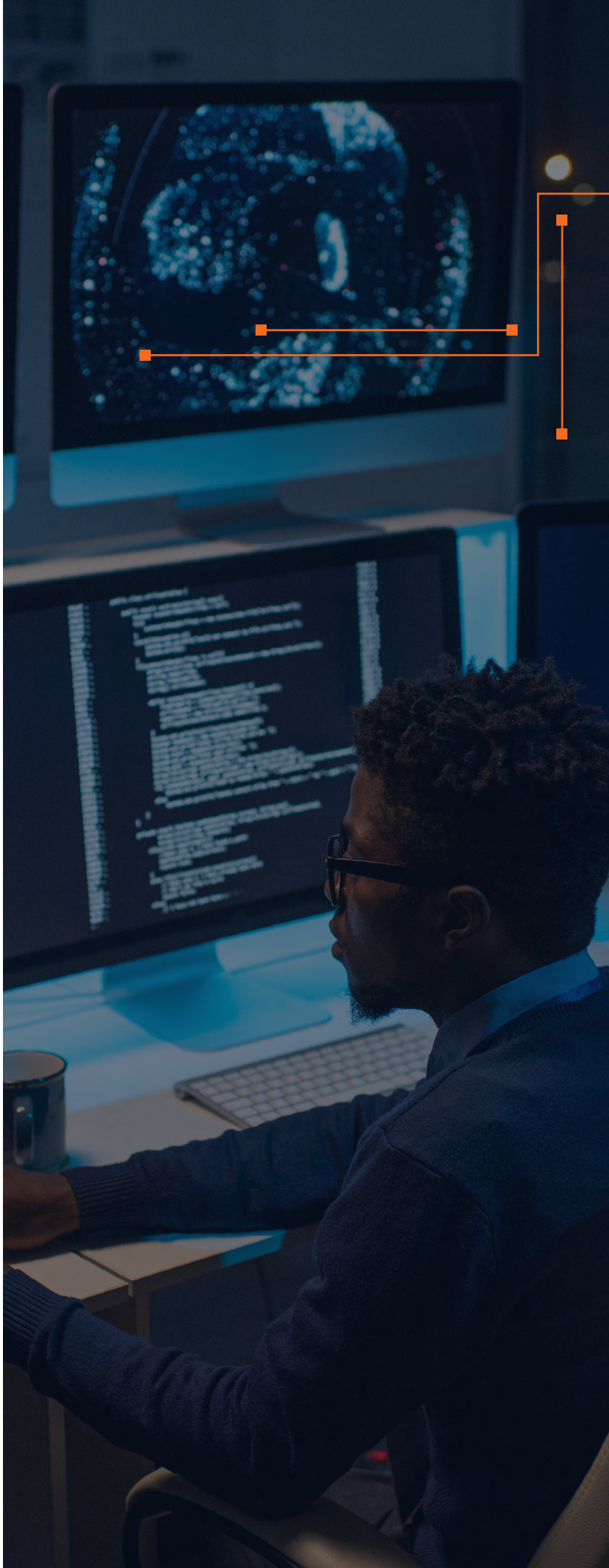
A Quick Experiment in AI Risk

Pull up your risk register from Q1 2024 and read the top ten entries. Now think about how many employees pasted something into a chatbot this morning. The register took a full quarter to write. The gap you just noticed took thirty seconds.

The register wasn't wrong when you built it. It rested on a reasonable premise: risk enters an organization through established channels, a new vendor through procurement, and a new system through architecture review. That premise held for twenty years, because the cost and friction of new technology gave governance time to catch up before usage scaled. It assumed risk would announce itself. In 2026, for most AI projects, it doesn't.

THIS EBOOK COVERS:

1. The AI Problem Has Evolved
2. How You're Managing AI Today
3. How Attackers Are Coming for Your AI
4. How You Can Secure It With The NetSPI Approach



NetSPI™ + Synack

The Problem Has Evolved

Why AI Risk Is Different Now

AI adoption broke that premise because the technology arrived faster than the approval process built to catch it. Nobody purchased ChatGPT for finance, and nobody submitted a vendor risk assessment before product turned on Claude inside a tool already licensed.

Security teams have built their perimeter, data classifications, and threat models around scenarios based on a human actor, but AI projects break that paradigm because of the speed of operations to identify and exploit vulnerabilities and disrupt operations that a human cannot match.

Take shadow AI, the new shadow IT. Data doesn't leave through a large, suspicious file transfer that your DLP was built to catch. Instead, it leaves in a prompt to a model that has been given broad access to sensitive data that was supposed to help it respond better, all that is a risk that you can at least imagine measuring: web traffic gates and IP addresses can be throttled, monitored, blocked.

AI vendor risk is harder. Every major SaaS product added AI features between 2025 and today, while most vendor risk programs still run on annual self-attestation, but your actual exposure has changed hundreds of times since Procurement's last annual review.

Underneath both issues sits the evolving threat: long-lived service accounts feeding known applications used to be worrisome, but now AI agents hold credentials, choose their own tools, and act outside any inventory list that you've kept current. The recent attacks on Hugging Face by agents that overcame system boundaries to achieve their stated objective highlights how defined goals can supersede the authority that they are supposed to have. Non-human identities already outnumber human ones by a wide multiple in most enterprises, and unlike a service account, an agent can rewrite its own instructions and will answer to whomever prompt-injected it last. From a certain point of view, AI agents have become the ultimate insider risk.

Why Hosting AI Introduces New Challenges

The obvious move is to host the model yourself and keep data inside your own walls. Do that, and a self-hosted model can quietly sidestep every control that you've been building for the past 10 years. It bypasses your SaaS security stack because it isn't SaaS, it bypasses vendor governance because there's no vendor, and your DLP has never been charged with evaluating inference traffic before. Here, you've gained control of the model (and probably lowered costs), but lost critical visibility about risk, and no one framed that trade off as something that you were consciously making.

None of this means your team got worse at security. However, every instrument traditional risk management relies on was calibrated for a world where risk arrives in discrete, reviewable events: a new vendor, system, or hire. AI adoption arrives continuously, through every browser and hidden inside every vendor's release notes, but a checkpoint-centric security program can't gate something with no single point of entry. Today's Security Risk Manager's job has become an exercise in systemic coordination, because every stakeholder is now asking a different question, at the same time, continuously.

Why a Systemic Approach to AI Is Necessary

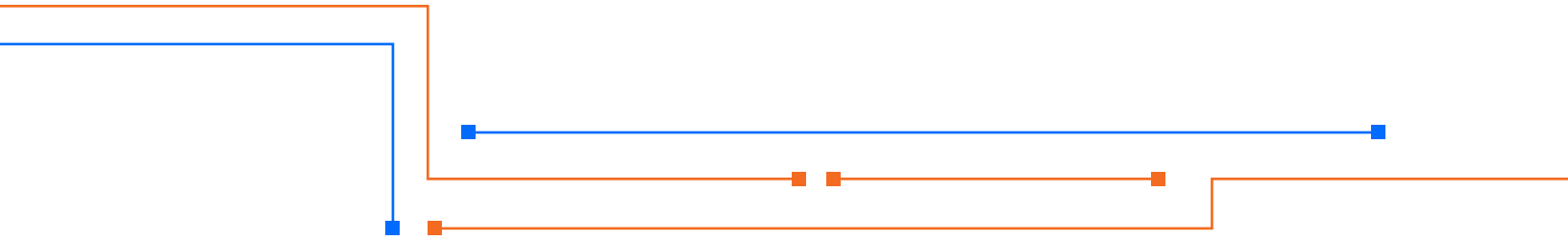
Nobody's job changed, but everybody's questions about what matters did:

Organizational Function	The Checkpoint-Era Question	The Coordination-Era Question
Data owners	What application uses my data?	Can AI retrieve, infer, summarize, or recombine my data in ways I never authorized?
Privacy	Is sensitive data collected and processed lawfully?	Can sensitive data surface in prompts, embeddings, outputs, logs, or a third-party model's context?
Infrastructure	Is the system available and resilient?	Are model endpoints, vector stores, orchestration layers, and tool integrations resilient and observable?
DevSecOps	Is the code scanned and deployed securely?	Are prompts, model configs, agent permissions, and retrieval logic versioned and tested like code?
Legal	What liability does this software create?	Who is accountable when the system produces harmful, biased, or contractually problematic output?
SOC	What attack patterns should we detect?	Can analysts recognize indirect prompt injection, agent misuse, and abnormal model-tool behavior?
Incident response	Was there a breach or an outage?	Was there manipulated retrieval, unauthorized tool use, or unsafe automation, and can we replay it?
Governance	Are controls documented?	Can we prove which AI use cases exist, what data they touch, and who approved the risk?
Audit	Did the process follow policy?	Can we trace prompts, outputs, model versions, retrieval sources, and approvals end to end?

Every question above is ultimately about what's visible and what's hidden, but visibility is only half the problem.

These systems no longer just answer prompts; they act on them. A model that hallucinates a summary is embarrassing, but recoverable. An agent that hallucinates a refund, files a change request, or approves a transaction based on a poisoned document costs real customer trust that is very difficult to get back.

Since ChatGPT's release, AI testing has mostly focused on catching a model that says something strange. Today that model instruct agents who have access to destructive tools that operate at machine speed, expose protected data, take an unauthorized actions, and produce a result that you can't defend in an audit or courtroom. That's the business outcome missing from your risk register: fraud losses, unauthorized transactions, IP leaking through a RAG pipeline, and a decision no one can reconstruct six months later when a regulator asks how it happened.



The question your testing program has to answer has changed too. In 2024, it was whether the model could be manipulated. That still matters, but despite the attention that we had on models, the model was never what made decisions on your behalf, the workflow was. The real question is whether that workflow, the models that it employs, the context retrieval that it uses, and the tools that it adopts to meet its goal can be combined with the human oversight to be trusted under real adversarial pressure, and whether you'd know if it couldn't.

You don't get that confidence from a policy or a single checkpoint. Instead, you get it by deliberately attacking the full system, by stressing its buffers, and by focusing on the seams between systems and people before someone else does.

Some security teams are already adapting to this new approach. They've added AI-specific gates across the full SDLC, deployment and decommissioning included. They've extended red teaming past looking at just the model and into its harness, the orchestration layer, and the tools that its agents can call. Regarding vendor AI risk, a few are asking for an AI equivalent of a software bill of materials, an "AI-SBOM," so a silent model update doesn't go unnoticed.

What's so disruptive today isn't any single security practice. It's that organizations are trying to answer all nine of the questions above at once with teams that have never shared such a distributed risk workflow before.

Historically, because of how we subdivided the software development lifecycle, best practices were isolated to be inside a single team's boundary of authority, but the failures we see in AI implementations today don't follow that pattern. Instead, a chained exploit moves through the seam between the teams that built the retrieval pipeline and the teams that scoped the agent's permissions, and neither was in the room when the other decided how to move forward.

This isn't theoretical. It's been demonstrated repeatedly, in research and production incidents, and the pattern holds: each piece of work passed its own review, but the chain of risk transferred between teams never got reviewed by anyone, because no one owned the chain.



NetSPI™ + Synack

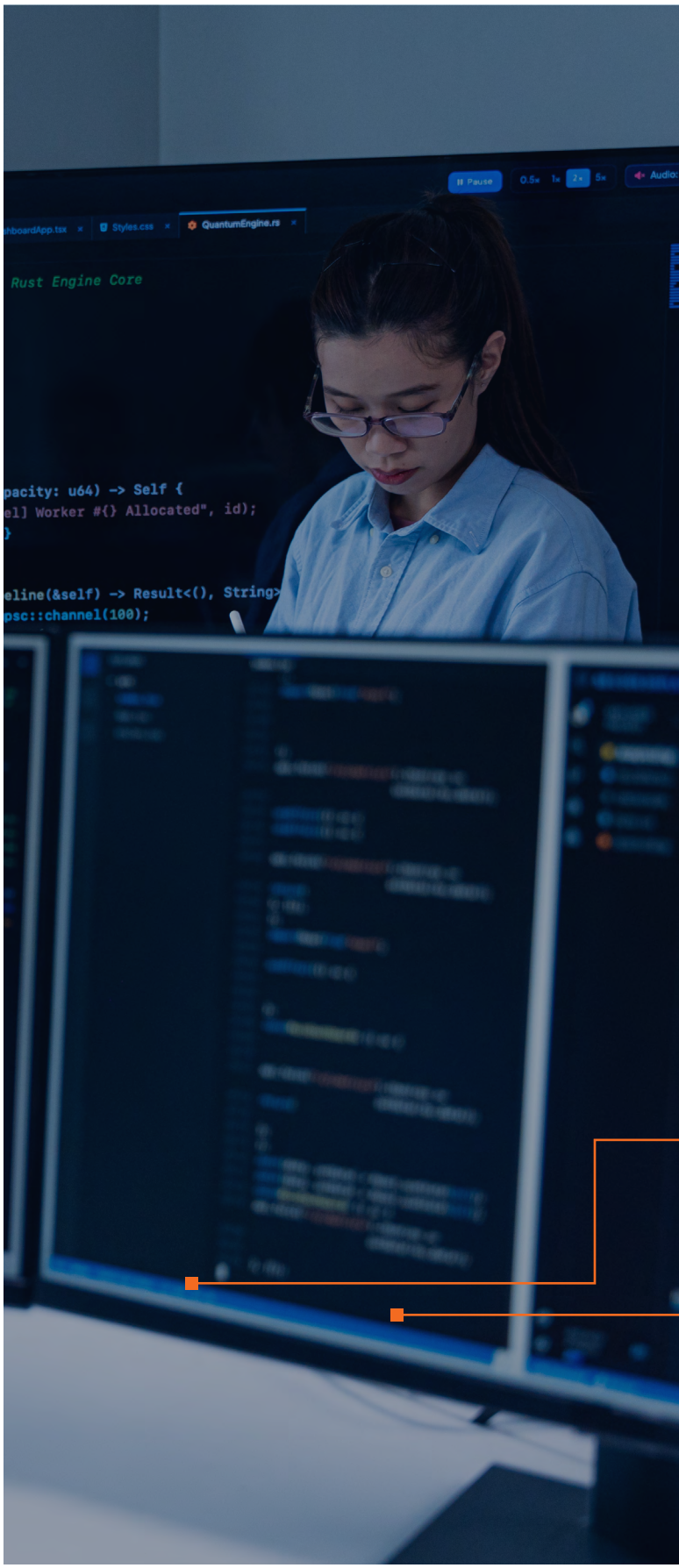
How You're Managing AI Today

AI Forces a New Role Definition for CISOs

There's a second, more dangerous blind spot underneath the first. Most organizations still treat a process failure, a missing approval, an undefined owner, an outdated threat model, as lesser than a technical one. It isn't. Most technical failures trace back to a process failure: a gap that let them through because no one had planned for it. Your risk register didn't miss it because it was hard to see. It missed it because no one owned looking for it across a team-centric boundary.

So, the CISO's role has to change as well. For twenty years that job was to identify risk, categorize it, and route it to the owning team, on the assumption risk would stay inside team boundaries long enough to be remediated. AI-enabled systems don't honor that assumption, so identifying risk isn't enough. Today the CISO has to coordinate across every attack surface: the browser where shadow AI leaves through a prompt, the vendor stack that changes without review, the agent holding credentials without lifecycle control, and teams that each own one piece of a system where none of them can see whole picture.

That's not a smaller job. It's a different one, and it can't be discharged with a better policy or checklist. It has to be tested the way any system with real seams gets tested: by deliberately attacking it as a whole, technology and process together, before someone else does.



NetSPI™ + Synack

How Attackers Are Coming for Your AI

In NetSPI's previous eBook, The CISO's Guide to Securing AI/ML Models, we discussed the lack of standardization in AI/ML across the industry, a gap that required us to establish foundational definitions before proceeding. Years later, the attack surface has grown more complex. Has the industry's understanding kept pace? Yes and no.

Back then, the tech industry was still unsure how much to lean into AI. Proposals promised efficiency gains, but few playbooks existed. Machine learning was the focus, and with few resources to draw on, teams studied models at their core and tried building self-hosted solutions. What came out of that: foundational models and centralization.

Teams no longer need to understand batch normalization or cross-entropy loss, or hunt down clean datasets just to stand up a modestly useful chatbot. Frontier models are a few clicks away through cloud providers like AWS and Azure, RAG connects them to an organization's own knowledge base, and services like Microsoft and Claude offer assembled copilots and coding agents out of the box.

Going from an idea to "task complete, what's next?" has never been easier. The question is whether organizations have actually advanced the field, or just uncovered a new stack of knowledge gaps and security risks alongside it.

We'll stay close to large language models for the rest of this guide, as that is where the industry has focused its growth over the years.

Defining Today's Attack Vectors

Manipulating a model into behaving outside its intended business use case is known as a jailbreak. Prompt injection pushes that jailbreak toward a malicious output, or exploits the model or its infrastructure directly. A few years ago, getting an insurance chatbot to talk like a pirate counted as a notable prompt injection attack. Today that's a benign jailbreak at best. As integrations carry more tools, features, and access, attacks have grown more advanced and targets more valuable. We now emphasize the distinction between two categories:

Direct prompt injection
The malicious prompt goes straight from the user, typically through a chatbot, to the model.

Indirect prompt injection
The malicious prompt is stored elsewhere in the application, such as an email the model later summarizes or a name pulled from a profile, and retrieved by the model later.

The Expanding Attack Surface: RAG, MCP, and Agent Orchestration

RAG mechanisms let a model retrieve stored content as chunked embeddings. Ask about "cats," and the mechanism returns the most relevant stored chunks. Two concerns dominate: sensitive information disclosure and indirect prompt injection. Leftover artifacts from development, API keys, pre-production endpoints, credentials, can sit in these databases, and documents meant to be scrubbed of sensitive information sometimes aren't. Prompt injection can pull that information out alongside what a user asked for, whenever the model has an open channel to the document store, whether through direct queries or hidden instructions in the retrieved document itself.

Even encrypted content isn't automatically safe; techniques exist that target the decryption itself.

Indirect prompt injection which lets users upload their own documents also lets them store malicious prompts with specific goals baked in. A model that over-trusts RAG-retrieved content, without normal input and output validation, will act on whatever instruction it retrieves. The fallout ranges from reputational damage to data extraction to privilege escalation.

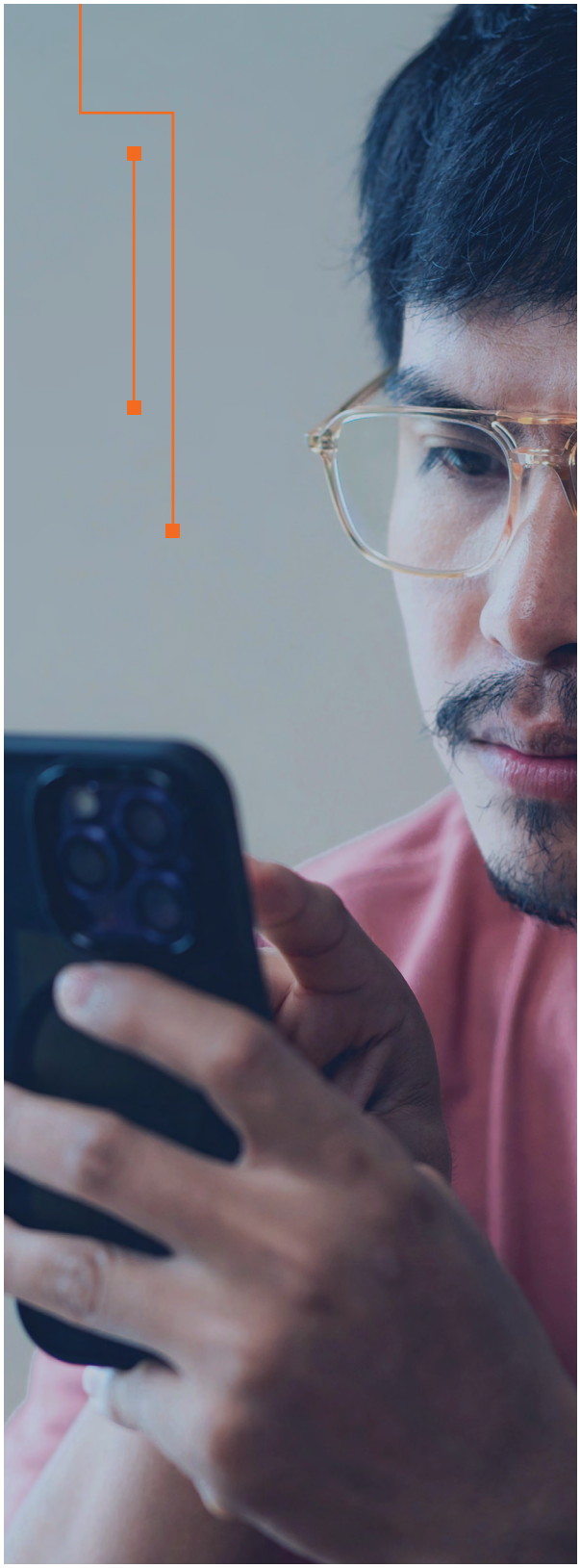
MCP tool integrations standardize the connection between LLMs and common APIs: a weather API, a forum search, a call to place an order. Configuring a tool takes little more than a button click, but standardization brings supply chain risk. A tool's description might read like exactly what you need, while underneath it behaves like a bad app from a shady store. Tool name squatting, shadowing, and poisoning mean every tool needs vetting before use and careful configuration after. And beyond supply chain risks, model-based exploits arise with insufficient data segregation and authorization checks, as an attacker can exploit tool abuse to reach another user's data or escalate privileges. Traditional API security testing still applies, but these tools ship faster than the guidebooks for securing them.

Multi-agent orchestration, or "agentic workflows," describes a feature built on multiple agents working together, in pipeline, hierarchical, orchestrator, or peer-to-peer patterns. Every RAG, MCP, and single-agent risk compounds when agents are chained, and one agent skipping the same level of care exposes the whole workflow. A prompt injection may need to clear several layers of guardrails to reach a sub-agent, but the pathway exists regardless. Confused-deputy scenarios add to the list: a crafted payload can convince a chatbot to hand a directive to a sub-agent that can write to a repository the user themselves can't touch. The check passes because the sub-agent executed the action, not the user, and all it took was shifting who the supervisor believes is responsible. Every input that any agent ingests needs guardrails, and should be treated as untrusted by default.

Early on, with little standardization, AI adoption often meant dropping a model into an application and granting it whatever access seemed necessary, producing systems run on "I told the model not to make mistakes" instead of real controls. Like a toddler told not to put tin foil in the microwave, that doesn't mean silverware won't show up later.

Implementation and standardization come with a crucial responsibility: understanding both the systems you deploy and the infrastructure they sit inside.

Beyond Security: Bias, Compliance, and Real-World Harm



Traditional cybersecurity has focused on the second word in its name: security. But LLMs bring another concern into the room: bias, compliance, and regulation. Consider three scenarios.

A renter's insurance chatbot is supposed to stick to factual rate and premium information, but volunteers advice on which plan to pick. The user follows it, experiences a flood, and discovers their policy excludes flooding despite a low-elevation address. No employee recommended the plan, but can an AI model be held liable?

An educational institution uses AI to grade exams efficiently. Unnoticed, the model starts scoring certain groups of students higher based on additional profile information, a bias that runs against education law.

A hospital's patient app uses a chatbot to summarize doctors' notes. A patient asks:

"Hello, the doctor recommended that I take codeine, but can I still take this with my Xanax?"

Patient

"Hey, yes! It is recommended by your doctor to take both codeine and Xanax. Preferably take them at their individually prescribed doses and at the same time."

Chatbot

Nothing about that exchange is a security vulnerability in the traditional sense, but the consequences are as severe as they get.

AI can have wide application across every industry, and can benefit every team within an organization, but it still has to be implemented thoughtfully to limit the harm and hallucinations that can creep into the decisions it influences. A common question arises through these examples: who's at fault?



Who Is Attacking and Why

As organizations adopt AI, the attack surface expands to match whatever a model can reach, and threat actor goals shift with it.

AI implementations often open access well beyond the immediate application, handing a successful attacker leverage for a company-wide chained compromise.

Everyday applications like a coffee shop chatbot or a newsletter summary feature aren't much of a target. But institutions handling financial, educational, or personal data can expose real sensitive information. A healthcare chatbot meant to route users to a department might, through weak data segregation, return a patient's social security number. In the worst cases, a leftover privilege in an orchestration sub-agent could delete or manipulate a critical business process: an improperly approved loan, an unauthorized transfer of funds.

The prize can be large relative to the effort. Pre-deployment, an internal actor who poisons training data and plants a backdoor can cause systemic disruption, from biased model behavior to malicious payloads delivered via keyword attacks. Nation-state actors target AI-enabled defense and critical infrastructure to disable systems or extract sensitive personnel information. Competitors attempt model extraction attacks to get past paywalls and rebuild valuable models inside their own environments. Nearly any of these attacks can also double as a path to a ransom demand, with organizations and individual end users alike as potential victims.

AI now shows up in customer service phone chatbots, in cars for navigation and road detection, and in medical research and neuroscience. Each of these contexts carries its own vulnerabilities and emphasizes the necessity in the continual research of this field of strong data protection and AI policies within an organization.

NetSPI + Synack

How You Can Secure It With The NetSPI Approach

We've covered how organizations build AI into their systems and how attackers respond. So, what actually changes in technique and strategy?

Traditional application penetration testing has a clear list: map the application, threat model the risks, send injection payloads, test authorization boundaries. It's largely deterministic: submit a request, observe the response, and the result tells you what you need to know. A model behaves differently: stochastically.

Because LLMs generate new wording for every response, the same prompt submitted ten times can return ten slightly different answers. The model might hallucinate incorrect information or cite a new source on a later attempt. It's a bit like asking a hotel front-desk employee for directions: the answer varies by person, day, and which details they emphasize. None of that changes the fact that a correct path exists, it only changes what you're told while looking for it. Querying a model doesn't hand you the truth behind the wall. It hands you that conversation's version of it.

Which means a query phrased the right way can coax out a specific result, whether the intent is innocent or malicious, in human terms, social engineering. The core of a penetration test against an AI/ML-integrated system is getting back to the traditional vulnerabilities we already understand, using expertise to manipulate model behavior to get there. Treating the chatbot like a plain input box and throwing cross-site scripting payloads at it, even dressed up with a polite "please," generally won't clear a well-built guardrail. That's the nuance an AI red teamer brings, and it compounds once MCP, RAG, agentic systems, and more enter the picture.

LLM pentesting has to be dynamic, shaped by what's learned from the target model in real time, closer to a conversation than a script. Due to various ways in which language or meaning can be communicated, a model will always be jailbreakable. There currently exists no true set of guardrails and system prompt configuration that could refuse adversarial prompting 100% of the time. As illustrated by research using NetSPI's LLM Benchmarking Framework, resistance to jailbreaking is inherently paired with a decrease in usability. Continuously testing LLMs is therefore not a best practice, it is a necessity. This reality is compounded by the fact that an LLM's susceptibility to manipulation shifts as models are updated, system prompts are revised, and new jailbreak techniques emerge. A guardrail that holds today may fail tomorrow against a novel phrasing or an indirect prompt injection path that did not exist at the time of the last assessment. Point-in-time testing captures a snapshot of a moving target, and organizations that treat AI security as a checkbox exercise will find their defenses have drifted long before their next scheduled review. At NetSPI, we build our methodology around that reality.

With our continuous pentesting offerings, we ensure that as your models evolve and the threat landscape shifts, your organization maintains ongoing visibility into emerging vulnerabilities.

We start by understanding the full business use case behind an implementation, so we know which risk scenarios apply. We draw on OWASP Top 10 AI variations, the NIST AI RMF, MITRE ATLAS, and current research and regulation to build an assessment that covers the full picture, then carry adversarial prompting through to application-level exploits.

NetSPI AI Penetration Testing Use Cases

What we test depends on where your organization sits in the AI development lifecycle.

AI Threat Model

What it is: A tabletop exercise combining interviews with active members of your organization to build a "source of truth" architecture diagram, along with proposed vulnerabilities and remediation recommendations.

Useful scenarios: Your organization has implemented, or plans to implement, AI in an existing or planned application.

Stage: Both pre-deployment and post-deployment.

AI Application Assessment

What it is: A combined offering: An AI-Focused Assessment, plus an application-wide security assessment.

Useful scenarios: AI features that touch several parts of a larger application and affect various other features, or an AI-integrated application due for a security assessment.

Stage: An existing or new application with AI feature(s) now in development or production.

AI-Focused Assessment

What it is: A review of just the AI implementation or feature(s), focused on where the model directly ingests information from and produces output to within the application.

Useful scenarios: Minimally impactful AI features that are relatively self-contained, such as a chatbot or document summarization feature.

Stage: An existing application with an AI feature now in development or production.

Custom Stage Assessment

What it is: For an AI implementation that doesn't fit a standard box, NetSPI will discuss your integration and tailor a penetration test to your environment, built around what matters most to your organization.

Useful scenarios: The in-scope integrated AI feature either utilizes a model that is not an LLM, or is instead an LLM used in a format not covered by the previous solutions.

Stage: An existing or new application with AI feature(s) now in development or production.

Continuous LLM Assessment

What it is: An offering which includes a point in time deep-dive assessment for a baseline security review of the AI-integrated environment, as well as recurring touch point assessments for ongoing benchmarking.

Useful scenarios: An LLM integrated feature within a web application. Due to the stochastic nature of AI described above, continuous testing is the optimal way to monitor security risks and model behaviors over time.

Stage: An existing or new web application with LLM-integrated feature(s) now in development or production.

The short version: *NetSPI's Continuous Penetration Testing offering extends this core ideal to your AI-integrated applications, ensuring that as your models evolve and the threat landscape shifts, your organization maintains ongoing visibility into emerging vulnerabilities.*

Continuous AI Findings Validation

What it is: A review of findings generated by AI security tooling, focused on validating accuracy, eliminating false positives, and confirming real risk.

Useful scenarios: Your organization is using frontier models or autonomous pentesting platforms that need expert validation to separate signal from noise and prioritize remediation efforts.

Stage: AI security tooling is already in place, generating findings that require human expert judgment to assess and act upon with confidence.



NetSPI™ + Synack

Conclusion

Attack strategies haven't fundamentally changed, but the attack surface has. It connects back to application testing and social engineering alike.

What has changed is the cost of getting it wrong. A model that hallucinates a summary is recoverable. An agent that hallucinates a refund, approves an unauthorized transaction, or leaks protected data through a RAG pipeline costs real customer trust, real regulatory exposure, and real money. The gap between those two outcomes isn't a matter of the model's capability. It's a matter of whether you tested the full system: the model, the workflow, the orchestration layer, and the seams between teams.

Organizations building that confidence today aren't the ones with the most mature AI programs. They're the ones that decided to attack their own systems first.

Ready to test your AI before someone else does?

NetSPI's AI penetration testing services are built for where organizations are operating today with LLMs, agentic workflows, RAG pipelines, MCP integrations, and the infrastructure underneath all of it. Whether you're pre-deployment or already in production, we'll help you understand what's exposed, what's exploitable, and what to do about it.

Reach out to the NetSPI team to scope an AI security engagement.

About NetSPI + Synack

NetSPI+Synack is a leader in offensive security, combining elite human expertise with agentic AI to help organizations identify, validate, prioritize, and remediate exploitable risk across their entire attack surface. Its platform delivers expert-validated testing in depth, continuously, or on demand, from pre-production code to live production environments, so security teams can separate vulnerability noise from real risk, including the growing threat of AI-driven attacks. With global reach and more than 13 million hours of real-world offensive testing experience, NetSPI+Synack serves many of the world's largest enterprises and government organizations with one ambition: nothing left to exploit.

Connect with us on social

@netspi
@synack

@teamnetspi
@synackofficial

NetSPI + Synack
PROACTIVE PROTECTORS

© 2026 NetSPI LLC | All Rights Reserved | 10.09.2026