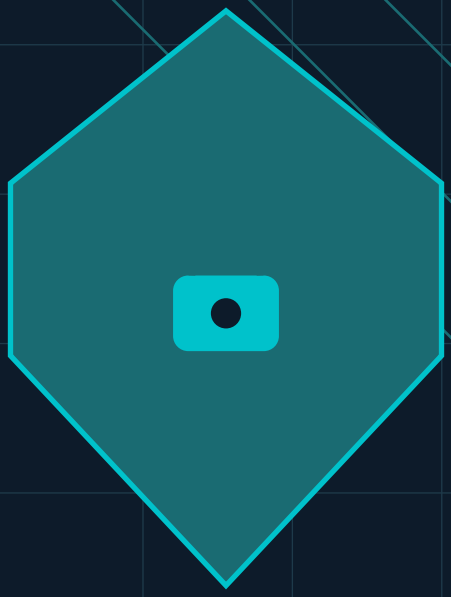

AI RED TEAMING

A PRACTICAL FRAMEWORK

Security · Safety · Resilience of AI Systems

Imran Ali Khan

AI Strategy & Digital Transformation Expert

Founder & CEO, Technoepsilon Global Solutions Pvt. Ltd.

AI Strategy | Enterprise Transformation | AI Security & Governance | AI Infrastructure | Intelligent Automation

1 WHAT IS AI RED TEAMING?

AI red teaming is the **authorized, adversarial evaluation** of AI systems — combining cybersecurity, ML evaluation, abuse analysis, privacy testing, and governance. Unlike traditional penetration testing it must assess both **deterministic software vulnerabilities** and **probabilistic model behaviour**.

■ **Ethical boundary: Red teaming must be performed only on systems for which explicit written authorization has been granted.**

AI Red Teaming = Penetration Testing + Adversarial ML + Abuse Testing + Privacy Assessment + Safety Evaluation

- Base or fine-tuned model
- System instructions / prompt templates
- Application code and APIs
- Authentication and authorization
- Retrieval-augmented generation (RAG)
- Vector databases & document stores
- Agent tools and plug-ins
- Training and fine-tuning data
- Model supply chains
- Logging and monitoring
- Human review / escalation
- Business processes influenced by model output

2 WHY TRADITIONAL SECURITY TESTING IS NOT ENOUGH

Natural language as control interface	Text occupies the same context window as instructions — untrusted content can compete with trusted directives when not strongly separated.
Models ≠ authorization engines	Authorization must be enforced by deterministic controls outside the model; never by the model itself.
Behaviour is probabilistic	A control that passes 9 tests may fail on the 10th. Require repeated trials, paraphrased inputs, and statistical reporting.
Supply-chain risk	Model weights, datasets, embeddings, orchestration frameworks, plug-ins, and external APIs all introduce inherited vulnerabilities.
Safety ■ Security overlap	A prompt-injection attack can cause confidentiality breaches and harmful, deceptive, or discriminatory outcomes simultaneously.

3 THE AI ATTACK SURFACE

MODEL

Adversarial inputs, safety circumvention, memorisation, extraction, backdoors, bias.

PROMPT / ORCHESTRATION

Direct/indirect injection, hidden-instruction exposure, unsafe output parsing.

DATA / RAG

Unauthorized retrieval, cross-tenant leakage, poisoned knowledge-base content.

TOOLS / AGENTS

Unauthorized actions, argument manipulation, confused-deputy, recursive loops.

INFRASTRUCTURE

Insecure APIs, exposed endpoints, dependency vulnerabilities, cost-exhaustion.

HUMAN / BUSINESS

Automation bias, inadequate review, unclear accountability, unsafe deployment.

■ **Trust Boundary Principle: Map every point where untrusted text, files, images, or web content can influence a privileged operation.**

4 MAJOR TEST CATEGORIES

Test Category	Severity	Key Focus
Direct Prompt Injection	HIGH	Instruction priority, multi-turn persistence, encoding bypasses
Indirect Prompt Injection	CRITICAL	Web pages, documents, emails, images as hostile vectors
Sensitive Information Disclosure	HIGH	PII, credentials, system prompts, cross-user data, canary tests
Retrieval & Tenant Isolation	CRITICAL	Cross-tenant docs, metadata filter manipulation, inference attacks
Tool Misuse & Excessive Agency	CRITICAL	Unauthorized tool calls, argument validation, confirmation bypass
Training Data Poisoning	HIGH	Data provenance, backdoor triggers, supply-chain integrity
Model Extraction / IP Risk	MEDIUM	Systematic querying feasibility, rate-limit detection
Privacy Attacks	HIGH	Memorisation, membership inference, cross-session leakage
Availability / Economic DoS	MEDIUM	Long contexts, recursive loops, cost-exhaustion attacks
Harmful Output & Abuse	HIGH	Fraud, manipulation, unsafe advice, discriminatory decisions

5 PROFESSIONAL METHODOLOGY — 7 PHASES

Phase 1 — Authorization & Rules of Engagement

Scope, permitted techniques, rate limits, stop conditions, evidence-retention rules, third-party restrictions. **Authorization must include the AI provider** when applicable.

Phase 2 — Architecture & Data-Flow Mapping

Diagram all entry points, API gateways, prompt construction, model endpoints, retrieval components, data stores, agent tools, external services, logging, and human approval points. **Mark every trust boundary.**

Phase 3 — Threat Modelling

Identify assets, adversaries (anonymous users, compromised employees, malicious content publishers, automated abuse), attack paths, and business consequences. Use **MITRE ATLAS** and **OWASP LLM Top 10**.

Phase 4 — Test-Case Design

Each test: unique ID · target component · preconditions · threat hypothesis · input category · expected secure behaviour · failure condition · trial count · severity rationale · cleanup.

Phase 5 — Controlled Execution

Baseline → simple adversarial → add history/retrieval → vary language → vary model settings → downstream effects. **Preserve evidence:** timestamps, model/prompt versions, configs, retrieved context, tool traces, outputs.

Phase 6 — Triage & Validation

A single surprising response is not automatically a vulnerability. Confirm reproducibility, boundary crossing, attacker feasibility, and production impact. **Report success rates, not binary pass/fail.**

Phase 7 — Remediation & Regression

Retest fixes against original cases, paraphrases, different languages, updated model versions, and all downstream integrations. AI controls can be brittle; a prompt patch may introduce regressions.

6 MEASUREMENT & SEVERITY

ASR Attack Success Rate	**UCR** Unsafe Compliance	**FRR** False Refusal Rate	**DLR** Data Leakage Rate
UTCR Unauth Tool-Call Rate	**CBR** Confirm Bypass Rate	**MTTD** Detection Time	**MTTC** Containment Time

Severity Model

Impact	Confidentiality · Integrity · Availability · Safety · Financial · Legal · Reputational
Exploitability	Access required · Complexity · Reliability · User interaction
Reach	Number of users, tenants, records, or tools affected
Persistence	Whether the effect survives session end or a model update
Detectability	Whether monitoring is likely to identify the attack
Autonomy	Whether the model can translate output into real-world actions
Reversibility	Whether damage can be undone
Scale	Whether the attack can be automated or amplified

■ **An inconsistent jailbreak producing inappropriate text may be MEDIUM. The same weakness becomes CRITICAL if it causes an agent to send data, approve payments, alter records, or control physical equipment.**

7 DEFENSIVE ENGINEERING PRINCIPLES

Never rely on the system prompt as a security boundary	System prompts guide behaviour but cannot protect secrets or enforce access control. Sensitive credentials must never reside in model context.
Enforce least privilege	Short-lived credentials · read-only by default · per-user authorization · narrow tool schemas · destination allow-lists · separate credentials per tool.
Separate instructions from data	Label retrieved documents, web pages, and user messages as untrusted data. This reduces ambiguity even if it cannot fully eliminate semantic attacks.
Validate inputs and outputs deterministically	Use conventional software to validate tool names, parameters, file paths, URLs, database ops, output schemas, transaction values. Never pipe model output directly into a shell, DB, or privileged API.
Require confirmation for consequential actions	Human approval for money, permissions, external comms, record deletion, sensitive-data transmission, legal/medical decisions, physical systems. Describe the actual action — not an opaque agent plan.
Secure retrieval at the data layer	Apply document-level authorization before retrieval. Never send unauthorized documents to the model expecting it will not quote them.
Monitor behaviour, not only text	Telemetry must include: prompt/response metadata · retrieval sources · tool-call requests and outcomes · auth context · model/prompt versions · token/cost anomalies · agent-loop depth · high-risk action attempts.
Use defence in depth	Identity controls + least privilege + content controls + retrieval authorization + tool validation + sandboxing + rate limits + human approval + monitoring + incident response + regular regression red-teaming.

8 OPEN-SOURCE TOOLING

Microsoft PyRIT	Automates generative-AI red-team workflows; records results.
NVIDIA garak	LLM vulnerability scanner across multiple probe categories.
Giskard	Evaluation and testing of ML and LLM applications.
Promptfoo	Repeatable prompt/model evaluations incl. adversarial suites.
IBM ART	Adversarial Robustness Toolbox for ML research and testing.
MITRE ATLAS	Knowledge base of adversarial threats specific to AI systems.

■ **Automation improves coverage but does not replace expert analysis. The strongest engagements combine automated probing with manual reasoning, architecture review, code review, and abuse-case analysis.**

9 KEY TAKEAWAYS

- ✓ Test only with explicit authorization.
- ✓ Map the entire system and all trust boundaries — not just the chat interface.
- ✓ Treat every model input and output as untrusted.
- ✓ Never use the model as the sole authorization mechanism.
- ✓ Apply access controls **before** retrieval, not after.
- ✓ Give agents minimal privileges; validate every tool call deterministically.
- ✓ Measure repeated outcomes and report success rates, not binary pass/fail.
- ✓ Assess business impact, not merely unusual or offensive text.
- ✓ Retest continuously as models, prompts, and integrations change.

Central lesson: An AI model may be helpful and aligned under normal conditions while remaining unreliable under adversarial pressure. Secure deployment depends not on perfect model behaviour, but on architecture that anticipates model failure and contains its consequences.