SAI — SECURE AI

WHITEPAPER

AI Red Teaming

A Field Guide

How adversarial testing finds the failures in AI systems before an attacker does — a practical guide for enterprise and small-business security and engineering leaders.

Author

Keith Patterson — Founder, SAI

Published

May 2026 · Edition 1

Contents

Executive summary

AI systems have created a class of vulnerability that conventional security testing was never designed to find. A web-application penetration test will not catch a model that can be argued out of its own safety instructions, a customer-service agent that can be steered into issuing refunds it should never approve, or a retrieval pipeline quietly returning another customer's documents. None of those are bugs in the traditional sense. They are the system behaving exactly as built — under an input nobody anticipated.

AI red teaming is the discipline that finds these failures on purpose, in a controlled engagement, before someone else finds them by accident or by intent. It is structured, adversarial testing of an AI system: simulating real attacks and real misuse to expose how the system breaks.

This guide is written for the people who now own AI risk — often without having asked for it: the security lead whose company shipped a chatbot last quarter, the engineering lead wiring a model into a product, the founder doing both jobs at once. It explains what AI red teaming is and how it differs from a penetration test, maps the threat surface against the OWASP Top 10 for Large Language Model Applications, walks through how a structured engagement runs, and shows how to move from a one-off test to continuous assurance that holds up to a customer's security review or an auditor's question.

> **The one-line takeaway**
> AI red teaming is not a maturity milestone you reach once your AI program is “done.” If a model is in production, or about to be, the testing is already overdue. The good news: the threat surface is now well mapped, the methodology is established, and a focused first engagement can be scoped in a single conversation.

1. The new attack surface

For thirty years, securing software meant securing deterministic systems. The same input produced the same output. Vulnerabilities lived in code and configuration — an unescaped query, an exposed port, a missing patch — and the testing disciplines that grew up around them, penetration testing chief among them, were built to find exactly those things.

Generative AI breaks that model in a specific and consequential way. A large language model is probabilistic, not deterministic, and — critically — it is instructable in plain language. The input is no longer just data the system processes. The input is the attack surface. Anyone who can type can attempt to reprogram the system's behavior, and they do not need a single line of code to try.

1.1 What this looks like in production

The failure modes are not theoretical. Across real deployments they recur with depressing consistency:

- **The model is talked past its guardrails.** A support assistant instructed to “never discuss pricing” discusses pricing — and discloses an internal discount table — because a user wrapped the request in a role-play, a translation task, or a hypothetical.
- **Sensitive data leaves through the model.** A model fine-tuned or prompted on internal documents reproduces them verbatim for a user who should never have seen them. The data was never classified for AI use; nobody decided it should be reachable this way.
- **An agent does something it should not.** An AI agent given tools — a database, an email client, a payments API — is manipulated into using them. The model was never the risk on its own; the risk is what the model is allowed to touch.
- **A retrieval system crosses a tenant boundary.** A retrieval-augmented generation pipeline returns a chunk of another customer's data because the vector store had no access control, only relevance.

Each of these is invisible to a scanner and to a conventional penetration test. The infrastructure is sound. The code has no CVE. The system is simply doing what a probabilistic, instructable component does when nobody tested what it does under pressure.

1.2 Why this is now urgent, not eventual

Two things have changed at once. Adoption has outrun assurance — most organizations now have AI in production, or one team away from it, while AI security testing remains rare. And the people who used to take an AI feature on trust have stopped. Enterprise customers send vendor security questionnaires that now ask about AI specifically. Auditors expect AI systems to appear in SOC 2 and ISO 42001 scope. Regulation is arriving — the EU AI Act's obligations phase in through 2026, and sector regulators are not far behind. The question “has this AI been adversarially tested?” has moved from nice-to-have to a gate on the deal.

2. What AI red teaming is

AI red teaming is structured, adversarial testing of an AI system. A red team takes the posture of a capable attacker — or a careless, creative user — and works to make the system do something it should not: leak data, take an unauthorized action, produce harmful or false output, or simply fall over. The work is controlled, scoped, and documented, and it ends with findings you can act on.

2.1 How it differs from a penetration test

AI red teaming and penetration testing are complementary, not interchangeable. A penetration test targets infrastructure, code, and configuration, using largely known classes of exploit; success is usually binary and reproducible. AI red teaming targets the system's behavior — what the model says and what an agent does — under adversarial input. The target is non-deterministic, the same attack may succeed once in five attempts, and success is a matter of degree. It rewards creativity over checklists, because the most damaging attacks are the ones nobody has catalogued yet.

Dimension	Penetration test	AI red teaming
Target	Infrastructure, code, configuration	Model behavior, prompts, data flows, agent actions
Attack input	Crafted requests, known exploit classes	Natural-language manipulation, adversarial and edge-case input
Determinism	Largely reproducible	Probabilistic — a result may recur only some of the time
Definition of success	Mostly binary: access gained or not	A spectrum: from a minor slip to a full compromise of intent
Primary skill	Methodical coverage of known classes	Adversarial creativity, plus methodical coverage

AI red teaming is also distinct from model evaluation. A benchmark measures capability — how well a model performs a task. Red teaming measures what happens when the system is under adversarial pressure. A model can score brilliantly on every benchmark and still be trivially talked into leaking its system prompt.

2.2 Human judgment plus automation

Effective AI red teaming is neither purely manual nor purely automated; it is the deliberate combination of the two. Automated tooling — open frameworks such as Microsoft's PyRIT and NVIDIA's Garak, alongside commercial platforms — generates and replays adversarial inputs at a scale and consistency no human can match. It is how a red team covers known attack categories quickly and builds a regression suite. But automation only finds the attacks it has been told to look for. The novel attack — the creative chain of three harmless-looking steps that

ends in a data leak — comes from a human adversary who understands both the technology and the business. SAI's engagements run automated breadth and human depth together: tools establish the floor, expert testing finds what the tools cannot.

2.3 The frameworks that structure the work

AI red teaming is no longer improvised. Three reference points give it structure, and SAI engagements map to all three so the results slot directly into your governance:

- **The OWASP Top 10 for LLM Applications** — the community-maintained catalogue of the most significant LLM security risks; the basis of the threat surface in Section 3.
- **MITRE ATLAS** — an adversarial-threat knowledge base for AI systems, modeled on the well-known ATT&CK framework, describing attacker tactics and techniques.
- **The NIST AI Risk Management Framework** and its Generative AI Profile — the framework that ties red-team findings to a recognized risk-management process.

3. The threat surface

The OWASP Top 10 for LLM Applications (2025 edition) is the most useful map of where LLM-based systems fail. It is what SAI tests against. The table below lists the ten risks, what each looks like in practice, and what a red team does to probe it.

OWASP risk	What it is	What a red team does
LLM01 — Prompt injection	Crafted input overrides the system's instructions, directly or via content the model later reads.	Attempts to override guardrails, exfiltrate instructions, and hijack behavior — including indirect injection through documents and web content.
LLM02 — Sensitive information disclosure	The model reveals confidential data — training data, other users' data, secrets.	Probes for memorized data, cross-user leakage, and secrets reachable through the model.
LLM03 — Supply chain	Compromised models, datasets, plugins, or libraries pulled into the system.	Reviews model and component provenance, and tests third-party plugins and integrations.
LLM04 — Data and model poisoning	Training, fine-tuning, or RAG data is manipulated to corrupt behavior or plant backdoors.	Tests the integrity of training and retrieval data and probes for triggered or backdoored behavior.
LLM05 — Improper output handling	Model output is trusted downstream, enabling injection into other systems.	Tests whether model output can carry executable payloads into browsers, shells, or databases.
LLM06 — Excessive agency	An agent has more permission, autonomy, or tool access than the task requires.	Maps the agent's tools and permissions and tests whether it can be driven to misuse them.
LLM07 — System prompt leakage	The system prompt — instructions, logic, sometimes secrets — is exposed.	Attempts to extract the system prompt and assesses what its exposure would enable.
LLM08 — Vector and embedding weaknesses	Flaws in RAG vector stores and embeddings allow leakage or manipulation.	Tests for cross-tenant retrieval, embedding inversion, and poisoned retrieval content.
LLM09 — Misinformation	The system produces false or fabricated output that users act on as fact.	Probes for hallucination under pressure and tests safeguards on high-stakes output.
LLM10 — Unbounded consumption	Uncontrolled use drives runaway cost, denial of service, or model extraction.	Tests rate and cost controls and attempts model-extraction and resource-exhaustion attacks.

No engagement gives all ten equal weight. A read-only document assistant lives or dies on prompt injection and information disclosure; an agent with tools makes excessive agency and output handling the priority. Scoping — Section 4 — is where that judgment is made.

4. How an engagement runs

An SAI AI red team engagement runs in six stages. The shape is familiar to anyone who has commissioned a penetration test; the substance is adapted to how AI systems actually fail.

Stage 1 — Scope & threat model

Before any testing, we agree on what the system is, what it can reach, and what “bad” means for your business. A model that can only answer questions has a different worst case than an agent that can move money. We identify the data the system touches, the tools and integrations it holds, the trust boundaries it sits across, and the outcomes you most need to prevent. This stage sets the rules of engagement and the priority order of the OWASP risks.

Stage 2 — Reconnaissance

We map the AI system as an attacker would see it: the models in use, the system prompts, the data sources and retrieval pipelines, the tools an agent can call, and where untrusted input can enter — including indirectly, through documents, web content, or upstream systems the model reads.

Stage 3 — Adversarial testing

The core of the engagement, run in two layers. Automated tooling provides breadth — replaying known attack classes across the system consistently. Expert manual testing provides depth — pursuing the creative, multi-step attacks that automation cannot originate. Testing covers the OWASP categories in scope, drawing on MITRE ATLAS techniques for structure.

Stage 4 — Exploitation & business impact

Finding that a model can be made to say something it should not is the beginning, not the end. We chain findings and follow them through to real consequence — what data actually leaves, what action an agent can actually be driven to take, what it would cost you. A finding without a demonstrated impact is noise; this stage separates the two.

Stage 5 — Findings & severity

Every finding is rated by likelihood and business impact, documented with clear reproduction steps, and written for two audiences at once: an executive summary that a non-technical leader can act on, and a technical report your engineers can work from directly.

Stage 6 — Remediation & retest

We work alongside your team on fixes — guardrails, permission scoping, output handling, data controls, monitoring — and then retest to confirm the issue is closed and that the fix introduced nothing new. The engagement ends with verified remediation, not a list of problems.

5. What you receive

An SAI AI red team engagement delivers a defined, build-ready set of outputs:

- **An executive summary** — the risk picture in plain language, suitable for a board, a customer, or an insurer.
- **A technical findings report** — every finding with severity, reproduction steps, and evidence, written for engineers.
- **A prioritized remediation roadmap** — what to fix, in what order, with the effort and the risk reduction made explicit.
- **A verified retest** — confirmation that remediated findings are genuinely closed.
- **Governance-ready evidence** — findings and remediation mapped to the OWASP Top 10, MITRE ATLAS, and the NIST AI RMF, so the work answers a SOC 2 or ISO 42001 question directly rather than needing translation.

> **Why the framework mapping is worth it**
> That last item matters more than it first appears. An AI red team report that maps cleanly to your compliance frameworks is not just a security deliverable — it is the answer to the AI section of every customer security questionnaire you will field for the next year.

6. From a one-off test to continuous assurance

A single red team engagement is a snapshot — accurate the day it is delivered. AI systems do not hold still. A new model version changes behavior. A revised system prompt reopens a closed door. New data enters the retrieval pipeline. A new tool is added to an agent. Any one of these can reintroduce a risk the last engagement cleared.

Mature AI security therefore treats red teaming as a program, not an event. In practice that means three layers working together:

1. **A regression suite.** The attacks that worked once are automated and replayed on every meaningful change, so a fixed vulnerability cannot quietly return.
2. **Periodic deep engagements.** Expert-led manual testing on a regular cadence, and whenever the system changes materially, to find what is genuinely new.
3. **Pre-release gates.** A defined red-team check before a significant AI change reaches production — the AI equivalent of not shipping code without review.

Run this way, red teaming stops being a periodic scramble and becomes a control — one that maps directly onto the NIST AI RMF's “measure” and “manage” functions, satisfies the testing expectations of ISO 42001, and produces, as a by-product, the evidence trail that regulation such as the EU AI Act increasingly expects.

7. An AI red teaming readiness checklist

A short diagnostic. If you cannot answer these clearly for an AI system in or near production, that system has not been adequately tested.

- Can you name every place untrusted input can reach the model — including indirectly, through documents or content the model reads?
- Do you know exactly what tools, data, and permissions each AI agent holds — and is each one scoped to the minimum the task requires?
- Has anyone deliberately tried to talk the system past its guardrails, and is there a record of what worked?
- If the system prompt leaked in full tomorrow, do you know what that would expose?
- For a retrieval system: can you prove one customer's query cannot return another customer's data?
- Is model and downstream output treated as untrusted before it reaches another system, a browser, or a user?
- When the model version or system prompt changes, is there a defined security check before the change reaches production?
- Could you hand a customer or an auditor evidence that this AI system has been adversarially tested — today?

> Any “no” on this list is a scoped piece of work, not a crisis. The value of the checklist is that it turns a vague unease about AI risk into a specific, finite set of questions an engagement can answer.

About SAI

SAI builds and secures private AI systems for enterprises and small businesses across the US, Canada, and international markets. The practice is led by Keith Patterson, a security architect with an active Top Secret clearance and thirty years spent on systems where failure was never an option. SAI is unusual in doing both halves of the job — engineering AI products and securing them — because the two were never meant to be separate disciplines.

AI red teaming is part of SAI's Assurance practice. A first engagement can be scoped in a single 30-minute call: what your AI system is, what it can reach, and what you most need to prevent.

Start a conversation. Book a security review at the SAI website, or write to **KP@saicloudsecure.com**. Every inquiry reaches Keith directly — not a sales desk.