Incident Response Preparation for AI Systems

Robert Saul
General Manager, AWS Customer Incident Response Team (CIRT)

May 14, 2025

aws security

Patterns in AI Security Incidents

The **AWS CIRT** supports customers responding to security incidents on the customer side of the shared responsibility model, 24/7, globally.

When AI workloads are involved, the teams who respond fastest share a common trait: they prepared differently. Not more tools, not bigger teams — better preparation in three specific areas.

Three opportunities that separate good response from great response:

Non-Human Identity Governance
Pre-Positioned Telemetry
Containment Decision Frameworks

aws security

NHI: The Gap, The Target State, The Path

- Where we see opportunities:
  AI systems create NHIs (service accounts, API keys, OAuth tokens, agent-to-agent trust, pipeline credentials, orchestration secrets) that never appear in IR asset inventories. When compromised, teams cannot identify blast radius, baseline behavior, or revocation path — and critically, cannot answer "what business function just lost its underlying trust layer?"

- What good looks like:
  A living inventory of every AI-created NHI mapped to the business function it supports, with permissions, behavior baselines, ownership, and a tested revocation playbook that accounts for business impact.

- How to get there:

  1. Map AI systems to the business functions they serve
  2. Catalog the six NHI categories each system creates
  3. Document blast radius for each identity in business terms (not just technical scope)
  4. Build revocation playbooks prioritized by business criticality (who, how, validation)
  5. Assign a human owner to every NHI —understands both the technical and business context
  6. Tabletop simulations

Telemetry: The Gap, The Target State, The Path

- Where we see opportunities:
  Customers do not have the AI-specific evidence trail: prompt/completion logs, model version history, training data access, embedding queries, agent decision traces, drift metrics. If not collected before the incident, it does not exist. And without mapping logs to business functions, teams can't prioritize which gaps matter most

- What good looks like:
  Every AI log source identified, collected, retained, and mapped to both investigation questions and the business function it protects — so analysts know what they're defending, not just what they're querying.

- How to get there:

  1. Start with business functions: which AI systems support revenue, customer trust, compliance, operations?
  2. Map each system's components to its evidence sources and retention policy
  3. Identify gaps (prompt logs, model lineage, agent traces) — prioritize by business impact of the blind spot
  4. Build a log source reference card: business function → investigation question → source → query
  5. Validate retention covers your mean-time-to-detect for business-critical systems first
  6. Practice: simulate a scenario against your highest-value business function, have analysts pull the evidence

Containment: The Gap, The Target State, The Path

- Where we see opportunities:

  Customers improvise containment during the incident. Killing an agent corrupts downstream state. Rolling back a model breaks dependent services. Isolating a vector store takes down production. The containment action causes more damage than the attacker — because no one mapped the technical components to the business functions they support before the decision had to be made.

- What good looks like:

  Pre-built decision trees per AI component anchored to business functions: what does this system do for the business, what are the containment options, what's the business cost of each option, and who has authority to accept that cost? Validated with IR, ML engineers, and business owners together.

- How to get there:

  1. Map AI components to the business functions they enable — start with revenue and customer-facing systems
  2. Define containment options per component (revoke, rollback, isolate, disable, fallback)
  3. Document cascading effects in business terms
  4. Build ordered decision sequences per scenario type, ranked by business exposure
  5. Define approval authorities based on business impact thresholds, not just technical severity
  6. Tabletop with ML engineers and business owners present

Governance Starts With One System

Governance isn't a policy document collecting dust in a wiki. It's a living map of every AI system's identities, data flows, and failure modes — maintained before you need it.

Our mental model: treat every AI system as its own blast radius. If you can't describe its identities, its data flows, and its failure modes right now, you don't have governance over it. You have hope.

Pick one AI system in your environment today and build its governance record:

1. Catalog its non-human identities — execution roles, API keys, service accounts, inter-service trust chains, delegation tokens
2. Map its data flows end to end — training pipelines, RAG sources, inference I/O, feedback loops, sensitivity classifications
3. Document its failure modes — how it degrades, what drift looks like, where compromise hides behind "normal" behavior
4. Build one containment decision tree — what can you isolate without breaking production? What's the cost of doing nothing?

This is the work. Not after the incident. Not during the incident. Before. The teams that respond well aren't faster — they did the governance work when it was boring.

Thank You

Robert Saul (saulrobe@amazon.com)
General Manager, AWS Customer Incident Response Team (CIRT)

aws security