CHECKLIST

Checklist: Securing AI Model APIs with Runtime Monitoring

A practical checklist for validating, monitoring, and responding to risky AI model API behavior in production.

Checklist: How to Secure AI Model APIs with Runtime Monitoring

Use this checklist to assess whether your AI model API has the monitoring, detection, and response controls needed for production use.

1) Define what must be protected

- [ ] Identify every AI model API endpoint in scope.
- [ ] Document which models, tenants, applications, and user groups can call each endpoint.
- [ ] Classify the data the API may process: public, internal, confidential, or regulated.
- [ ] List the highest-risk use cases, such as chat, retrieval-augmented generation, tool use, summarization, or code generation.
- [ ] Define the security and compliance outcomes that monitoring must support.

2) Establish baseline controls before monitoring

- [ ] Require authentication for every request.
- [ ] Enforce authorization at the endpoint, tenant, and role level.
- [ ] Apply request throttling and rate limits.

- [ ] Validate input size, format, and content type.
- [ ] Restrict outbound network access and tool permissions where applicable.
- [ ] Confirm that sensitive data handling rules are documented and enforced.

3) Monitor request-layer signals

- [ ] Log caller identity, tenant, endpoint, timestamp, and request ID.
- [ ] Capture request size, frequency, retries, and burst patterns.
- [ ] Detect prompt injection attempts and instruction override patterns.
- [ ] Flag requests containing secrets, personal data, or suspicious payload structures.
- [ ] Watch for repeated near-duplicate prompts or automated probing.
- [ ] Track unusual geographic or infrastructure source patterns when relevant.

4) Monitor model-behavior signals

- [ ] Validate response format and schema conformance.
- [ ] Detect disallowed content, policy violations, and unsafe instructions.
- [ ] Flag references to internal data, secrets, or unauthorized context.
- [ ] Monitor refusal rates, fallback rates, and safety filter outcomes.
- [ ] Track output length, token usage, and unexpected verbosity changes.
- [ ] Review hallucination indicators when they affect safety or compliance.

5) Monitor service and infrastructure signals

- [ ] Track latency, timeout rates, and error rates by endpoint and tenant.
- [ ] Watch for token consumption spikes and cost anomalies.
- [ ] Monitor throttling events and blocked request counts.
- [ ] Correlate model activity with downstream access patterns.

- [ ] Alert on sudden changes in traffic volume, topic distribution, or output quality.
- [ ] Include telemetry from gateways, wrappers, and orchestration layers.

6) Preserve useful telemetry without overexposing data

- [ ] Decide what must be logged in full and what must be redacted.
- [ ] Avoid storing raw prompts and outputs unless there is a clear business need.
- [ ] Prefer hashes, structured summaries, labels, or sampled payloads where possible.
- [ ] Apply retention limits to sensitive monitoring data.
- [ ] Restrict access to logs, traces, and alerts containing model content.
- [ ] Verify that telemetry itself does not create a new data leakage path.

7) Build a clear decision workflow

- [ ] Define which signals are informational, warning, or blocking.
- [ ] Map each alert type to an owner and an escalation path.
- [ ] Decide when to alert, rate limit, quarantine, or require human review.
- [ ] Ensure pre-inference controls can stop unsafe requests before model execution.
- [ ] Ensure post-inference controls can stop unsafe outputs before delivery.
- [ ] Validate that events are emitted by the authoritative control point.

8) Create response playbooks

- [ ] Document the steps for suspected prompt injection.
- [ ] Document the steps for sensitive data exposure.
- [ ] Document the steps for rate abuse or automated probing.
- [ ] Document the steps for policy violations or harmful output.
- [ ] Document the steps for model quality degradation that may indicate drift.

- [ ] Include temporary blocking, scoped throttling, tenant quarantine, and ticketing actions.

9) Test the monitoring controls

- [ ] Simulate prompt injection against a non-production environment.
- [ ] Send requests that exceed expected size, rate, and frequency thresholds.
- [ ] Test secret leakage detection with controlled sample data.
- [ ] Verify schema validation on malformed or manipulated outputs.
- [ ] Confirm alerts trigger at the right severity and reach the right responders.
- [ ] Measure false positives and false negatives before production release.

10) Review runtime monitoring against drift and quality changes

- [ ] Compare monitoring signals with baseline behavior from normal traffic.
- [ ] Check whether refusal rates or topic distributions changed unexpectedly.
- [ ] Correlate drift indicators with security alerts to separate benign change from abuse.
- [ ] Revisit thresholds when traffic mix, tenants, or model versions change.
- [ ] Treat drift as an operational signal, not automatically as a security incident.

11) Verify production readiness

- [ ] Confirm the API can be observed end to end.
- [ ] Confirm alerts are actionable and not just informational.
- [ ] Confirm response teams know what to do when a threshold is crossed.
- [ ] Confirm logging, retention, and privacy requirements are met.
- [ ] Confirm monitoring is enabled before broad user access is granted.

- [ ] Confirm owners review the system regularly after launch.

Quick readiness check

If you can answer yes to all of the following, your AI model API is in a much better position:

- [ ] Can you see who called the API and what context they used?
- [ ] Can you detect prompt abuse, data leakage, and anomalous usage in real time?
- [ ] Can you stop unsafe requests or outputs before they cause harm?
- [ ] Can you correlate model behavior with identity, tenant, and service signals?
- [ ] Can you respond quickly with a clear playbook?

Final note

Runtime monitoring is most effective when it is treated as a production control, not a reporting layer. The goal is to detect risky behavior early, validate model behavior continuously, and give the team a fast path to containment when something changes.