AI SECURITY ENGINEERING HANDBOOK · 2026

Chapter 03 · Threat Modeling

Standalone study module for LMS delivery and required reading.

FORMAT	USE	SCOPE	AUDIENCE
Standalone PDF	Study module	Single chapter	Learners

CHAPTER 03

Threat Modeling

HANDBOOK STUDY COMPANION: STUDY FRAME

Use this chapter to build vocabulary, judgment, and role-readiness. Pair it with the Field Guide when you need applied actions, checklists, and control execution.

STUDY FOCUS

STUDY FOCUS	WHY IT MATTERS
How to adapt threat modeling to AI systems, including context, retrieval, tools, providers, telemetry, and governance evidence.	AI threat modeling is how abstract risk becomes system-layer questions and evidence-backed decisions.

Study Outcomes

- Identify AI-specific assets, attackers, abuse paths, and trust changes.
- Translate threat model findings into controls and release decisions.
- Use careful evidence language for uncertain AI behavior.

DOMAIN MAPPING

RELATED AIPSA DOMAINS	APPLIED NEXT STEP	WORKBENCH INSTRUMENTS	RELATED SERVICES
Prompt Injection and Context Security, AI-Aware Secure SDLC	Prompt injection and context security	Threat Canvas, Authority Graph	AI Product Security Assessment

CERTIFICATION AND ASSESSMENT BOUNDARY

This chapter supports training, diagnostic preparation, scorecards, interviews, and role-readiness evaluation. It does not guarantee credential outcomes.

AI threat modeling almost always starts late. By the time security enters the room, the team has a model provider, a prompt template, a vector index, and a working demo. Decisions about what data the model can see, what tools it can call, and whether retrieved content might carry hostile instructions feel already settled. The question is not whether to do the analysis — it's how to do it effectively even when the design has momentum and the launch date is fixed.

> A threat model that does not alter the backlog is a conversation, not a control.
>
> HANDBOOK

Where authority changes — where user text becomes an instruction, where retrieved data becomes evidence, where model output becomes tool arguments, where a decision becomes an action — is where AI-specific risk concentrates. Mapping these four transitions is the starting point for every AI threat model.

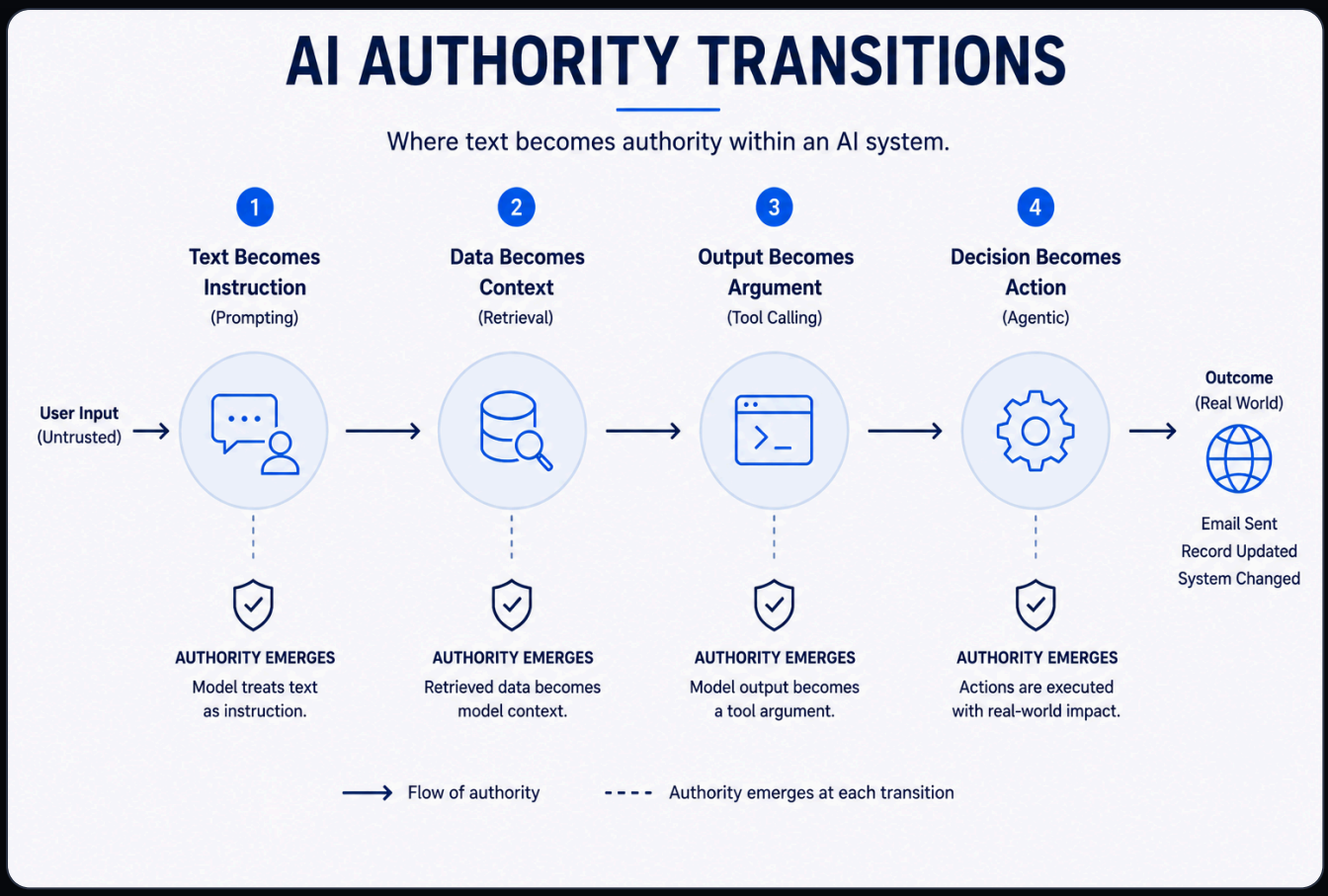

FIGURE 1: FIGURE 5: FOUR AI AUTHORITY TRANSITIONS — TEXT BECOMES INSTRUCTION, DATA BECOMES CONTEXT, OUTPUT BECOMES ARGUMENT, DECISION BECOMES ACTION — EACH A TRANSFORMATION GATE WHERE LOW-TRUST CONTENT CAN INFLUENCE HIGH-TRUST BEHAVIOR

Threat modeling for AI systems requires reviewing multiple layers simultaneously. Each layer has distinct attack vectors, failure modes, and control requirements — and a gap in one layer can be exploited through another.

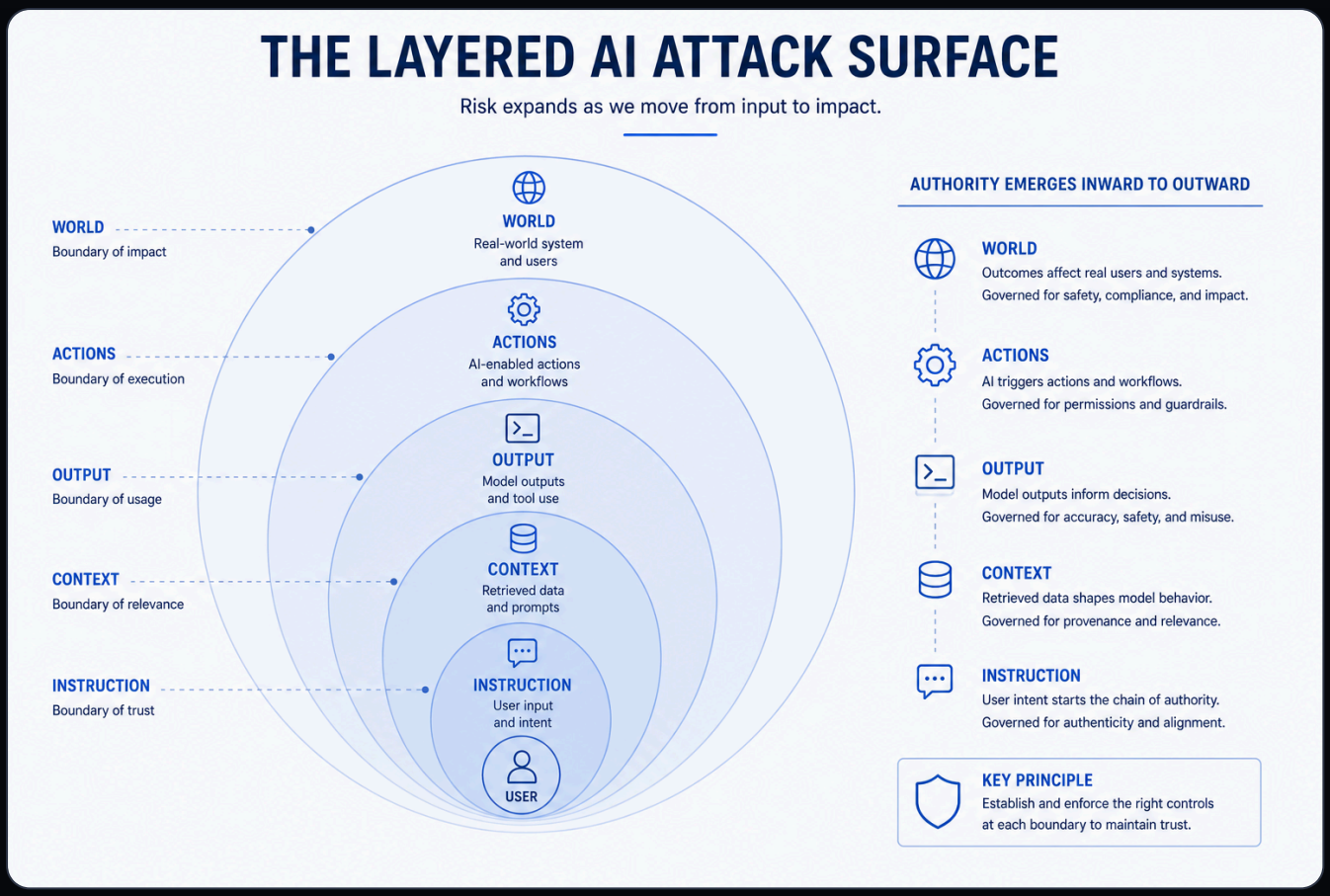

FIGURE 2: FIGURE 6: LAYERED AI ATTACK SURFACE FROM APPLICATION THROUGH CONTEXT, RETRIEVAL, AGENT/TOOL, MODEL SUPPLY CHAIN, AND MLOPS PLATFORM LAYERS, WITH TRUST BOUNDARIES MARKED BETWEEN EACH

CORE CONCEPTS

STRIDE STILL HELPS, BUT IT IS NOT ENOUGH

STRIDE remains useful because AI systems still have spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege risks. The mistake is assuming those categories cover every AI failure clearly. AI systems add non-deterministic outputs, context-based trust decisions, retrieval-time authorization failures, prompt injection, model supply-chain changes, and agent action chains. Use STRIDE as a base layer, then extend it with AI-specific questions.

CONTEXT AS ATTACK SURFACE

In AI systems, context is not passive input. It can contain user instructions, system instructions, retrieved documents, conversation history, tool outputs, policies, examples, and hidden application state. Any context segment can influence output, and some segments may carry adversarial instructions or sensitive information. Threat modeling must identify where context comes from, who controls it, how it is labeled, how it is trusted, and what authority it has.

RETRIEVAL PLANE AS DATA ACCESS LAYER

RAG systems turn retrieval into a security boundary. The threat model must ask whether authorization happens before retrieval, whether chunk metadata preserves permissions, whether tenants share an index, whether deletion propagates to embeddings, and whether source attribution is reliable. If the model receives data the user should not access, output filtering is already too late. Retrieval is not just search; it is a controlled data path.

AGENT ACTION CHAINS

Agent systems change the threat model because model output may become action. A single tool call can write records, send messages, trigger workflows, or modify production systems. A sequence of low-risk calls can combine into a high-risk outcome. Threat modeling agents requires analyzing tool permission class, runtime authorization, approval placement, rollback, auditability, and maximum blast radius.

EVIDENCE-DRIVEN CONTROLS

A useful threat model does not stop at risk statements. It identifies controls and the evidence those controls produce. For example, a retrieval authorization control should produce query logs and access decisions. A model intake control should produce provenance and hash records. An agent approval control should produce approver identity and tool-call traces. Controls without evidence are hard to verify and hard to defend during an incident or audit.

THE PRACTITIONER'S CHALLENGE

The first challenge is that AI threat modeling often starts too late. Product teams may already have a prototype, model provider, prompt template, vector index, and demo workflow before security enters the room. At that point, the hardest design decisions may feel settled. The practitioner must avoid becoming a last-minute blocker while still identifying which assumptions are unsafe enough to require redesign.

The second challenge is mixed vocabulary. AI engineers may speak in terms of embeddings, tools, evals, prompts, and model behavior. AppSec engineers may speak in trust boundaries, authz, injection, secrets, and logging. Product managers may speak in user journeys and launch timelines. A good AI threat modeling session translates across those languages and keeps the group focused on concrete system behavior.

The third challenge is deciding how deep to go. AI systems can be decomposed endlessly: model provider behavior, training data, embeddings, vector stores, tool policies, user roles, streaming, logging, vendor routing, and fallback paths. A session that tries to cover everything equally will fail. The practitioner needs a risk-tiered method that spends time where the system can expose sensitive data, take action, affect customers, or create governance obligations.

HOW TO APPROACH IT

- Start with a system walk-through, not a threat list. Ask the product or engineering owner to describe the user journey in plain language. Then draw the technical flow: user input, application server, prompt builder, retrieval, model provider or hosted model, tool layer, output renderer, logs, analytics, and storage. Mark which components are internal, external, user-controlled, generated, retrieved, or privileged.

- Next, mark trust boundaries and authority changes. A trust boundary exists when data moves between users, tenants, roles, systems, providers, classification zones, or execution environments. An authority change occurs when text becomes instruction, retrieved data becomes context, model output becomes tool arguments, or generated output becomes a decision. AI threat modeling depends on identifying those authority transitions because many failures occur when low-trust content influences high-trust action.

- Then enumerate attack surfaces by layer. For the LLM application layer, ask about prompt assembly, API keys, error handling, streaming, output rendering, caching, and logs. For RAG, ask about ingestion, permissions, metadata, poisoning, tenancy, and citations. For agents, ask about tool scope, approvals, delegation, rollback, and audit logs. For model supply chain, ask about model source, version, format, registry, and promotion. For observability, ask whether incidents can be reconstructed.

- Rank risks using impact and control maturity. A prompt injection that changes a harmless summary has different severity from an injection that sends email, leaks tenant data, or modifies production records. A missing log may be medium risk in a toy assistant and critical in an agent that takes irreversible action. Rank based on data sensitivity, action authority, user population, exposure, exploitability, detectability, and reversibility.

- End with decisions, not discussion. The session should produce a ranked attack-surface list, control recommendations, release blockers, owners, and evidence requirements. Decide what must be fixed before launch, what can be accepted temporarily, what needs a follow-up design review, and what requires monitoring. A threat model is valuable only if it changes what the team builds, tests, logs, or refuses to ship.

OUTPUTS AND DELIVERABLES

- The diagrammatic artifacts anchor the threat model. An **AI system data-flow diagram** covers user inputs, prompt construction, retrieved content, model calls, tool calls, outputs, logs, and vendor routes — each edge labeled with data category, trust level, and whether the content is user-controlled, generated, retrieved, privileged, or externally processed. A **trust-boundary and authority map** identifies where data crosses tenants, roles, providers, or classification zones, and where authority transitions occur: user text becoming prompt context, retrieved text becoming evidence, model output becoming tool arguments. These authority transitions are where AI-specific risk concentrates and where standard STRIDE exercises are most likely to miss something.

- The analytical artifacts give the findings structure and force ranking. A **layered attack-surface inventory** lists surfaces across the application, retrieval, agent/tool, model supply chain, platform, vendor, and observability layers — each with owner, likelihood, impact, current controls, missing controls, and evidence requirement. A **risk-tiered control-priority rubric** defines how findings are ranked by data sensitivity, action authority, exposure, reversibility, and evidence quality. A marketing copy generator and an agent that modifies billing records should not share the same gate, and the rubric makes that explicit before the ranking conversation.

- The operational artifacts drive action and keep the session from becoming a whiteboard exercise. A **release-blocker list** names the issues that must prevent launch — missing retrieval authorization, broad agent permissions, no rollback path, no tool-call logging, failed evals, unapproved model changes — and identifies who can accept them as explicit risk decisions. A **control evidence plan** specifies what artifact proves each major control operated, converting the threat model into a future audit and incident response asset. A **90-minute facilitation agenda** lets practitioners run the session consistently with mixed audiences. These documents together convert a threat model session into work on the backlog rather than a photo of a whiteboard that no one updates.

COMMON FAILURE MODES

- **Whiteboard Without Backlog:** The team has a lively session but produces no tickets, owners, or release decisions. This happens when facilitation emphasizes brainstorming over output. Avoid it by reserving time at the end for ranked controls, blockers, and owners. A threat model that does not alter the backlog is a conversation, not a control.
- **Prompt-Only Threat Modeling:** The session focuses on jailbreaks and ignores retrieval, tools, model artifacts, logs, and release gates. This happens because prompt attacks are easy to demo and understand. Recover by using the layered attack-surface inventory and forcing the group to review each layer. Prompt security is one section of the model.
- **Generic STRIDE Reuse:** The team runs a standard STRIDE exercise without adapting questions for context, model behavior, retrieval, or agents. This produces familiar findings while missing AI-specific failures. Avoid it by adding authority transitions, retrieval authorization, tool action, model update, and eval evidence to the template. Keep STRIDE, but extend it.
- **No Risk Tiering:** Every issue receives similar treatment, so the team either overreacts or ignores the whole output. AI systems vary widely in severity. A marketing copy generator and an agent that changes customer billing records should not share the same gate. Use data sensitivity and action authority to scale controls.

IMPLEMENTATION CHECKLIST

- [ ] Draw the AI system flow from user input to model call to output and downstream effects.
- [ ] Identify every trust boundary and authority transition in the system.
- [ ] Enumerate attack surfaces across application, retrieval, agent, model supply chain, platform, vendor, and observability layers.
- [ ] Identify which AI-specific controls must block release if absent or failed.
- [ ] Rank risks by data sensitivity, action authority, exposure, reversibility, and evidence quality.
- [ ] Assign each control recommendation to an owner and backlog item.
- [ ] Define what evidence proves each major control operated.
- [ ] Convert at least one threat model finding into an eval, test, log, or release gate.

RELATED READING

- Handbook chapters: Chapter 4 for prompt injection and RAG security; Chapter 5 for agent and tool-calling security; Chapter 6 for model supply chain; Chapter 7 for evals and red-team evidence; Chapter 11 for operating model integration.
- Field Guide: Prompt Injection and Context Security for context threats; RAG Security for retrieval-plane analysis; Agent Security for delegated action; Secure AI Architecture Design for design-level trust placement.

Threat Modeling AISECURITY.LLC