Xiaotime *Labs*

Closed-loop continuous monitoring

Closing the *Loop*

How Xiaotime Labs built the continuous monitoring runtime the industry has been describing for fifteen years.

A position paper for CISOs, heads of platform, and compliance leaders weighing how to operationalize continuous monitoring without buying another tool that stops at the report.

Position paper. The architecture, the argument, and how it operationalizes the standards that already exist.

VERSION 1.0
May 2026

EXECUTIVE SUMMARY

TL;DR

For fifteen years, the cybersecurity industry has known what good continuous monitoring should look like. NIST wrote it down in 2011. The Cybersecurity Framework codified it. HITRUST, ISO 27001, SOC 2, and the new NIST AI Risk Management Framework all describe the same destination from different vantage points. A defender who detects, classifies, decides, and responds to threats fast enough and with enough context that adversaries lose decision dominance.

Nobody built it.

What got built instead is a tool ecosystem. SIEMs that detect. GRC platforms that report. SOAR runbooks that block-and-isolate. Code-scanning tools that find vulnerabilities. Ticketing systems that track them. Humans who spend their weeks ferrying signals between these silos. Each tool solved its slice. None of them owned the integration runtime. The defender's response cycle stayed human-paced and context-fragmented.

ICRG is that integration runtime. It ingests from whatever security tools the customer already runs, projects every observation through every applicable compliance framework simultaneously, advances findings through a closed remediation pipeline that ships actual code changes through actual review gates, and preserves human authority at exactly one place: production-deploy authorization, by deliberate architectural commitment.

This document explains why that combination has been the missing piece, what changes in defender economics when you have it, and where the architectural choices land for a customer evaluating it.

1. The Diagnostic

A defender today operates inside an asymmetry that has not shifted materially since 2011.

On the attacker side, the OODA loop runs at machine pace. Reconnaissance is automated. Vulnerability matching is automated. Exploit selection is automated. Lateral movement is increasingly automated. Exfiltration is automated. The attacker's *Observe, Orient, Decide, Act* cycle closes in minutes-to-hours.

On the defender side, the loop runs at human pace, broken across silos.

- Observe is split across SIEM, EDR, CSPM, code scanners, attack-surface telemetry, secret scanners, and a dozen other panes of glass.
- Orient happens in an analyst's head: fusing signals across those silos, classifying severity, recalling which compliance framework this maps into, checking historical baseline from memory.
- Decide is a triage queue plus a change-approval board, paced in days.
- Act is an engineer-authored patch, an engineer-requested review, an engineer-clicked deploy, paced in days-to-weeks.
- The loop never truly closes. Verification that the fix worked is rarely systematic, and posture state stays frozen between quarterly audits.

Boyd's foundational insight from fighter combat, that the side closing the loop with better orientation at higher velocity wins decision dominance, has been applied honestly to *attack* for over a decade. Defenders have been quoting it for almost as long without operationalizing it.

The gap is not at any single phase. SIEMs handle Observe credibly. SAST and SCA tools detect vulnerabilities credibly. CSPMs surface cloud misconfigurations credibly. GRC platforms aggregate control state credibly. The gap is that **no one assembled these into a single runtime where the loop actually closes**. Findings stop at Step 4 of NIST's ISCM model (Analyze and Report). The Respond step has been a ticket, and the Review-and-Update step has been a slide deck.

That is what ICRG is built to fix.

2. Two Halves of One Insight: Velocity and Fidelity

Loop velocity is the obvious half. If your detect-to-remediate cycle takes weeks while the attacker's exploit window is hours, you lose by default.

Loop fidelity is the half most security vendors duck. A fast loop pointed at bad information just lets you be wrong faster. The reason traditional defenders cannot safely accelerate their cycle is precisely that their orientation is fragmented, framework-blind, and historically uncontextualized. They lack the rich worldview that would make rapid decisions trustworthy.

ICRG is built around the conviction that **both halves must be restored together**. A runtime that automates the loop without lifting orientation quality is a liability. A runtime that improves orientation without accelerating the loop is just a better dashboard. The combination is what changes defender economics.

ICRG's orientation substrate has six components, working as a unified synthesis layer.

1. **Multi-source fusion across the customer's existing stack.** Every observation gets correlated with every other observation across whatever security tools the customer already runs. The customer keeps their tools. The runtime provides the integration layer. A vulnerability surfaced by one scanner, an attack-surface signal surfaced by another, and a misconfiguration surfaced by a third all become facets of the same finding, not eight tickets in eight queues.
2. **Multi-framework projection.** Every finding is simultaneously projected through every compliance framework the organization is bound by. NIST CSF 2.0, NIST 800-53, HITRUST i1, ISO 27001, SOC 2, NIST AI RMF, and additional frameworks as wired. An analyst can hold maybe two of these in working memory at once. The runtime holds all of them, always, in real time.
3. **Historical baseline and drift detection.** Posture snapshots compare current state to prior state per framework, per function, per tenant. Signals get contextualized as deteriorating, stable, or improving before they reach decision. Most defenders see today's alerts without yesterday's baseline.
4. **Knowledge enrichment.** A knowledge-retrieval layer injects relevant standards excerpts, prior decisions on similar findings, and standard operating procedures into the agent stack's working context for every classification. The closest human equivalent is a senior analyst with twenty years of pattern memory. The runtime gives every classification that depth.
5. **Policy contextualization.** A policy engine weighs each finding against severity, blast-radius metadata, tenant risk tolerance, and framework-specific weighting. Priority is computed against organizational context, not raw CVSS scores.
6. **AI synthesis on ambiguous evidence.** The agent stack, with purpose-built personas for audit, maturity coaching, telemetry intelligence, requirements coaching, code review, and knowledge retrieval, produces structured narratives over messy signal, not just rule-matched classifications.

This is where the runtime's value density lives. Velocity is the visible result. Superior orientation is the durable moat.

3. The Closed Loop, Concretely

The runtime is organized as two cooperating subsystems that close the OODA loop end-to-end.

ICRG owns Observe, Orient, and Decide. It ingests telemetry, synthesizes orientation across the six substrates above, advances findings through a policy-driven workflow stage machine, and surfaces remediation hypotheses ready for execution.

AI-SDLC owns Act. It receives the remediation hypothesis as a code change, runs the change through a structured review gate (a dedicated review persona evaluates the diff against six dimensions: security, quality, patterns, tests, dependencies, and architectural fit), and ships the validated change through the customer's existing change-management infrastructure to deployment.

The loop closes back to Observe through post-deploy telemetry confirmation. The runtime watches the same signal that surfaced the finding. When the signal clears, the workflow advances to `deployed_confirmed`. When the watch window expires without the signal clearing, the workflow re-opens as `regressed` and the loop fires again.

The Stage Machine

Every finding moves through an explicit stage progression.

```
finding_surfaced
   → specs_relayed
      → developer_proposed
         → codereview_passed
            → pending_approval    ← Human-in-Loop gate
               → deployed_awaiting_confirmation
                  → deployed_confirmed    OR    regressed
```

Every transition writes an append-only audit row, emits a real-time event stream the dashboard and downstream consumers subscribe to, and updates the running posture snapshot. An auditor asking *"when did this control failure get remediated and how do we know it actually cleared?"* gets a complete timestamped timeline, not a ticket comment thread.

The Human-in-Loop Guarantee

ICRG is autonomous everywhere except one stage: `pending_approval`. The transition from `codereview_passed` to deploy is gated on a human click. By deliberate architectural commitment, not by missing automation.

NAMED COMMITMENT

"Auto-approving without human intervention violates the Human-in-Loop Guarantee. Production deploys require explicit human approval."

From the runtime's own source. The commitment is enforced in code, named in policy, documented for reviewers, and surfaced to operators at every approval surface.

The gate exists because production-deploy authority is the one decision that should never be delegated entirely to an agent stack, no matter how richly oriented. The agent stack has done every cognitive step that traditionally consumed analyst weeks. Fused observation. Classified the finding. Projected it through the customer's compliance frameworks. Weighed it against blast radius and risk tolerance. Authored the remediation code. Reviewed the code for blocking issues. The human's role compresses to one click: *yes, ship this change to production.*

This is Boyd's Implicit Guidance and Control concept made operational. When orientation is rich enough, the right action follows from the worldview directly, and the operator's decision is reflexive validation, not deliberation. The cognitive labor that historically filled the operator's week now lives in the agent stack. The operator's contribution shifts from analysis to authority.

For enterprise buyers asking *"do AI agents deploy code to my production environment without my approval?"* the answer is a direct **No, by design.**

4. Proof: One Remediation Cycle End-to-End

Consider a representative example, traced from detection to confirmation.

A dependency vulnerability scanner identifies a high-severity CVE on a tracked package in the customer's repository and opens a remediation pull request with the version bump applied. The webhook fires the runtime.

Observe. The telemetry event lands. A workflow row is created. The finding is projected into the control failure registry under the supply-chain risk category. Every compliance framework that maps a vulnerability-management control records the event automatically. NIST CSF Detect.Continuous-Monitoring, SOC 2 CC7.1, HITRUST 09.f, ISO 27001 A.12.6.1, NIST 800-53 RA-5.

Orient. The policy engine evaluates severity against the customer's configured threshold and against blast-radius metadata for the affected package. The knowledge layer pre-enriches the

review context with the package's prior CVE history, the customer's standard operating procedure for dependency updates, and the relevant control documentation. The posture snapshot updates.

Decide. Workflow advances to `developer_proposed`. The remediation pull request is now under runtime ownership.

Act. The code review agent reads the diff against six review dimensions and posts a structured verdict comment to the pull request. Clean verdict advances workflow to `codereview_passed`. The notification surface fires. The operator sees the finding in their portal feed labeled awaiting approval.

Human-in-Loop. The operator clicks Approve. The runtime advances workflow to `pending_approval`, approves the pull request via the configured merge identity, and merges. Deploy fires automatically via the customer's existing CI/CD pipeline.

Loop close. Post-deploy telemetry watches. When the next dependency scan reports the CVE absent, workflow advances to `deployed_confirmed`. The posture snapshot updates again, this time reflecting the cleared control failure. The audit canon now carries the complete timeline.

> **Elapsed time from detection to deployed-confirmed:** in the customer's most common case, under one hour. The operator's contribution: approximately one second of conscious decision.
>
> **Compare to the traditional defender's cycle for the same finding class.** Detect (minutes), analyst triage (hours), ticket creation (hours), engineer assigned (days), engineer authors PR (hours-to-days), reviewer assigned (days), reviewer review (hours), merge (hours), deploy (hours), manual retesting (days). Total: typically two-to-six weeks. Operator's contribution: full cognitive engagement across multiple roles.

The cycle hasn't gotten longer because security got harder. The cycle has stayed long because the runtime to compress it didn't exist.

5. The Same Loop, Generalized

The dependency-CVE remediation above is not a special case. It is the canonical shape of the runtime's loop, instantiated against one finding class. The same pipeline shape handles every finding class, distinguished only by the ingester on the front and the fix-generator on the back.

FINDING CLASS	TRIGGER	FIX GENERATOR
Dependency CVE	Dependency-vulnerability scanner	Dependency-bump pull request
Container image CVE	Container-scan tool	Base-image update or package removal
Host operating-system CVE	Distribution security feed	Patch application via configuration change
Secret committed to code	Secret-scanning tool	Rotation plus scrub
External attack surface	Attack-surface monitor	Infrastructure-as-code change
Code-quality pattern	Static-analysis scanner	Pattern-corrected pull request
Posture drift on a control	Maturity engine	Control implementation change
AI governance gap	AI-compliance assessor	Policy or technical control change

One audit canon. One posture engine. One review gate. One human-in-loop authorization. Different ingesters, different fix-generators. Coverage of finding classes is bounded only by which ingesters are wired in a given customer deployment, and the runtime is deliberately source-agnostic, accepting any tool that emits findings as a potential telemetry source. The customer keeps their existing security stack. The runtime provides the integration layer over it.

6. Multi-Framework Projection in Real Time

A second architectural commitment, distinct from the closed loop itself, is that **every observation is simultaneously projected through every applicable compliance framework**.

The traditional pattern is single-framework-at-a-time, human-curated, quarterly. An organization preparing for a SOC 2 audit assembles control evidence in SOC 2 vocabulary. Preparing for HITRUST a quarter later, the same underlying evidence gets re-curated in HITRUST vocabulary. The evidence is the same. The lens is different. The work is done twice.

ICRG inverts this. Evidence is collected once, in a framework-neutral taxonomy. Framework projection happens on read, through crosswalk mappings maintained as a single source of truth. An auditor for HITRUST and an auditor for SOC 2 read the same underlying control state through different lenses, with no re-projection work required at audit time.

For organizations bound by multiple frameworks, the typical enterprise, the typical healthcare provider, the typical financial services firm, this reframes the audit-readiness problem. The Plan of Action and Milestones, the canonical artifact NIST defines for tracking control failures and their remediation, stops being a quarterly spreadsheet compilation and starts being a live query against an evidence layer that already reflects the current state of every framework.

Said simply: the POA&M didn't go away. The POA&M became real-time.

7. Why Existing Architectures Don't Get Here

A reasonable question from a buyer considering ICRG against the broader security-tools ecosystem: *why hasn't this been done already?*

The answer, category by category.

SIEM platforms stop at Step 4 of NIST's ISCM model. They aggregate, correlate, and report. They don't author fixes, don't drive PRs through review gates, and don't close the loop with telemetry-confirmed remediation.

SOAR platforms do automate response, but the response is operational. Block this user. Isolate this host. Run this playbook. The response surface is countermeasures, not root-cause remediation. The control failure that triggered the alert is still present after the SOAR action. The SOAR has bought containment time. ICRG's response is architectural. The patch that closes the underlying weakness ships through change management and deploys.

GRC platforms aggregate control state and report against frameworks, often well. They don't gather data themselves (they rely on connectors and human entry), and they don't remediate. They are the audit-prep layer, not the response layer.

Cloud-native security hubs (the offerings from major cloud providers) close some loops within their own walled gardens. A cloud misconfiguration can be auto-remediated within that provider's surface. They don't extend to code repositories, dependency ecosystems, on-premises systems, or the customer's other clouds.

Code-scanning tools detect vulnerabilities credibly. They don't classify against frameworks, don't aggregate cross-source, don't drive remediation through change management, and don't confirm remediation via telemetry.

Dependency-management tools open remediation pull requests for known dependency CVEs. They don't review, classify into frameworks, gate against policy, or close the loop with confirmation.

Each category solves a slice. None owns the integration runtime. The defender's value problem isn't a missing category. It's the missing assembly. ICRG is that assembly.

8. The Cognitive Economics

The honest pitch for ICRG is not *AI replaces the security team.* It is *AI shifts what the security team's time gets spent on.*

Traditional defender economics:

- 60 to 80 percent of senior analyst time on triage, correlation, ticket queue management
- 10 to 20 percent on actual investigation depth
- 5 to 10 percent on policy, posture, framework-readiness work
- 5 to 10 percent on incident response
- Effectively zero on proactive control improvement

The breadth-of-monitoring problem absorbs the most expensive humans on the team. The senior analyst spends their week handling alerts that an agent with multi-source fusion and framework projection could have triaged in seconds. The investigation depth where humans are uniquely valuable gets squeezed.

ICRG's runtime takes over the breadth problem. The agent stack handles routine cycle compression. The dependency-CVE remediations. The secret rotations. The container-base-image bumps. The attack-surface remediations. The posture-drift responses. The framework-projection work. The senior analyst's time reallocates:

- 5 to 10 percent on approval clicks at the Human-in-Loop gate
- 30 to 50 percent on investigation depth (the work humans are uniquely good at)
- 20 to 30 percent on policy refinement, posture targeting, framework configuration
- 20 to 30 percent on incident response and threat hunting

The team headcount doesn't drop. The work the team does shifts toward the work that actually requires human judgment.

9. What ICRG Doesn't Do

A whitepaper that doesn't enumerate boundaries is selling something else.

ICRG does not:

- Replace existing security tools. It ingests from them.
- Make decisions on zero-telemetry events. If no ingester emits, the runtime has nothing to fuse.
- Substitute for senior analyst judgment on novel threat categories. New attack patterns require human contribution to the taxonomy before the agent stack can incorporate them.
- Deploy to production without human approval. By design.
- Handle every operational countermeasure SOAR platforms handle. The runtime ships fixes, not block-and-isolate playbooks. Operational response remains complementary to architectural response, not replaced by it.
- Generate compliance evidence the underlying telemetry doesn't support. Framework projection is faithful to source data. If the source telemetry has gaps, the projection has gaps.

These boundaries are intentional. Crossing them would either reduce safety (auto-deploy without approval), reduce defensibility under audit (synthesized evidence), or duplicate value better delivered by other categories (operational countermeasures). The runtime is opinionated about what it owns.

10. We Didn't Write Another Standard

The continuous-monitoring problem has not lacked for standards. NIST SP 800–137 described the ISCM process in 2011. NIST CSF 2.0 organized cybersecurity outcomes into six functions across multiple tiers. HITRUST i1 codified controls for healthcare. ISO 27001 codified them for information security broadly. SOC 2 codified them for service-organization assurance. NIST AI RMF codified them for AI-specific risk.

Each of these answered the question *what does good look like.* Each of them, read carefully, describes the same destination. A defender who detects, orients, decides, and responds to threats at a velocity and with a context fidelity that maintains decision dominance over adversaries.

What was missing was the runtime. The standards described the destination. Nobody built the road.

Xiaotime Labs built the road. ICRG is the runtime that operationalizes what the standards have been describing. Closing the OODA loop with rich orientation at machine velocity. Preserving human authority at the one decision that should stay human. Projecting evidence through every framework simultaneously. Delivering audit-grade traceability as the natural exhaust of the system rather than a quarterly compilation project.

We didn't write another standard. We made the best ones actually work.

Appendix A · Standards Mapping

This appendix maps the ICRG runtime against the major continuous-monitoring and control frameworks. Each mapping is summary-level. Deeper per-control mappings are available under technical NDA.

NIST SP 800-137 · Information Security Continuous Monitoring

The 2011 federal continuous-monitoring specification defined six process steps. ICRG operationalizes all six within one runtime.

ISCM STEP	ICRG CAPABILITY
Define ISCM strategy	Tenant-configurable framework targets, multi-framework projection via crosswalk
Establish program	Continuous cadence, no quarterly cycle, telemetry refresh as policy not schedule
Implement program	Telemetry ingestion runtime across the customer's existing security stack
Analyze and report findings	Control failure rollup engine, posture snapshots, real-time event stream
Respond to findings	AI-SDLC pipeline, agent-authored remediation through review gate to deployed-confirmed
Review and update strategy	Drift detection drives posture revision, cross-framework consolidation surfaces fix-once-satisfy-many opportunities

NIST CSF 2.0

The 2024 update added Govern as a sixth function alongside Identify, Protect, Detect, Respond, and Recover. ICRG covers all six.

CSF FUNCTION	ICRG CAPABILITY
Govern	Policy engine plus framework projection plus posture targeting per tenant
Identify	Asset register plus integration catalog plus control catalog
Protect	Control register plus assessment register plus remediation tracking
Detect	Telemetry ingestion across infrastructure, code, dependencies, secrets, attack surface, identity
Respond	AI-SDLC pipeline through Human-in-Loop gate to deployed remediation
Recover	Telemetry-confirmed remediation closure plus posture re-projection

HITRUST i1

HITRUST's CSF i1 implementation level is the most common entry point for healthcare and regulated-industry organizations. ICRG provides the runtime substrate for evidence collection, control implementation tracking, and continuous monitoring against HITRUST controls. Evidence collected in framework-neutral form is projected through the HITRUST lens via the crosswalk mapping. Control failures surface in HITRUST vocabulary alongside their other-framework projections.

ISO 27001

The ISO 27001 control set is one of six framework projections available through the crosswalk. Evidence and findings flow through the same runtime, surfaced in ISO vocabulary for ISO-scoped audits.

SOC 2

The SOC 2 Trust Services Criteria, Security, Availability, Confidentiality, Processing Integrity, Privacy, are projected through the same crosswalk. SOC 2 audit-readiness becomes a query against the live evidence layer rather than a quarterly compilation project.

NIST AI Risk Management Framework

The 2023 AI RMF is supported as a first-class framework projection. ICRG includes AI-specific control categories in the control failure registry. Model inventory. Prompt-injection detection capability. Agent blast-radius mapping. Output-monitoring telemetry. The same closed loop

applies. AI-governance findings flow through detection, projection, policy, remediation, confirmation.

Appendix B · Architectural Commitments

For technical evaluators, the runtime's key architectural commitments.

Source-agnostic ingestion. The runtime accepts findings from any tool that can emit them. The customer's existing security stack is preserved. ICRG provides the integration substrate over it.

Single audit canon. Every state-mutating action writes to one append-only audit table with a chain-of-custody signing layer. Workflow stage transitions, evidence operations, policy decisions, and remediation actions all share one auditable artifact.

Workflow chokepoint. Findings advance through stages via one canonical mutation path. Direct stage manipulation is forbidden by lint enforcement. This guarantees the audit canon and the real-time event stream never disagree with workflow state.

Workspace isolation. Tenant data isolation is enforced at three layers. Application-level filtering. Database row-level security with per-tenant Postgres roles. Per-tenant network scoping. The runtime operates safely in shared-multi-tenant and dedicated-tenant deployment models.

Capability-flagged feature surfaces. Every feature surface is gated by an explicit capability flag readable from a single source of truth, evaluated on every request. New customer onboarding sets flags. Flags are never read from environment configuration at write sites.

Human-in-Loop Guarantee. Production-deploy authorization requires explicit human action by deliberate commitment, documented in the runtime's own source. No code path bypasses this gate.