AI SECURITY ENGINEERING HANDBOOK · 2026

Chapter 12 · Incident Response

Standalone study module for LMS delivery and required reading.

FORMAT	USE	SCOPE	AUDIENCE
Standalone PDF	Study module	Single chapter	Learners

CHAPTER 12

Incident Response

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
AI incident classification, context-chain reconstruction, containment actions, forensic evidence, and post-incident control improvement.	AI incidents often involve prompt, retrieval, tool, model, provider, and telemetry layers at the same time.

Study Outcomes

- Classify AI incidents by failure class and affected boundary.
- Explain containment options for retrieval, agents, providers, and prompts.
- Describe the evidence needed to reconstruct an AI incident.

DOMAIN MAPPING

RELATED AIPSA DOMAINS	APPLIED NEXT STEP	WORKBENCH INSTRUMENTS	RELATED SERVICES
Incident Response and AI Observability	Incident response and observability	Runtime Proxy, Threat Canvas	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.

Containment decisions made without context are guesses with consequences. AI incident response differs from conventional incident response in one critical structural way: scope and severity depend on dynamic context state, not only on code version or deployment history. A prompt injection incident may affect only the sessions that retrieved a specific poisoned document during a specific time window. A retrieval authorization failure may affect only users in a specific tenant who queried while the index was in a specific state. A model update regression may affect only requests that matched a specific behavioral pattern after a provider-side routing change. Scope determination requires querying against context-aware telemetry — not counting records modified since the last deploy.

Containment decisions made without context are guesses with consequences. Scope determination requires querying against context-aware telemetry — not counting records modified since the last deploy.

HANDBOOK

FIELD USE

Use this chapter before an incident. Name the failure class. Name the evidence source. Bound the scope. Choose containment. Record the decision. Turn the finding into a control change.

TRIAGE RULE

Do not classify from output alone. Rebuild the context chain first. Check retrieval. Check tools. Check approvals. Check model changes. If evidence is missing, widen scope. Record the gap.

CORE CONCEPTS

AI INCIDENT CLASSIFICATION

AI incidents fall into identifiable failure classes that determine the investigation path and containment options. Classifying before containing prevents wasted effort on actions that address the consequence while leaving the mechanism intact. The primary failure classes are: prompt injection incidents, where adversarial context influences model behavior or tool calls; retrieval authorization incidents, where unauthorized data enters the context window; agent unauthorized action incidents, where tool calls exceed user-authorized scope or are influenced by injected context; model behavioral regression incidents, where provider or version changes alter security-relevant behavior; and supply chain incidents, where compromised or unauthorized model artifacts affect production behavior. Each class has different evidence requirements, different containment logic, and different remediation scope.

SCOPE DETERMINATION FROM CONTEXT-AWARE TELEMETRY

Scope in AI incidents is a query against telemetry, not a timestamp since the last deployment. Scoping a retrieval authorization incident requires querying retrieval logs for the affected source document, the time window it was active in the index, and which users received it in context. Scoping a prompt injection incident requires tracing which sessions retrieved the poisoned document, what actions followed in those sessions, and whether any tool calls or outputs resulted from the injection that require notification. Scoping a model behavioral regression requires identifying the provider routing change timestamp and the query patterns that triggered the affected behavior. When telemetry gaps prevent accurate scope determination, the scope is widened to the boundary of available evidence, and the gap is documented as a contributing factor in the incident record.

AI-SPECIFIC CONTAINMENT ACTIONS

Standard containment — blocking network addresses, revoking credentials, rolling back code — is necessary but not sufficient for AI incidents. AI-specific containment actions include: removing a poisoned document from the retrieval corpus and triggering index rebuild, suspending a prompt template version and reverting to a prior approved version, disabling a specific agent tool or connector without disabling the entire agent system, revoking an agent's OAuth token for an external integration, switching to a pinned model version from a prior provider routing configuration, invalidating cached response sets from a specific time window, and disabling streaming for high-risk contexts pending investigation. These actions require playbooks and operational procedures defined before incidents occur. Improvised containment under pressure produces incomplete remediation.

FORENSIC RECONSTRUCTION FOR AI INCIDENTS

AI incident forensics requires reconstructing the complete context chain: what the user requested, what context was assembled, which documents were retrieved and from which sources, what the model received, what tools were called with what arguments, what was approved, and what the user saw. This reconstruction depends on retrieval traces, prompt context logs, model call records,

tool-call audit trails, approval records, and output logs — all linked by shared correlation identifiers. Without this chain, investigators can describe the consequence but not the mechanism. Root cause analysis that terminates at the output layer cannot produce accurate remediation or reliable regression test cases.

POST-INCIDENT CONTROL IMPROVEMENT

An AI incident that closes without improving detection, telemetry, or architectural controls is a missed opportunity. Post-incident review for AI incidents should produce specific improvements with named owners and defined timelines: a new detection rule written with test cases, a specific telemetry field added to the trace schema, a retrieval authorization control strengthened, a prompt template change reviewed and approved, a model intake requirement added, or an architectural change with threat model justification. Recommendations documented in narrative form but not tracked to completion produce the same incident twice.

THE PRACTITIONER'S CHALLENGE

The political challenge is classification pressure. When a senior stakeholder asks "was this a security incident?" during an active investigation, the answer must be grounded in evidence rather than in what the stakeholder wants to hear. Premature classification in either direction — over-escalating a product quality defect as a security incident, or under-scoping a retrieval authorization failure as a model quality issue — produces incorrect notification decisions, wasted investigation effort, and damaged credibility when the accurate classification emerges. The practitioner must hold the classification question open until the context chain evidence supports an answer.

The structural challenge is that AI incident response requires coordination across teams that do not normally operate under incident pressure together. The investigation requires telemetry access from platform engineering, model version metadata from the AI team, retrieval corpus access from data platform, vendor communication for provider-side incidents, and legal review for notification obligations — all in parallel, under time pressure. Without pre-established roles, escalation paths, and access procedures, coordination overhead consumes investigation time. Incident response playbooks must define not only what to do but who does it and what they need access to.

The technical challenge is distinguishing failure classes that produce similar output characteristics. A retrieval authorization failure, a prompt injection through retrieval, and a model hallucination can all produce an answer that contains sensitive or unexpected content. Distinguishing between them requires the context chain: did unauthorized content enter the context window before the answer was generated? Were the retrieved documents authorized for this user? Was there content in the retrieved chunks that directed the model's behavior? This distinction determines the remediation scope, the notification obligation, and the post-incident control improvement. Teams that do not have the telemetry to make this distinction cannot close the incident accurately.

HOW TO APPROACH IT

- Build AI incident response playbooks for each primary failure class before incidents occur. Each playbook should name the triggering detection signal or escalation path, the immediate containment actions for that failure class, the evidence sources required for scope determination, the forensic reconstruction steps, the AI-specific remediation actions, the stakeholder notification criteria and timeline, and the post-incident control improvement checklist. Playbooks should be reviewed by the teams that will execute them and tested through tabletop exercises at least annually. A playbook that has not been exercised will not perform correctly under pressure.

- Define scope determination procedures using retrieval and context traces as the primary evidence source. For each primary failure class, specify: which telemetry queries determine the affected user population, what fields are required to bound the time window, how missing telemetry changes the scope estimate, and what the decision rule is for widening scope when evidence is incomplete. When retrieval traces are not available for a time window, assume the scope includes all users who queried during that period rather than assuming the absence of evidence means absence of impact. Document the telemetry gap as a contributing factor and add it to the post-incident engineering backlog.

- Verify that AI-specific containment actions are operational capabilities before they are needed. The on-call team should know how to: remove a specific document from the retrieval corpus and trigger a targeted index rebuild with confirmed completion, suspend a prompt template version and revert to a prior approved version, disable a specific agent tool connector without affecting unrelated tools, rollback to a pinned model version from a prior provider routing configuration, and invalidate a specific cached response set. Document the exact commands, access requirements, and confirmation steps for each action. Verify that access controls allow on-call responders to execute containment without requiring approval chains that extend the containment window.

- Apply classification rigor during triage. Before determining the investigation path, answer: did the output result from a control failure, or did the system perform as designed and produce an unexpected outcome? If there was a control failure, which class? Use the context chain to answer — not the output content alone. A compliant-looking output can still result from a retrieval authorization failure. An incorrect output may be a model quality issue rather than a security failure. Getting classification right determines everything downstream: the investigation approach, the containment actions, the notification obligations, and the post-incident remediation scope.

- Conduct post-incident reviews that produce specific, tracked improvements with named owners. The review should cover: what detection rule would have caught this failure earlier, what telemetry field or trace type would have made scope determination faster, what architectural or process change would reduce the probability of recurrence, and what playbook update is required. Each improvement is expressed as an engineering artifact — a detection rule with test cases, a trace field

specification, an architecture change with threat model justification — assigned to an owner with a completion date. The incident is formally closed after improvements are complete or explicitly accepted as deferred risk with a documented owner and timeline.

OUTPUTS AND DELIVERABLES

- The playbook artifacts are the **AI incident response playbooks by failure class**, **containment action runbooks**, and **stakeholder notification decision tree**. Playbooks cover each primary failure class with triggering signals, investigation steps, containment actions, scope determination procedures, and post-incident improvement checklist. Containment runbooks document the exact operational steps for each AI-specific containment action. The notification decision tree maps incident classification and severity to notification obligations and timelines.

- The investigation artifacts are the **AI incident forensic reconstruction template**, **scope determination telemetry query library**, and **incident record template**. The forensic template defines the context chain fields to reconstruct for each failure class. The query library contains the retrieval and context trace queries used to bound scope for each failure class. The incident record template captures classification, evidence sources, scope determination, containment actions, stakeholder notifications, root cause, and improvement tracking.

- The improvement artifacts are the **post-incident review template**, **control improvement tracking record**, and **playbook update log**. The review template structures the feedback loop between incident findings and detection, telemetry, and architectural improvements. The tracking record connects each improvement to the incident that produced it and records completion status. The playbook update log documents changes made to playbooks following incidents.

COMMON FAILURE MODES

- **Scope Underestimation from Telemetry Gaps:** The incident appears contained to one session because the telemetry does not have retrieval traces or context assembly records for other sessions. The organization communicates a contained incident while the actual scope remains unknown. When broader scope emerges later, the resulting communication problem is worse than a more conservative initial estimate would have produced. Fix: when telemetry is incomplete, widen scope to the evidence boundary; document telemetry gaps as contributing factors and add them to the engineering backlog.

- **Session-Level Containment of a Corpus-Level Problem:** A prompt injection through a poisoned retrieval document is identified, the session is terminated, and the incident is closed. The poisoned document remains in the retrieval index. Future users who query with semantically similar terms retrieve the poisoned content into their context, and the injection risk persists. Fix: verify that containment actions address the persistence mechanism, not only the immediate session; for retrieval injection incidents, containment is not complete until the source document is removed and the index is rebuilt with confirmed propagation.

- **Misclassification as Model Quality Issue:** A retrieval authorization failure or a prompt injection event produces an unusual or inaccurate answer and is classified as a model hallucination or output quality problem. The investigation stops at the output layer without asking what the model received, whether unauthorized data entered context, or whether a control failed. Remediation targets model quality while the security failure remains unaddressed. Fix: require context chain reconstruction before classification; classification based on output characteristics alone without examining what the model was given is incomplete triage.

- **Post-Incident Review Without Tracked Improvements:** The incident is investigated, root cause is documented, and the immediate vulnerability is remediated. The post-incident review produces a narrative and architectural recommendations. Neither detection engineering nor platform engineering receives a specific ticket with an owner and timeline. The next occurrence of the same failure class is detected by its consequences again. Fix: define the review protocol to produce specific engineering artifacts — detection rules with test cases, telemetry trace specifications, architecture changes — assigned to named owners with defined completion dates before the incident is closed.

IMPLEMENTATION CHECKLIST

- [ ] Write AI incident response playbooks for each primary failure class before deployment.
- [ ] Define scope determination procedures using retrieval traces and context assembly records for each failure class.
- [ ] Verify that AI-specific containment actions are operational capabilities with documented runbooks and tested access procedures.
- [ ] Test playbooks through tabletop exercises annually and after significant architectural changes.
- [ ] Require context chain reconstruction before finalizing incident classification.
- [ ] Define the stakeholder notification decision tree with classification and severity criteria mapped to obligations and timelines.
- [ ] Define the post-incident review protocol to produce specific engineering artifacts with named owners and completion dates.
- [ ] Track post-incident improvements to completion before formally closing the incident record.

RELATED READING

- Handbook chapters: Chapter 10 (Logging and Telemetry) for the context-aware telemetry required for scope determination and forensic reconstruction; Chapter 11 (Detection Engineering) for the detection rules and feedback loop that feeds AI incident response; Chapter 5 (RAG Authorization) for retrieval corpus remediation following authorization failures.
- Field Guide: Incident Response and AI Observability for incident classification, scope checks, containment actions, and post-incident evidence.