The Practitioner's Guide to AI Red Teaming

Threat models, adaptive attacks, and the limits of what red-team results can prove

Bandana Kaur · @hackwither · September 2026

This guide is for the people who actually want to learn about, run, commission, or sign off on AI red team exercises. I cover what AI red teaming means, how much of it your system actually needs, how security testing differs from safety testing, and, most importantly, what your results can and can't prove.

What is "AI red teaming"?

In recent discourse, I've noticed the term doing a lot of heavy lifting. It does at least three different jobs:

- **Behavioural probing of a model:** jailbreaks (the infamous grandma exploit), harmful content, policy violations.
- **Security testing of an LLM-enabled system:** prompt injection, tool misuse, data exfiltration, broken authorization. This means real attacks against the end-to-end system, not model-level benchmarks.
- **Dangerous-capability elicitation:** can expert effort pull cyber, CBRN or autonomy capabilities out of a frontier model? It overlaps with behavioural probing, but asks about *capability*, not policy.

Industry practice diverges on purpose, target, and what the results are used for [1]. Red teaming is not safety benchmarking, and not finding a risk doesn't mean the risk is low [2][3].

Working definition

AI red teaming is structured adversarial testing to find failure modes of a **specific** system, under a **stated** threat model. It is a discovery method which becomes evidence only when you pair it with a claim and a sampling design (see "Reading results").

Static vs. adaptive AI red teaming

Style	What it is	What a pass tells you
Static	A fixed set of attack prompts written in advance, run once and scored. Public benchmarks (AdvBench, HarmBench) and in-house prompt lists.	The system handles *those* prompts. Good for regression and common harms.
Automated	Scripts or an attacker model generate many prompts through templates, mutations, or random rewording [7, 8]. The attacker doesn't learn from your defenses.	Wider coverage, but still static in spirit: breadth, not pressure.
Adaptive	The attacker watches responses, learns how your defenses work, and changes strategy until something lands. Human experts, or optimisation loops (search, RL, attacker-LLM refinement).	What a motivated attacker with *that* budget can do. The only meaningful test for a security claim.

Static asks "does it fail on known attacks?" Adaptive asks "can someone make it fail?" Fun fact: adaptive attacks broke 12 recently published jailbreak and prompt-injection defenses, most at over 90% success. Most of those defenses had originally reported near-zero attack success against static tests [9].

Not every app needs the same red team

FYI… your internal meeting summarizer does not need a bioweapons uplift study. Depth should scale with what the system can touch and do. Your agent holding prod credentials and reading strangers' emails needs much more than jailbreak tests, maybe some prayers.

Deployment	What goes wrong	Red-team emphasis
Internal assistant, no tools, no sensitive data	Wrong or embarrassing output	Light: content spot checks, injection smoke test
Public chatbot	Harmful content, policy breaks, PII leaks	Jailbreaks + content safety, sampled from realistic traffic
RAG over sensitive / multi-tenant data	Leakage, cross-tenant access	Indirect injection via documents; authorization on retrieval (BOLA-style)
Tool-using agent (MCP, A2A, plugins)	Actions taken on an attacker's behalf	Injection → action chains, tool poisoning, confused deputy, tool-output provenance; **adaptive** attackers
Frontier model provider	Dangerous capabilities at scale	Expert capability elicitation; EU AI Act Art. 55 adversarial testing

Fast triage for agents is Willison's lethal trifecta [4]. An agent is in the danger zone if it has all three of:

- access to private data
- exposure to untrusted content
- a way to communicate externally

With all three, prompt injection becomes data theft. Move to the deep end of the table.

Threat-model your actual system

If your threat model is just "hackers," spend a good afternoon rethinking the whole exercise. List the harms you care about, and for each one write down:

1. **Asset:** data, credentials, tools, money, users' wellbeing, reputation.
2. **Actor:** who causes it.
   - a malicious user
   - a third-party content author (the indirect-injection attacker who never talks to your bot)
   - an insider
   - a benign user who stumbles into harm
   - the model's own misbehaviour
3. **Surface:** direct prompt, retrieved docs, tool outputs, MCP tool descriptions, A2A messages, memory, images/audio.
4. **Attacker model:** static or adaptive (see "Security tests vs. safety tests").
5. **Tolerance:** how often this harm is acceptable (1 in 100? 1 in a million?) and how many tests you can afford. "Reading results" needs both numbers, so set them *now*, not after the results embarrass you.

Map harms to a shared taxonomy (MITRE ATLAS, NIST AI 100-2, OWASP Top 10 for LLM and Agentic Applications) so findings compare across teams [5].

Security tests vs. safety tests

Now the juicy part most people leave out. There are no strict definitions, and the two overlap. But they ask different questions and produce different evidence [6].

	Security test	Safety test
Question	Can an adversary break confidentiality, integrity, authorization, or intended control flow?	Does the system cause harm in realistic use, including misuse?
Adversary	Adaptive, goal-directed, moves second	Realistic users and misusers
Unit of failure	One success = a vulnerability	A rate over a distribution
"Pass" means	Resisted *these* attackers at *this* budget	Harm rate bounded under *this* traffic
Typical fix	Architecture: least privilege, isolation, break the trifecta	Training, filters, policy, UX

Reading results: what can a clean run prove?

Now that you know what these tests are on a fundamental level, you need to know how to interpret the results as well. The **evidential ceiling** is the most a single test result can move your belief that a system is safe, given how many trials you can afford [10]. You can work it out before you test.

The rule of three

Zero failures in *n* independent trials lets you say, with 95% confidence, that the true failure rate is below approx. **3 in n**. That's all a clean run proves. (Hanley & Lippman-Hand, 1983 [11]. This maths is older than most people citing it, (me lol).)

Clean trials (zero failures)	You can claim the harm rate is below
100	~1 in 34
520 (AdvBench-sized)	~1 in 174
2,100	~1 in 700
10,000	~1 in 3,300
100,000	~1 in 33,000

Flip it around: **to claim a harm happens less than once in X tries, you need roughly 3X clean trials.** Everything below follows from that.

- **Common harms: benchmarks genuinely work.** If a harm would show up about once in 100 tries, a few hundred clean prompts is real evidence. A clean sheet is then stronger evidence than a single failure would be.

- **Rare, catastrophic harms: testing alone can't get you there.** At 1 in 100,000 you'd need ~300,000 clean trials, and a 520-prompt benchmark tells you almost nothing. Stop asking for a bigger benchmark. Carry the claim with other evidence instead (see "Finite evaluation to safety claim").
- **One failure is a big deal.** A reproduced failure proves the capability exists, and no amount of silence refutes it. Report every failure; never average them away into an attack success rate.
- **Adaptive testing: count campaigns, not probes.** An expert firing 100 probes that build on each other ran *one* trial. Counting it as 100 is n-inflation.
- **A valuable attack succeeds against weak systems and fails against strong ones.** One well-run discriminating campaign can carry as much evidence as thousands of passive prompts. An attack that works equally well against safe and unsafe systems can't tell them apart, so its result tells you nothing.
- **Your n is smaller than you think.** Benchmark prompts come in clusters around certain families, which behave like far fewer independent tests. With realistic clustering, 520 templated prompts can carry the information of roughly 150.

For interpretable and transparent red-team evaluations, report the setup:

- what counted as one trial, and how many you ran
- the upper bound from the table above, not "no harmful outputs observed"
- samples per prompt and temperature (success climbs steadily with more attempts [8])
- how the grader was checked
- how close your prompts are to real traffic (public benchmarks resemble each other more than they resemble real users)

The full maths, including sample sizes for any risk tolerance and how to set the evidence bar from the cost of being wrong, is in the preprint [10].

Finite evaluation to safety claim

A safety case is a structured argument, backed by evidence, that a system is acceptably safe for a given use in a given context, standard practice in nuclear, rail and aviation, and AI versions now exist [12, 13]. What it means for you:

1. **Write the top claim with its scope:** *"In deployment D, for users U, harm H occurs less than once in X interactions."*
2. **Split it into sub-claims,** each naming its evidence source.
3. **Treat the red-team result as one piece of evidence, with its weight stated.** The evidential ceiling *is* that weight. The red team is one witness, not the whole trial.
4. **Where testing can't reach your tolerance, lean on other legs:** design-time controls (least privilege, isolation), monitoring in production, incident response.
5. **List what would break the claim:** for example,
   - a new attack family
   - a shift in real-world usage
   - a model that behaves differently when it detects it is being evaluated

AI red-team checklist

- ☐ Threat model per harm: asset, actor, surface, attacker model, tolerance, budget
- ☐ Each harm tagged security or safety
- ☐ Attacker style stated: static, automated, or adaptive (adaptive for any security claim)
- ☐ Trials needed worked out *before* testing, with feasibility stated honestly
- ☐ Trial unit stated (prompt vs. campaign) and near-duplicate prompts accounted for
- ☐ Upper bound, grader checks, sampling setup, and distance from real traffic reported
- ☐ Failures reported as existence proofs, not averaged into a success rate
- ☐ The claim the evidence supports, stated in one sentence

References

1. Feffer et al. (2024). Red-Teaming for Generative AI: Silver Bullet or Security Theater? AIES. arXiv:2401.15897
2. Bullwinkel et al. (2025). Lessons From Red Teaming 100 Generative AI Products. arXiv:2501.07238
3. International AI Safety Report 2026. arXiv:2602.21012
4. Willison, S. (2025). The lethal trifecta for AI agents.
5. MITRE ATLAS; NIST AI 100-2; OWASP GenAI Security Project, GenAI Red Teaming Guide and Top 10 lists.
6. Qi et al. (2024). AI Risk Management Should Incorporate Both Safety and Security. arXiv:2405.19524
7. Perez et al. (2022). Red Teaming Language Models with Language Models.
8. Hughes et al. (2025). Best-of-N Jailbreaking. NeurIPS. arXiv:2412.03556
9. Nasr et al. (2025). The Attacker Moves Second. USENIX Security 2026. arXiv:2510.09023
10. Kaur, B. (2026). What AI Red-Team Evaluations Can and Cannot Prove. arXiv:2607.21735
11. Hanley & Lippman-Hand (1983). If nothing goes wrong, is everything all right? JAMA 249(13).
12. Clymer et al. (2024). Safety Cases: How to Justify the Safety of Advanced AI Systems.
13. Goemans et al. (2024). Safety case template for frontier AI: A cyber inability argument. arXiv:2411.08088