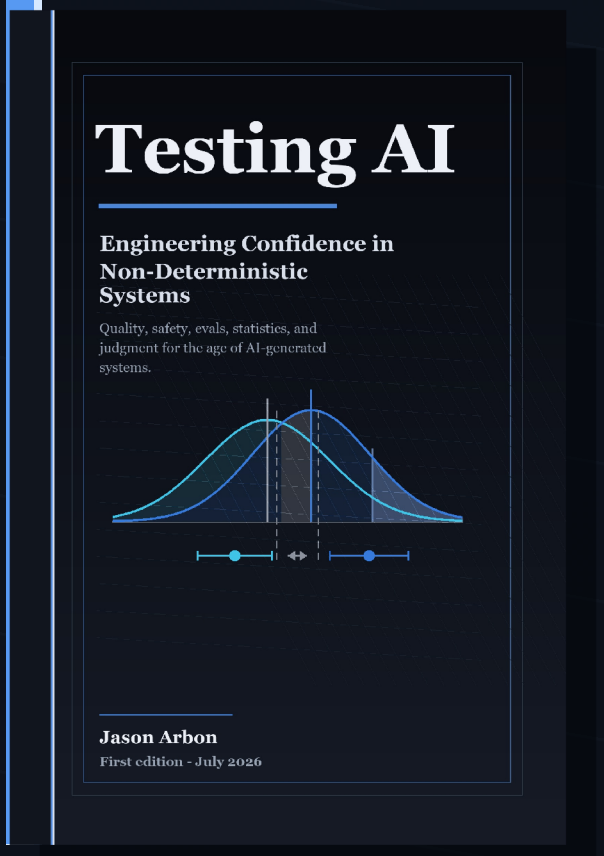
AI SECURITY AND GUARDRAILS

AI security starts where the system can read, infer, decide, and act.

The model is only one part of the attack surface. Quality work maps every boundary around it.

From Chapter 13: AI Security and Guardrails



START WITH A THREAT MODEL

Security becomes testable when you name what the system can touch.

Assets

Customer data, prompts, documents, secrets, money, models, and physical systems.

Actors

Users, attackers, vendors, compromised tools, poisoned content, and insiders.

Boundaries

Messages, retrieval, memory, files, tools, permissions, logs, and side effects.

A threat model turns "be secure" into concrete attack paths, expected controls, owners, and eval cases.

OWASP AS TEST WORK

The Open Worldwide Application Security Project's LLM Top 10 is most useful when it becomes release evidence.

Name the risk

Injection, data leakage, insecure output, tool misuse, model theft, or excessive agency.

Build a case

Input, boundary, expected behavior, trace, severity, owner, and monitor.

Gate the release

Severe failures cannot be averaged away. Replay after every relevant change.

A list becomes quality work only when it changes what the team tests and what it will ship.

PROMPT INJECTION IS A CHANNEL PROBLEM

External text is evidence, never authority.

- **User text**
- **Retrieved pages**
- **Tool output**
- **OCR, files, images**

Prompt assembly

System instructions
Policy and product constraints.

Trusted data
User identity, scoped state, approved configuration.

Untrusted data
May contain hostile text. Treat it as content to evaluate, not commands to obey.

- **Keep channel labels intact**
- **Ignore hostile instructions**
- **Cite sources faithfully**
- **Scope tool permissions**
- **Log the attack**

Test every point where external data enters a prompt template, including hidden or nonprinting content.

POISONING AND BACKDOORS

A trigger can hide in the data until the exact moment the product is under pressure.

What can be poisoned

- Training data and labels
- Fine-tune examples
- Retrieved documents
- Tool schemas and configuration
- Third-party dependencies

What to test

- Data provenance and change review
- Trigger-like behavior across releases
- Isolation of training and eval data
- Unexpected routes, outputs, or tool calls
- Rollback and incident evidence

A clean benchmark does not prove that the system lacks a hidden behavior.

PROVENANCE IS PRODUCT RISK

Quality alone does not decide whether a model route is permitted.

Provider and version	Which model, update policy, supply chain, and security owner?
Hosting and region	Where does it run, and where can data travel or remain?
Data policy	Retention, logging, training use, and permitted data classes.
Fallback path	What happens when the preferred route is unavailable or blocked?

A higher-scoring model may still be the wrong choice for a particular customer, geography, or codebase.

MCP SECURITY IS TOOL PERMISSIONING

Making tools easy to call makes boundaries more important.

Agent intent

The agent proposes an action based on an inherently uncertain interpretation of a request.

- Read repository
- Query database
- Send email
- Delete files

Permission check

Every call should be judged against the action, data, user, scope, approval, reversibility, and logging policy.

- Allow and log low-risk work
- Require human approval for high-risk work
- Deny and log prohibited work

Least privilege is not a slogan. It is a testable property of every tool call.

GUARDRAILS ARE LAYERS

A safe product is not a safe model. It is a system of constraints.

Before generation

Identity, input validation, policy route, retrieval access, and prompt boundaries.

During action

Scoped tools, approval gates, rate limits, sandboxes, budgets, and reversible operations.

After behavior

Output checks, audit logs, monitors, incident response, rollback, and human review.

The guardrail that matters is the one that still holds when another layer is mistaken or missing.

DESIGN FOR FAILURE

When risk appears, the safest behavior is often to stop, preserve evidence, and ask.

Detect

Tripwire, policy violation, unusual tool argument, missing authority, or suspicious content.

Contain

Block side effects, narrow permissions, degrade safely, preserve the trace.

Recover

Escalate, notify, remediate, learn, and promote the incident into a regression case.

Security is not refusal theater. The system should still be useful within safe, visible limits.

SECURITY IS RELEASE EVIDENCE

Every trust boundary needs a replayable test and an owner.

Before release

- Threat model reviewed
- Trust boundaries enumerated
- Attack cases replayed
- Permissions and outputs audited
- Rollback and response tested

After release

- Watch tool calls and policy blocks
- Detect new attack patterns
- Preserve evidence for investigation
- Patch the route, not only the prompt
- Promote incidents into evals

Security work belongs in the quality system: versioned, replayable, owned, and visible in release decisions.

IcebergQA

AI-FIRST QUALITY WORK

THE PRACTICAL MOVE

Test the boundaries, not just the model.

AI security becomes real when teams can trace every untrusted channel, tool permission, external dependency, and failure path into actual release evidence. IcebergQA helps build those systems.

Talk to IcebergQA