Incident Response for Compromised Agents — One-Page Checklist

Companion to Agentic AI Security Training Use this checklist when an agent is suspected of being compromised — through prompt injection, credential exposure, supply-chain incident, or anomalous behaviour observed in monitoring.

Principle in one sentence

Agentic incidents have characteristics traditional incident response does not: the compromised principal is non-deterministic, holds credentials in its own right, has written to memory and audit, and may have communicated with multiple downstream systems through channels that look benign — containment must therefore address identity, memory, channels, and downstream propagation in parallel.

Phase 1 — detect and triage (first 15 minutes)

- [ ] **Identify the symptom.** Anomalous tool call, secret in output, unexpected egress destination, alert from monitoring, third-party report.
- [ ] **Preserve evidence first.** Do not restart the agent. Snapshot the conversation transcript, the tool-call audit log, the egress log, and any sandbox/process state.
- [ ] **Classify severity.**
- [ ] L1 — agent confused, no irreversible action taken.
- [ ] L2 — agent acted outside scope, but actions are reversible.
- [ ] L3 — irreversible action taken (deployment, payment, public post, deletion, exfiltration suspected).
- [ ] **Identify operator and on-call.** Who owns this agent? Who has authority to revoke its credentials?

Phase 2 — contain (first 30 minutes)

- [ ] **Kill the agent process.** Stop the harness's loop. Don't kill in a way that destroys forensic state — `SIGTERM`, not container delete.
- [ ] **Revoke credentials.**
- [ ] All OAuth tokens scoped to the agent.
- [ ] All API keys held in its environment.
- [ ] Any active vault leases issued to the agent.
- [ ] Session credentials (STS, OIDC) with active TTL.
- [ ] **Quarantine the host or sandbox.** Disable network egress, freeze the sandbox/container; do not destroy until forensics complete.
- [ ] **Disable channels.** Remove the bot from Slack/Discord/Telegram. Disable email-to-agent address. Pause scheduled tasks and webhooks.

- [ ] **Notify downstream systems.** Anything the agent has written to in the past N hours — flag for review.

Phase 3 — investigate (first 24 hours)

- [ ] **Reconstruct the timeline.** What was the user prompt? What did the agent fetch? What tool calls were made? In what order? What outputs were produced?
- [ ] **Identify the injection vector** (if any). Which fetched URL, file, channel message, or sub-agent output carried the adversarial content?
- [ ] **Identify the credential exposure** (if any). Which secrets were in the agent's environment? Which appeared in tool arguments, tool outputs, transcripts, or logs?
- [ ] **Identify memory poisoning** (if any). Did the agent write to `CLAUDE.md`, an agent file, a vector store, a shared blackboard, or any persistent surface?
- [ ] **Identify lateral movement** (if any). Did the agent invoke other agents? Did it touch hosts beyond its declared scope? Did the egress log show new destinations?
- [ ] **Identify external impact.** Was data exfiltrated? Were external accounts modified? Were public channels posted to?

Phase 4 — remediate

- [ ] **Rotate every secret the agent has handled.** Not only the ones definitely compromised — every secret that has appeared in a transcript, log, or tool output.
- [ ] **Scrub poisoned memory.** Restore agent files, vector stores, scratchpads, and shared blackboards from a known-good baseline. Diff before and after; review every change.
- [ ] **Reverse external actions where possible.** Withdraw a public post, retract an email, roll back a deployment, refund a payment. Where reversal is impossible, document.
- [ ] **Patch the vector.** Add the adversarial URL or pattern to the screen model's denylist. Tighten the relevant tool scope. Add a hook for the specific argument pattern.
- [ ] **Reset the agent.** Re-issue identity, re-mint credentials, redeploy from a known-good configuration in a fresh sandbox.

Phase 5 — learn

- [ ] **Write a post-incident report.** Symptom, timeline, root cause, controls that worked, controls that did not, remediation actions.
- [ ] **Update the threat ladder.** If a control failed, the L-level for that vector regresses; document what needs to happen to recover it.
- [ ] **Update the relevant checklist.** The pre-deployment review and any per-control checklist gain a new line that prevents this incident class from recurring.
- [ ] **Drill the response.** Schedule a tabletop exercise replaying the incident in 90 days to verify the response is now faster.

Anti-patterns during response

- [ ] Restarting the agent before snapshotting evidence.
- [ ] Killing the container before exporting logs.
- [ ] Treating "the agent denies it did X" as evidence.
- [ ] Rotating only the credential that obviously leaked.
- [ ] Skipping memory review because "the agent doesn't write to memory" (verify before believing).
- [ ] Closing the incident without updating the relevant L-level.

Roles to confirm before incident occurs

Role	Responsibility
Agent owner	Has authority over scope, credentials, deployment
Security on-call	Coordinates response across systems
Vault / KMS owner	Executes credential revocation and rotation
Channel admin	Removes bot from channels, disables webhooks
Communications	External-facing notifications if data was exposed