AI SECURITY ENGINEERING HANDBOOK · 2026

Chapter 10 · Logging and Telemetry

Standalone study module for LMS delivery and required reading.

FORMAT	USE	SCOPE	AUDIENCE
Standalone PDF	Study module	Single chapter	Learners

CHAPTER 10

Logging and Telemetry

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
Prompt context logs, retrieval traces, tool-call records, model versions, output logs, evidence retention, and telemetry completeness.	AI incidents, eval findings, and governance claims collapse when teams cannot reconstruct what happened.

Study Outcomes

- Name the telemetry required for AI detection, forensics, and evidence.
- Explain log minimization and sensitive-data handling tradeoffs.
- Connect telemetry fields to investigations and control evidence.

DOMAIN MAPPING

RELATED AIPSA DOMAINS	APPLIED NEXT STEP	WORKBENCH INSTRUMENTS	RELATED SERVICES
Incident Response and AI Observability	AI governance, risk, and compliance	Runtime Proxy, Scorecard diagnostic	AI Security Maturity Benchmark

CERTIFICATION AND ASSESSMENT BOUNDARY

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

The question that most AI incident investigations cannot answer is: "What was the model given?" Standard application logs capture what the user sent and what the system returned. AI incident investigations also need what the system assembled and sent to the model — the complete context including retrieved documents, conversation history, system instructions, and tool outputs that shaped the model's response. That gap between what happened at the network layer and what happened in the context window is where most AI incident investigations stall.

That gap between what happened at the network layer and what happened in the context window is where most AI incident investigations stall.

HANDBOOK

CORE CONCEPTS

FULL-STACK AI TRACE

An AI trace documents the complete path from user request to model response: user identity, session identifier, tenant, prompt template version, assembled context (or a hash reference to it), retrieval query and results, model provider and version, model call parameters, model response, output filter decisions, tool calls with arguments and results, approval decisions, final rendered output, and downstream state changes. All elements share a correlation identifier that makes it possible to reconstruct a complete session from logs. Without this linkage, incident investigations reconcile separate log streams manually and often fail to establish causality.

PROMPT AND CONTEXT LOGGING POLICY

Raw prompt content is the most complete forensic evidence for AI incidents and also a source of privacy risk: it may contain personal data, credentials inadvertently pasted into context, confidential business information, and regulated data categories. A prompt logging policy defines three tiers: metadata-only logging (session ID, template version, retrieval source IDs, model version, output hash), redacted content logging (prompt structure with sensitive field values replaced), and full content logging under restricted access. Each tier specifies the trigger conditions, data classification requirements, access controls, retention period, and break-glass procedure.

RETRIEVAL TRACE DESIGN

For RAG systems, the generated answer is the least informative artifact for incident investigation. Retrieval traces must record: the retrieval query that was executed, the filters applied, the chunk identifiers and similarity scores returned, the source document identifiers, the authorization decision for each chunk, and whether each retrieved chunk appeared in the final assembled context. Without retrieval traces, investigators cannot determine whether an incident is a generation problem, a retrieval authorization failure, a retrieval poisoning event, or a source attribution error.

TOOL-CALL AND AGENT ACTION LOGGING

Agent systems require a complete audit trail at each tool call: tool name, proposed arguments, authorization policy decision, approval decision and approver identity if applicable, execution result, target resource, reversibility classification, any side effects, and downstream state changes. Each tool call record must link to the model call that generated it through the shared correlation identifier. This chain of records makes it possible to determine whether an agent action was user-directed, model-inferred, influenced by retrieved content, or policy-permitted — which is the essential question in most agent security incidents.

TELEMETRY VALIDATION

A telemetry design that looks complete in documentation may have gaps in implementation. Telemetry validation involves running specific incident scenarios against the system and verifying

that the telemetry produced would be sufficient to investigate the scenario. The validation exercise asks: given only the logs this system produces, can I answer who initiated the session, what context was assembled, what the model received, what tools were called, and what the user saw? If any answer is missing or requires manual reconstruction from multiple unlinked sources, the telemetry has a gap.

THE PRACTITIONER'S CHALLENGE

The political challenge is that comprehensive AI telemetry appears to conflict with privacy obligations. Teams that log raw prompt content for forensic purposes create a concentrated store of sensitive data. Teams that minimize logging for privacy leave incident investigations blind. The practitioner must design tiered telemetry that provides forensic sufficiency for high-risk workflows while minimizing data collection for lower-risk ones — and must do so before the first production deployment rather than as a post-incident remediation.

The structural challenge is that AI telemetry crosses multiple systems. The application emits request logs. The retrieval platform emits query and retrieval logs. The model provider may emit API call metadata. The tool layer emits action logs. The output filter emits decision logs. Without explicit design to correlate these streams with a shared identifier, incident investigation requires reconstructing causality manually across systems that may have different retention policies and different access controls.

The technical challenge is that streaming responses and partial outputs create telemetry gaps that standard logging does not handle. A streaming response that exposes sensitive content before the complete response is filtered or logged requires either output buffering with full-response capture before delivery, partial-output capture at configurable intervals, or pre-emission validation for high-risk contexts. Logging the final output state does not capture what the user received through the streaming channel.

HOW TO APPROACH IT

- Start with a forensic sufficiency analysis before designing the telemetry stack. Define the AI-specific incidents most likely to occur in the system — prompt injection through retrieval, unauthorized agent action, cross-tenant data access, model behavioral anomaly — and identify exactly which log records would be required to investigate each. This analysis produces the telemetry requirements that the system must implement before launch. Telemetry gaps identified in the analysis become engineering requirements, not post-incident lessons.

- Define the trace schema before implementing any logging. The schema should specify: all required fields for each event type (request, retrieval, model call, tool call, output), the shared correlation identifier format, the format for sensitive field handling (hash vs. redact vs. restrict), the metadata fields that must appear in every event, and the linkage between parent and child events in agent workflows. The schema is a security artifact; it should be reviewed by security and privacy together, not only by the engineering team implementing it.

- Write the prompt logging policy as a prerequisite for enabling any logging. The policy should define: what system types fall into each sensitivity tier, what fields are redacted in each tier, who can access raw logs in the highest sensitivity tier, what the retention period is for each tier, and how the break-glass access procedure works. The policy must be reviewed by privacy counsel before the logging infrastructure is deployed. Retroactively classifying and restricting logs already in production is significantly harder than designing the classification upfront.

- Design retrieval traces as a separate concern from application request logs. Retrieval traces are the most forensically important logs for RAG systems but are the most commonly missing from standard application instrumentation. The retrieval trace pipeline must emit chunk-level records that include authorization decisions, source identifiers, and similarity scores — not just the final generated answer. These records should be retained at least as long as the application request logs they correspond to.

- Validate the telemetry design through incident simulation before launch. Run three tabletop scenarios: a prompt injection through a retrieved document, a cross-tenant retrieval attempt, and an unauthorized agent tool call. For each scenario, walk through exactly which log records would be generated, what information each provides, and what questions about the incident remain unanswerable. Gaps identified in the simulation become engineering tasks before the system goes to production.

OUTPUTS AND DELIVERABLES

- The design artifacts are the **AI trace schema**, **event type specification for each system component**, and **correlation identifier design**. The trace schema defines all fields for all event types with types, required/optional status, and sensitive field handling. The event type specification covers request, retrieval, model call, tool call, output, and approval event types. The correlation identifier design ensures events from different system components can be linked into a complete session trace.

- The policy artifacts are the **prompt logging policy**, **sensitive telemetry access control specification**, and **retention schedule by data classification**. The logging policy defines sensitivity tiers, trigger conditions, redaction rules, and break-glass procedures. The access control specification defines who can access each tier, what logging is required for access, and how access is reviewed. The retention schedule maps data classification to retention periods across all telemetry types.

- The validation artifacts are the **telemetry completeness checklist**, **incident simulation exercise results**, and **telemetry gap remediation record**. The completeness checklist tests each event type against forensic requirements. The simulation results document the outcome of pre-launch tabletop exercises. The gap remediation record tracks identified telemetry gaps to engineering completion before production deployment.

COMMON FAILURE MODES

- **Analytics-Only Instrumentation:** The system emits telemetry designed for product analytics — sessions, responses, user satisfaction — while missing the forensic context required for security investigation. No retrieval traces, no prompt context records, no tool-call audit logs. Fix: treat forensic sufficiency as a launch prerequisite; run the telemetry validation exercise before production deployment.

- **Prompt Log Sprawl:** Comprehensive prompt logging is enabled for debugging and never reviewed, classified, or restricted. Over time, the logs become a sensitive data store with broad engineer access and undefined retention. Fix: write the prompt logging policy before enabling any logging; classify and restrict logs from the first record.

- **Correlation Gap:** Application logs, retrieval logs, model API logs, and tool logs are stored in separate systems with different identifiers and no shared correlation key. Incident investigation requires manual reconciliation across systems. Fix: design the shared correlation identifier and trace linkage as a required element of the telemetry architecture before implementing any component.

- **Streaming Blindspot:** The telemetry captures the complete buffered output but not what was delivered to the user through the streaming channel before the output was complete. Incidents that involve partial output exposure are systematically under-reported. Fix: add pre-emission validation or partial-output capture for high-risk contexts before enabling streaming output.

IMPLEMENTATION CHECKLIST

- [ ] Define the minimum telemetry required to investigate each primary AI incident type before designing the stack.
- [ ] Define the AI trace schema with all required fields and the shared correlation identifier format.
- [ ] Write the prompt logging policy before enabling any logging, with sensitivity tiers and access controls.
- [ ] Design retrieval traces as a separate instrumentation concern with chunk-level authorization records.
- [ ] Specify tool-call audit log fields for agent systems with parent-child event linkage.
- [ ] Validate the telemetry design through incident simulation before launch.
- [ ] Implement sensitive telemetry access controls with audit logging for access to high-sensitivity tiers.
- [ ] Define retention schedules by data classification across all telemetry types.

RELATED READING

- Handbook chapters: Chapter 11 (Detection Engineering) for using telemetry in detection rules; Chapter 12 (Incident Response) for using telemetry in incident investigation; Chapter 7 (Data Exposure and Privacy) for sensitive data handling in prompt logs.
- Field Guide: Incident Response and AI Observability for trace sufficiency checks, forensic reconstruction, and sensitive telemetry handling.