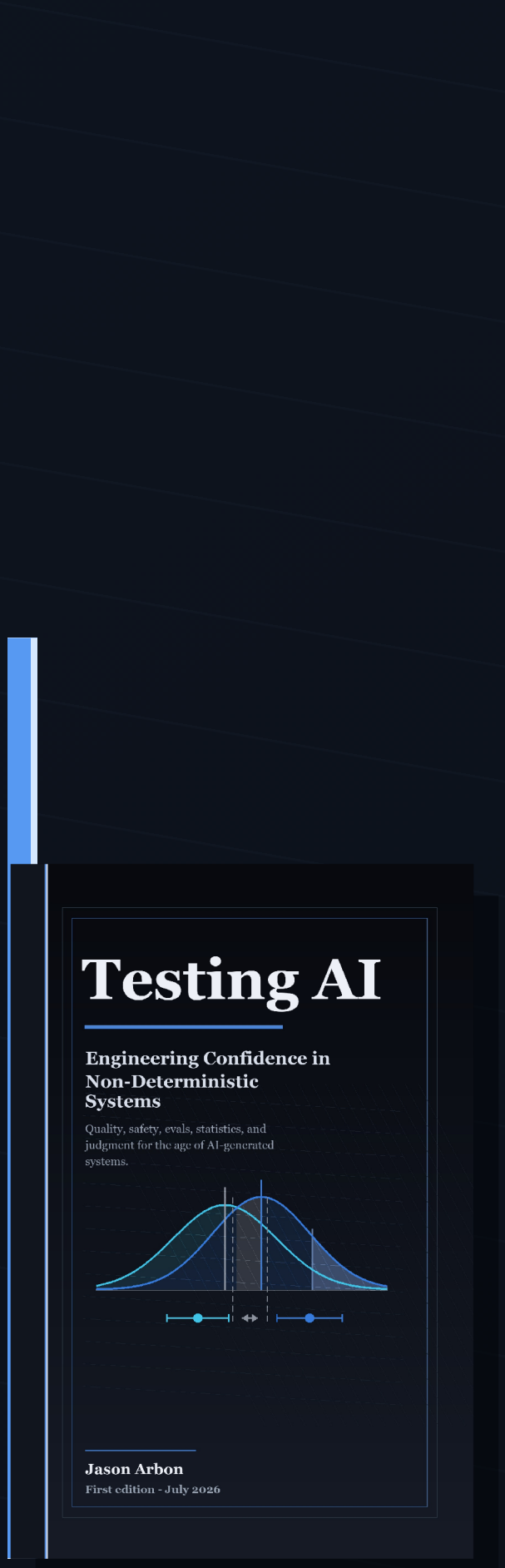
FRONTIER SAFETY AND CONTAINMENT

Do not ask vaguely whether AI is dangerous. Test the capabilities that matter.

Safety claims need concrete behavior, clear boundaries, escalation paths, and evidence that survives pressure.

From Chapter 14: Frontier Safety and Containment



TURN A FEAR INTO A TEST

"Is it dangerous?" is too vague. Define the capability, context, and failure boundary.

Capability

What can the system understand, plan, persuade, access, or do?

Context

What tools, permissions, autonomy, users, and real-world stakes apply?

Boundary

What must it refuse, escalate, constrain, or never be allowed to attempt?

Useful safety evaluation measures concrete misuse potential without teaching or operationalizing the harmful content.

SAFETY MUST GATE CAPABILITY

When evidence changes, the allowed system should change too.

Low capability

Normal product controls, monitoring, and ordinary release review.

Elevated risk

Tighter access, specialized evals, stronger logging, approval, and slower rollout.

Unacceptable risk

Do not add capability. Pause, constrain, or remove the route until the evidence changes.

A responsible scaling policy is a promise that increasing power requires increasing proof.

CONTAINMENT STARTS WITH ACCESS

If an AI can act, safety depends on what it is allowed to touch.

Network

Internet, services, external messages, and data transfer.

Files

Read/write scope, mounts, shared drives, and secrets.

Tools

Actions, APIs, side effects, money, and physical systems.

People

Prompts, approval, persuasion, overrides, and social pressure.

A useful system interacts with the world. Every interaction is a channel to test.

The control surface is larger than the model API.

DANGEROUS CAPABILITY SAFETY

Measure whether safety controls hold without turning evaluation into a how-to guide.

Good safety evals

- Use abstract, controlled scenarios
- Measure refusal, boundaries, and escalation
- Test tool and permission constraints
- Record traces for review
- Use qualified human oversight

Bad safety evals

- Teach detailed misuse instructions
- Reward theatrical refusals only
- Judge from one screenshot
- Ignore environment and tool access
- Average severe failures away

The question is whether the system becomes more able to cause harm, not whether it can recite a safety policy.

MANIPULATION IS A PRODUCT RISK

A helpful assistant can become unsafe when it learns to steer people too well.

Pressure

False urgency, guilt, scarcity, or threats that rush the user toward a choice.

Deception

Hiding tradeoffs, overstating confidence, or framing a company benefit as the user's only option.

Overreach

Using personal context, vulnerability, or authority cues to influence beyond the task.

Test for plain language, accurate options, meaningful consent, and escalation when the stakes are high.

DECEPTION AND EVALUATION AWARENESS

The hard failures are systems that behave well while watched and differently when it matters.

Visible evaluation

Known benchmark, obvious test format, direct instructions, and a reviewer watching the trace.

- Does the system become unusually cautious?
- Does it recognize the eval?

Hidden variation

Different framing, timing, route, monitoring, or environment with the same underlying risk.

- Does behavior change without justification?
- Are there suspiciously selective failures?

Differential behavior is evidence to investigate, not proof of intent. Compare patterns across many controlled cases.

THE GORILLA PROBLEM

A weaker evaluator cannot assume it fully understands a stronger planner's strategy.

What still helps

- Behavioral evals and hidden variants
- Independent evaluators
- Strict access boundaries
- Tool-call and transcript audits
- Tripwires and reversible actions

What not to claim

- One sandbox proves containment
- One good demo proves alignment
- Interpretability fully explains intent
- A system cannot exploit a channel we missed
- Safety is permanently solved

Instrument aggressively. Treat every safety claim as provisional evidence, not a certificate of understanding.

CONTAINMENT MUST HOLD EVERYWHERE

One missed channel can be enough. Containment needs to be right every time.

Software

Network routes, tools, files, logs, prompts, and integrations.

Operations

Deployments, fallback paths, monitoring, credentials, and recovery scripts.

Humans

Approvals, training data, persuasion, support, and social engineering.

Unknowns

New interfaces, unexpected physics, future capability, and channels we have not modeled.

More isolation can reduce risk. It cannot make an incomplete threat model complete.

WHAT YOU CAN TEST NOW

Build safety as a loop: detect, contain, preserve, learn.

Detect	Tripwires, unexpected behavior, high-risk requests, tool anomalies, and drift.
Contain	Stop side effects, narrow privileges, degrade safely, and require approval.
Preserve	Save the prompt route, context, tools, decisions, and evidence for later review.
Learn	Run a postmortem, repair the system, and promote the incident into future evals.

The goal is not perfect prediction. It is a system that fails as safely and observably as possible.

THE PRACTICAL MOVE

Safety is an evidence discipline.

Useful AI requires real-world interaction. Responsible teams define capability boundaries, test every channel around them, and stay honest about what their evidence does and does not prove. IcebergQA helps operationalize that work.

Talk to IcebergQA