althor.dev

PRODUCTION INFRASTRUCTURE FOR AI AGENT SYSTEMS

Agent Security Review Checklist

A pre-flight checklist for shipping AI agents into regulated environments.

AUTHOR Samuel · Althor Consulting

VERSION v1.0 · April 2026

WEBSITE althor.dev

Run this checklist against any agent system that touches production data, customer information, or financial records — before InfoSec runs it against you.

How to use this checklist

Run this against any agent system that touches production data, customer information, or financial records — before InfoSec runs it against you.

Each layer has 5–7 yes/no items. Answer honestly. Anything below 80% per layer means that layer is the one that fails review.

If you're missing a whole layer, jump to the "Decision tree" on the last page. Most agent projects fail review because of a missing layer, not weak controls within one.

The five-layer pattern this checklist runs against is documented at length in Making agent deployments pass security review. Read that first if you haven't.

Layer 1 · Identity

> Every agent action is attributed to a specific actor.

#	CHECK	YES	NO	PARTIAL
1.1	The agent runs under a workload identity (machine account / managed identity / workload identity federation), not a developer's personal credentials	☐	☐	☐
1.2	User-triggered agent actions use on-behalf-of (OBO) delegation back to the invoking user's identity, not the agent's identity	☐	☐	☐
1.3	The workload identity is registered in your IdP (Entra ID / Okta / Workspace) with the principle of least privilege	☐	☐	☐
1.4	Service principals or app registrations have an owner tagged who is responsible for periodic review	☐	☐	☐
1.5	The agent's identity does not have admin or owner-level roles on any system	☐	☐	☐
1.6	Identity rotation procedure is documented and tested	☐	☐	☐

What InfoSec will actually ask:

- "Show me the workload identity. Who owns it?"
- "What roles does it have? In which environments?"
- "When was the last rotation?"

Common gaps:

- The agent runs under a developer's personal token because "we'll fix it before prod" — and never does.

- A single service principal with global admin “for development convenience” was never narrowed.
- The workload identity has the right roles in the right tenants, but no documented owner.

Layer 2 · Credential broker

> Agents never hold long-lived secrets.

#	CHECK	YES	NO	PARTIAL
2.1	Every secret used by the agent is stored in a centralized broker (Key Vault, Vault, Secrets Manager)	☐	☐	☐
2.2	The agent retrieves credentials at runtime via managed identity, not via embedded API keys	☐	☐	☐
2.3	No secrets in source control. No secrets in environment variables on long-running hosts. No secrets in app settings panels	☐	☐	☐
2.4	Tokens minted for downstream tools are short-lived (15 min – 1 hour max) and scoped to the specific call	☐	☐	☐
2.5	Every credential request to the broker is logged with caller identity, scope, and timestamp	☐	☐	☐
2.6	The broker is monitored for anomalous access patterns (volume spikes, unusual scopes, unusual hours)	☐	☐	☐
2.7	Credential rotation is automated; rotation does not require a code or config change	☐	☐	☐

What InfoSec will actually ask:

- “Walk me through how the agent obtains credentials at runtime.”
- “If I compromise the agent’s runtime, what credentials do I get?”
- “Show me the audit log for the broker over the last 24 hours.”

Common gaps:

- Secrets in Key Vault but the agent’s app settings still has the API key as a fallback.
- Credentials are short-lived in theory but the broker is configured to mint 24-hour tokens.
- No alerting on broker anomalies — logs exist but no one’s watching.

Layer 3 · Scoped tool access

> Tools expose the smallest surface the agent needs.

#	CHECK	YES	NO	PARTIAL
3.1	Tools are domain-shaped (`list_open_invoices`), not transport-shaped (`http_get`)	☐	☐	☐
3.2	Read-only tools are clearly distinguished from write tools (naming convention or explicit metadata)	☐	☐	☐
3.3	Every tool has a description that the orchestrator/planner can use to make safe routing decisions	☐	☐	☐
3.4	Tools paginate or truncate large outputs; no unbounded result streams	☐	☐	☐
3.5	Tool errors are returned as structured objects (code, message, suggested action), not raw exceptions	☐	☐	☐
3.6	The agent does not have direct access to admin SDKs or raw database connections	☐	☐	☐
3.7	Adding a new write tool requires a documented review step, not just a code merge	☐	☐	☐

What InfoSec will actually ask:

- “Show me the full list of tools the agent has access to.”
- “Which tools can write? Which can delete? Show me the policy that gates them.”
- “What stops the agent from running an arbitrary SQL query?”

Common gaps:

- A `db_query` tool was added “for flexibility” and gives the agent unrestricted SQL access.
- Tool descriptions are auto-generated from function signatures and convey nothing useful to the orchestrator.
- The MCP server returns full table dumps because pagination wasn’t implemented.

Layer 4 · Policy + approval gating

> Every action goes through a policy check before execution.

#	CHECK	YES	NO	PARTIAL
4.1	A policy engine sits between the agent's plan and tool execution; no tool runs without a policy check	☐	☐	☐
4.2	Safe, reversible actions auto-execute. Risky actions queue for human approval	☐	☐	☐
4.3	The list of "risky actions" is explicit and documented, not implicit in code	☐	☐	☐
4.4	Approvals are first-class workflow objects (queryable state, reviewer, justification, timestamps), not chat messages	☐	☐	☐
4.5	Auto-execution decisions are logged with the policy that approved them, so a reviewer can reconstruct why	☐	☐	☐
4.6	Confidence thresholds (when used) are configurable per action class, not hardcoded globally	☐	☐	☐
4.7	A revocation path exists: the policy can be tightened or the tool surface restricted without a code deploy	☐	☐	☐

What InfoSec will actually ask:

- "Show me a recent action that was auto-executed. Show me the policy decision."
- "What's the approval queue look like? Who's been reviewing?"
- "If the agent tried to issue a refund right now, what would happen?"

Common gaps:

- Approvals "happen in Slack" — meaning they're not queryable, not auditable, not in a system of record.
- The policy engine exists but has been bypassed via a "trusted operator" override that never gets reviewed.
- Confidence thresholds are set so low that everything auto-executes.

Layer 5 · Audit

> Every suggestion, approval, override, and failure produces a structured event.

#	CHECK	YES	NO	PARTIAL
5.1	Every agent decision produces a structured audit event (not just a log line)	☐	☐	☐
5.2	Audit events include: actor identity, tool called, inputs (or input hash if PII), outputs (or output reference), policy decision, timestamp	☐	☐	☐
5.3	Audit events are stored in an append-only or version-controlled store, retained per your data retention policy	☐	☐	☐
5.4	Compliance and incident response query the audit store directly — not the application logs	☐	☐	☐
5.5	Failures (refusals, timeouts, exceptions) are first-class audit events, not just stack traces in logs	☐	☐	☐
5.6	The audit dashboard exists and has been used at least once by a non-engineer (governance, compliance, manager)	☐	☐	☐
5.7	Audit retention meets your industry/regulatory minimum (HIPAA, SOX, PCI, etc.) without manual export	☐	☐	☐

What InfoSec will actually ask:

- “Pull all actions taken on customer X over the last 30 days.”
- “Show me every refused or queued-then-denied action this week.”
- “What’s the retention period? How is it enforced?”

Common gaps:

- “Audit” is just a Sentry feed of stack traces — useful for debugging, useless for compliance.
- Events exist but inputs are recorded raw, including PII, which now creates a different compliance problem.
- The dashboard exists but has never been opened by anyone except its author.

Decision tree — if you're missing a layer

MISSING	SEVERITY	MINIMUM FIX TO SHIP
Identity	🚨 Stop. Don't ship.	Workload identity registered in your IdP, owned by a named person, with least-privilege roles. Should take ~1 day.
Credential broker	🚨 Stop.	Move every secret to your existing broker (Key Vault if Azure, Secrets Manager if AWS, Vault if you have it). Use managed identity to retrieve. ~2–3 days.
Scoped tools	🛑 Hard stop for production; OK for internal pilot	Wrap the underlying SDK calls in domain-shaped tools. Write descriptions. Pagination on every list endpoint. ~1 week.
Policy gating	⚠️ Ship to pilot only	At minimum: a hard-coded "approval required" list for write actions, queue + email notifier. ~3–5 days for a v1.
Audit	⚠️ Ship to pilot only	Structured event emission on every tool call, into your existing log/event store. Build the query interface later. ~2–3 days.

If you're missing two or more layers, you don't have an agent system — you have a script with a chat interface. Don't ship.

What good looks like

The five layers are the minimum bar. Production-grade systems add:

- **Confidence-based auto-apply with cancellable countdown** (3-second window to abort an auto-applied action)
- **Tool surface change detection** (alert when the MCP server adds new tools, especially write tools)
- **Per-environment policy** (the same agent has different policies in dev vs. staging vs. prod)
- **Reversible action preference** (the agent prefers a propose-then-confirm pattern over direct mutation when both are available)
- **Read-only by default** (every new tool starts read-only; adding a write capability is an explicit action)

When to bring in outside help

If you've completed this checklist honestly and three or more layers came back red — or if your security review is in less than four weeks and you don't have time to fix all five — that's the moment to bring in someone who's done this before.

Samuel runs fixed-fee architecture reviews for agent deployments under exactly this constraint. The output is a written report mapping your current state to the five layers, a prioritized remediation list, and a one-page summary you can hand to InfoSec.

BOOK A 30-MIN REVIEW →

OR READ THE LONGER ESSAY →

Samuel · Althor Consulting LLC

althor.dev · contact@althor.dev

Maryland, U.S. · Remote-first