Overwatch

The Cognitive Monitoring Shield for GOPEL

Proof of Concept: From Security Findings to Operational Protection

A Working Paper to Support AI Provider Plurality

Basil C. Puglisi, MPA
Human-AI Collaboration Strategist | AI Governance Consultant
May 2026

HAIA-RECCLIN Framework Series | Version 2.4 | Proof of Concept

Abstract

Overwatch started with a security concern, not a specification. I had built GOPEL, a non-cognitive governance layer that enforces policy through seven deterministic operations and performs zero cognitive work. That constraint makes GOPEL immune to prompt injection. But I could see the blind spot: attack chains, data poisoning, tool abuse, and multi-turn manipulation all operate through meaning rather than syntax, and a deterministic system cannot evaluate meaning. I needed something that could defend against malicious inputs, learn from the attacks it catches, and adapt as new threats emerge. The initial question was whether this belonged inside GOPEL or outside it.

The answer came from GOPEL's own architecture. GOPEL's internal integrity mechanisms, the sentinel, the dead-man's switch, the hash chain verification, the witness files, all operate inside GOPEL's trust boundary. If GOPEL is compromised, those instruments are compromised. An attacker who replaces GOPEL can replace the sentinel that monitors GOPEL. The Council for Humanity Proposal v1.4 names this directly: a superintelligent system's first move would be to blind the detection system. That forced the architectural decision. This could not be a GOPEL extension. Extensions add capabilities inside the trust boundary. This had to be a separate component watching from outside, with its own specification, its own trust model, and its own relationship to CBG.

The 2026 threat landscape confirmed the gap was urgent. OWASP published the Top 10 for Agentic Applications. NIST opened a Federal Register RFI on AI agent security and received 932 stakeholder comments. MITRE ATLAS expanded to cover agentic kill-chain tactics. The MCPTox benchmark documented a 72.8% attack success rate for tool-description poisoning across 20 prominent LLM agents. NIST's Apostol Vassilev confirmed that finite guardrails will always have adversarial bypass gaps, a limitation rooted in incompleteness theory. The attacks I was concerned with, indirect injection, multi-turn chains, context poisoning, confused deputy exploitation, were not theoretical. They were measured, documented, and actively succeeding against shipping products.

I wrote the specification, built the v1.0 codebase in a single session, then ran it through seven independent AI platform reviews. ChatGPT found 15 structural gaps. MiniMax found 10 operational issues. Grok red-teamed the system with adversarial scenarios designed to break it. Every platform caught something different. Over 60 findings across fourteen versioned releases shaped what Overwatch became. The strategy was deliberate: lock down GOPEL first by fixing all 11 security findings from the formal Security Review, then build the shield around the hardened infrastructure. During that process, I identified the semantic manipulation gap, the threat class where individually clean prompts collectively steer a model toward an unauthorized outcome. That discovery became the v2.3 Trajectory Gatekeeper. One design constraint held from the first conversation through every release: Overwatch does not interfere with GOPEL other than to stop the pipe and route to human.

This paper tells that story. It covers the 2026 threat landscape that made the gap urgent, the security findings that drove the architecture, the 16 operational modules, the Trajectory Gatekeeper for catching semantic manipulation early, the v2.4 Factics Calibration for tamper-evident self-tuning thresholds, and the proof of concept results: 390 automated tests across 15 test suites, all passing, with zero cross-imports between Overwatch and GOPEL verified by static analysis.

Keywords: *Overwatch, GOPEL, AI security, cognitive monitoring, prompt injection, supply chain integrity, semantic manipulation, trajectory gatekeeper, Factics calibration, tamper evidence, HAIA, proof of concept, OWASP agentic, NIST AI RMF, MITRE ATLAS*

1. Why Overwatch Exists

GOPEL was designed to solve one specific problem: build AI governance infrastructure that the AI systems it governs cannot manipulate. The solution is the non-cognitive constraint. Every GOPEL operation is a deterministic mathematical function. SHA-256 hashing. HMAC-SHA256 signing. AES-256 encryption. Set membership testing. Threshold comparison. No interpretation, no evaluation, no judgment. You cannot inject a prompt into a hash function.

That constraint is a real security property, not marketing. Prompt injection, the single most dangerous attack vector against AI systems today, is structurally irrelevant to GOPEL. But the non-cognitive constraint also creates a blind spot. GOPEL can verify that a record was not tampered with, but it cannot evaluate whether the content of that record represents a threat. GOPEL can enforce a policy threshold, but it cannot assess whether a novel pattern falling just below that threshold is the leading edge of a new attack technique. GOPEL can require human approval at an evidence gate, but it cannot judge whether the human providing approval is acting under duress or social engineering.

The first instinct was to build this capability inside GOPEL, as another extension alongside CPE, CICE, and the Post-Quantum Amendment. That instinct was wrong. Extensions add capabilities to GOPEL's deterministic operations. They extend what GOPEL does. What I needed was something GOPEL architecturally cannot do: watch GOPEL from outside GOPEL's trust boundary. The Council for Humanity Proposal v1.4 made this clear. If GOPEL's own integrity controls begin failing across multiple nodes simultaneously, something is attacking the governance infrastructure itself. A superintelligent system's first move would be to blind the detection system. GOPEL's sentinel, dead-man's switch, hash chain verification, and witness files all operate inside GOPEL's trust boundary. An attacker who replaces GOPEL can replace the sentinel that monitors GOPEL.

The analogy that locked the architecture: GOPEL is the airport screening checkpoint. Overwatch is the operations center watching the checkpoint cameras, monitoring badge access to the screening area, flagging when the checkpoint hardware gets physically moved, and tracking whether the officers running the checkpoint are behaving within their training profiles. Different trust boundary. Independent observation. Same governance authority chain through CBG.

I built Overwatch to cover that blind spot. Not by replacing GOPEL or weakening its constraints, but by adding a complementary layer that provides what deterministic enforcement cannot: cognitive threat detection. Overwatch has exactly one action authority: stop the pipe and route to human. It does not fix threats. It does not block transactions. It does not make governance decisions. It watches, and when something looks wrong, it pulls the brake.

2. The 2026 Threat Landscape That Drove Development

Overwatch was not built in a vacuum. The design decisions in this paper trace directly to specific threat research, industry benchmarks, and regulatory signals that converged in early 2026. This section documents the sources that shaped what Overwatch became and why each one matters for the architecture.

2.1 OWASP Top 10 for Agentic Applications (2026)

OWASP published the Top 10 for Agentic Applications specifically addressing the security risks of AI agent systems. Several entries map directly to Overwatch modules. ASI06 (memory and

context poisoning) drove the Context Inspector's provenance tagging and decay enforcement. Goal hijacking drove the Intent Analyzer's scope drift and privilege gradient tracking. Tool misuse drove the Output State Evaluator's role-behavior envelope enforcement. Identity abuse drove the integration with GOPEL's existing operator authentication plus Overwatch's behavioral baseline verification.

2.2 NIST AI Agent Security RFI and COSAiS

NIST published a Federal Register Request for Information on AI agent security that received 932 stakeholder comments, signaling that the federal government recognized agentic AI as a distinct security domain requiring new guidance. NIST's COSAiS (Composable Overlay for Secure AI Systems) initiative began developing agentic use case overlays. Overwatch provides candidate control implementations for those overlays, particularly mapping to NIST SP 800-53 control families: SI (System and Information Integrity) through Overwatch's structural verification, AU (Audit and Accountability) through its independent hash-chained audit trail, and IR (Incident Response) through the escalation engine and independent notification channel.

2.3 NIST Incompleteness Acknowledgment

In March 2026, NIST's Apostol Vassilev confirmed that finite guardrails will always have adversarial bypass gaps, a limitation rooted in incompleteness theory. This is not a GOPEL-specific limitation. It is a mathematical reality for any deterministic defense system. Overwatch does not claim to close this gap. The Factics adaptation loop is asymptotic, continuously narrowing the gap, rather than convergent, claiming to close it. The human checkpoint is the explicit acknowledgment that the gap will always exist and that a named human must stand in it.

2.4 MITRE ATLAS Agentic Expansion

MITRE ATLAS expanded in 2026 to cover agentic kill-chain tactics, providing a structured taxonomy for the multi-step attack patterns that agentic AI systems face. Overwatch's chain signature library maps directly to ATLAS agentic kill-chain tactics. The Execution Graph Engine supports ATLAS technique mapping for incident forensics, and the Trajectory Gatekeeper's convergence detection addresses the multi-turn chain patterns that ATLAS documents.

2.5 MCPTox Benchmark

The MCPTox benchmark documented a 72.8% attack success rate for tool-description poisoning across 20 prominent LLM agents. That number is not a theoretical projection. It is a measured success rate against shipping products. Tool-description poisoning is a confused deputy attack: the agent trusts a tool description that has been tampered with and executes actions the user never authorized. Overwatch's Output State Evaluator catches confused deputy conditions by comparing implied state changes against declared task scope and role-behavior envelopes.

2.6 Industry Recognition of the Gap

The concern was not mine alone. Bessemer Venture Partners identified agentic AI as the defining cybersecurity challenge of 2026, reporting that 48% of cybersecurity professionals identified agentic AI as the single most dangerous attack vector. The average shadow AI breach cost reached $4.63M. Meta released LlamaFirewall as an open-source guardrail system. Microsoft released the Agent Governance Toolkit. Proofpoint launched what it called the industry's first intent-based AI security solution. Akamai and Silmaril both shipped firewall products specifically designed for AI applications. The difference between those approaches

and Overwatch is architectural: most commercial solutions operate inside the system they protect or retrain without human authorization. Overwatch operates from a separate trust boundary and requires human approval for every adaptation through CBG.

2.7 GTG-1002: The First Documented AI-Orchestrated Cyber Espionage Campaign

GTG-1002 used precisely the multi-turn chain technique that GOPEL's per-transaction evaluation cannot catch. Each task appeared legitimate when evaluated in isolation. The attack distributed itself across individually benign interactions that formed a malicious sequence only when viewed together. This is the threat class that drove the Intent Analyzer's rolling intent profile and the Trajectory Gatekeeper's convergence detection. GOPEL evaluates each transaction independently. Overwatch watches the trajectory across transactions.

2.8 EU AI Act Compliance Requirements

The EU AI Act established binding requirements for AI systems operating in the European market. Overwatch supports Article 14 (human oversight) compliance through the CBG escalation model, where every CRITICAL or HALT finding triggers mandatory human review. Overwatch supports Article 12 (record-keeping) compliance through its independent, hash-chained, tamper-evident audit trail that preserves the full evidence chain for every escalation decision.

2.9 The Semantic Manipulation Gap

The attacks that converged in 2026, indirect injection, multi-turn chains, context poisoning, confused deputy exploitation, share a common property: they operate through semantic manipulation rather than structural exploitation. GOPEL's deterministic pattern matching cannot detect them by design. That is not a failure. That is correct behavior for a non-cognitive enforcement layer. The failure would be leaving the gap uncovered. Overwatch exists because the 2026 threat landscape demanded a cognitive monitoring layer that could watch for threats operating through meaning, sitting outside the enforcement layer it protects.

3. The Security Findings That Drove the Architecture

Overwatch did not come from a whiteboard exercise. I built it in direct response to security findings that surfaced during multi-platform code reviews of the GOPEL infrastructure. Seven independent AI platforms reviewed GOPEL across multiple versions and produced over 60 findings that shaped what Overwatch became.

3.1 The Review Cycle

GOPEL went through review by ChatGPT, MiniMax, Grok, Gemini, Kimi, Claude, and DeepSeek across versions 0.5 through 0.6.2. Each platform brought a different analytical lens. ChatGPT identified 15 structural gaps, including missing provenance signing and execution graph tracking. MiniMax found 10 operational issues, including an advisory accumulation bug that was silently preventing proper escalation. Grok red-teamed the system with 10 adversarial scenarios designed to break it. Gemini stress-tested resource boundaries. Kimi audited logging and type safety. Claude served as Navigator, documenting dissent without forcing consensus.

The GOPEL v0.6.1 Security Review (May 2026) formalized the pre-lockdown hardening assessment across all 14 production modules, approximately 8,266 lines of Python. It identified

11 security findings: 2 CRITICAL (thread-unsafe sequence counter in Sentinel, HMAC tag not verified on obfuscation decrypt), 3 HIGH (cross-gate replay of approval signatures, incomplete integrity hash on BreachReport, escalation rule exception crashes), 4 MEDIUM (unauthenticated witness file, encapsulation violations, O(N) list eviction, TOCTOU race in file locking), and 2 LOW/ADVISORY. The review confirmed zero remote code execution, injection, or authentication bypass vulnerabilities. All findings were structural.

3.2 Categories of Findings

The findings clustered into four categories, and each category mapped directly to Overwatch modules. Prompt injection and cognitive manipulation drove the Intent Analyzer and Context Inspector. Supply chain and infrastructure integrity drove the Structural Verifier. Output drift and behavioral anomalies drove the Output State Evaluator. Operational resilience and observability drove the Escalation Engine, Random Audit Generator, and Factics Engine. Nothing in Overwatch was built on speculation. Every module traces back to a real finding from a real review.

3.3 The Agent Operating Model Gap

Under Agent Operating Models 1 and 2, GOPEL provides the governed channel between the human governor and the AI platforms. HAIA-Agent determines how that channel is used. The reduction in human touchpoints under these models creates a security gap that Model 3 (Manual Human AI Governance) does not have, because in Model 3 the human IS the external monitor. Overwatch fills that gap. It automates what the human does naturally in Model 3: watching the pipeline behavior, noticing anomalies, correlating patterns across transactions, and making judgment calls about whether the process is functioning as expected.

4. What Overwatch Is

Overwatch is a cognitive monitoring shield for GOPEL infrastructure. It is deliberately cognitive, operating in its own trust boundary, separate from GOPEL's non-cognitive enforcement layer. It observes, classifies, and escalates. It does not enforce, block, or modify. Its entire job is to watch.

4.1 The Shield Metaphor

A shield protects what is behind it without changing what is behind it. Overwatch protects GOPEL's integrity by catching threats before they reach the enforcement layer, verifying that the enforcement layer itself has not been compromised, and alerting human operators when cognitive judgment is needed. Think of GOPEL's cryptographic enforcement as armor. The shield does not replace the armor. It adds active threat detection that armor, by design, cannot provide.

4.2 Architectural Position

Overwatch sits beside GOPEL, not between GOPEL and the AI pipeline. It receives copies of GOPEL's audit records through a read-only observer interface and analyzes them independently. That position, external to the enforcement path, is a deliberate design choice. If Overwatch fails completely, GOPEL keeps enforcing. The pipeline keeps processing transactions, the hash chain keeps recording them, and the evidence gates keep requiring human approval. What you lose is monitoring, not enforcement. That asymmetry is the whole point.

4.3 The Isolation Invariant

Overwatch and GOPEL are separate programs. Overwatch never imports GOPEL modules, never calls GOPEL functions, never touches GOPEL internal state. I verify this with static analysis: zero GOPEL imports across all 16 Overwatch modules. If someone compromises Overwatch, they have not compromised GOPEL. The blast radius shrinks from total governance failure to partial monitoring loss.

5. The Sixteen Modules

Overwatch is organized into 16 independent modules across five functional groups: threat detection (Intent Analyzer, Context Inspector, Output State Evaluator), infrastructure verification (Structural Verifier, GOPEL Observer), operational management (Escalation Engine, Random Audit Generator, Channel Manager), adaptation (Factics Engine, CAIPR Dispatcher), and integration (Execution Graph Engine, Provenance Manager, Crypto Module, Pipeline, Models, Structured Logger).

5.1 Threat Detection

The Intent Analyzer checks whether prompt sequences line up with the declared RECCLIN role or are quietly building toward something undeclared. It keeps rolling intent profiles in bounded windows and matches against a library of known attack chain signatures. In v2.3, the Intent Analyzer gained the trajectory gatekeeper capability described in Section 6.

The Context Inspector scans input payloads for embedded directives, verifies provenance tags, enforces trust-tier decay rules, and catches content trying to override higher-tier instructions. It automatically decodes base64 and hex-encoded content, then rescans for injection payloads that were hiding behind the encoding.

The Output State Evaluator looks at whether platform responses imply state changes that go beyond the declared task scope. It catches confused deputy conditions, unauthorized state changes, and violations of the role-behavior envelope. String normalization preprocessing collapses obfuscation tricks before the scan even starts.

5.2 Infrastructure Verification

The Structural Verifier checks GOPEL's infrastructure integrity from outside GOPEL's trust boundary. It computes SHA-256 hashes of every source file and compares them against signed deployment manifests. It checks configuration integrity and behavioral baselines. Randomized jitter on check intervals prevents anyone from timing their way around the verification schedule.

The GOPEL Observer provides the read-only integration layer: frozen dataclass interface, SHA-256 chain validation, thread-safe concurrent submission, and bounded stale record flushing.

5.3 Operational Management

The Escalation Engine runs the dual-mode operating model. In RAI mode (Responsible AI), everything moves at machine speed. In AIG mode (AI Governance), human checkpoints activate. The engine manages five severity levels from NOMINAL through HALT. In v2.3, it gained trajectory-aware evaluation (Section 6).

The Random Audit Generator samples transactions using cryptographically random selection. The sampling probability ratchets up automatically when findings appear and only ratchets down by human decision. Reports are hash-chained and formatted for SIEM integration.

The Channel Manager handles signed message transport with a dead-man's switch: if the channel goes silent when it should not be silent, that silence itself becomes an alert. It also tracks delivery failures within bounds to prevent alert floods.

5.4 Adaptation

The Factics Engine runs the Facts + Tactics + KPIs adaptation cycle. Every confirmed attack becomes a detection rule. Every confirmed false positive refines a threshold. It persists everything in JSONL logs and requires human approval before acting on critical findings.

The CAIPR Dispatcher (Consensus Across Independent Platforms Review) coordinates cross-platform consensus with asymmetric security weighting. The rule is simple: any single CRITICAL flag from any platform overrides a majority consensus of lower severity. The system deliberately biases toward security, accepting some false positive noise as the cost of not missing real threats.

5.5 Integration

The remaining six modules handle plumbing. The Execution Graph Engine reconstructs audit trails as directed acyclic graphs. The Provenance Manager tracks HMAC-signed provenance chains. The Crypto Module centralizes signing with key rotation support. The Pipeline orchestrates module execution. Models defines shared data types with __slots__ optimization for memory efficiency. The Structured Logger writes JSON Lines with log injection prevention built in.

6. The Trajectory Gatekeeper: Catching Semantic Manipulation

Overwatch v2.3 introduced the Trajectory Gatekeeper to address a gap that neither GOPEL's deterministic enforcement nor Overwatch's pattern matching could reach: semantic manipulation.

6.1 The Semantic Manipulation Gap

Semantic manipulation is designed to not look like an attack. Every prompt is syntactically clean. Nothing triggers an injection signature. Everything falls within behavioral envelopes and satisfies all formal policy requirements. But across a sequence of turns, the conversation is quietly steering the model toward an outcome the attacker wants, through meaning, not structure. GOPEL cannot catch it because GOPEL does not process meaning. Overwatch's existing pattern detection (regex signatures, statistical baselines, chain signature matching) cannot catch it because the patterns are clean. Each individual transaction passes every check. The manipulation lives in the trajectory, not in any single transaction.

6.2 How the Gatekeeper Works

The Trajectory Gatekeeper does not try to understand what words mean. It watches three soft signals across the conversation window and fires a CBG (Checkpoint-Based Governance) trigger when those signals start converging.

Signal 1: Scope Drift. The gatekeeper compares how many scope indicators appear in the first half versus the second half of the conversation window. If the second half shows significantly more reconnaissance patterns, system probing, or information gathering language, scope is

drifting. No single turn triggers a finding, but the aggregate direction is moving toward system internals.

Signal 2: Privilege Gradient. The gatekeeper tracks the rate of increase in privilege indicators (escalation attempts, access probing, security bypass language) across the window. A positive gradient means the conversation is progressively testing higher authority operations, even though no single turn crosses the escalation threshold.

Signal 3: Coherence Decay. The gatekeeper measures how fast role coherence scores are declining across the window. If someone declared as a Researcher starts producing content that increasingly matches a Coder or Administrator profile, coherence is decaying. That might be legitimate scope evolution. It might be semantic manipulation gradually pushing past the declared role boundary.

6.3 Convergence and the CBG Trigger

No single signal triggers escalation on its own. Scope drift alone could be legitimate exploration. Privilege gradient alone could be a researcher investigating access controls. Coherence decay alone could be a task that genuinely spans multiple role domains. But when two or more signals trend in the same direction across the conversation window (configurable, default 2 of 3), the convergence says: a human should look at this.

When the Escalation Engine detects a TrajectoryAlert, it escalates to AIG mode regardless of individual finding severity. The escalation gets tagged as TRAJECTORY (distinct from severity-based escalation) and the full alert details go into the escalation history for the human reviewer.

The human sees the conversation history, the three signal values, the convergence count, and the evidence chain. They decide whether this trajectory represents real manipulation or legitimate scope evolution. That decision feeds back through the Factics loop: confirmed manipulation tightens trajectory thresholds, confirmed false positives loosen them. The gatekeeper gets sharper through operational experience.

6.4 Configuration

Parameter	Default	GOPEL Mode
trajectory_window_min	4 turns	4 turns
trajectory_scope_drift_threshold	0.3	0.2 (tighter)
trajectory_coherence_decay_threshold	0.15	0.10 (tighter)
trajectory_privilege_gradient_threshold	0.1	0.05 (tighter)
trajectory_convergence_required	2 of 3 signals	2 of 3 signals

When Overwatch runs in GOPEL mode (monitoring a production GOPEL instance), all trajectory thresholds tighten automatically. The principle is simple: in production, route to human earlier rather than later.

6.5 What the Gatekeeper Does Not Do

The gatekeeper does not understand semantic meaning. It does not run embedding models. It does not inject queries into the pipeline. It does not make governance decisions. It watches trajectory trends using the same signals the Intent Analyzer already computes (scope indicators,

privilege indicators, role coherence scores) and fires a CBG trigger when those trends converge. The actual semantic manipulation judgment stays with the humans. The gatekeeper's job is making sure humans see the pattern in time to act on it.

6.6 Factics Calibration: Closing the Feedback Loop (v2.4)

Kimi's stress-test review of v2.3 validated the trajectory architecture but raised a critical operational question: how do these thresholds converge to useful sensitivity without oscillating into uselessness? The v2.3 paper acknowledged that confirmed attacks tighten thresholds and confirmed false positives loosen them, but left four engineering details unspecified. Overwatch v2.4 answers all four in code.

Asymmetric Adjustment. A confirmed threat tightens the relevant threshold by 10%. A confirmed false positive loosens it by 5%. The asymmetry is deliberate: missing a real attack is more expensive than one extra human review. If the scope drift threshold is 0.30 and a confirmed attack involved scope drift, the new target becomes 0.27. If it was a false positive, 0.315. The system always errs toward caution.

Per-Signal Independence. Each of the three signals calibrates independently based on its own incident history. If scope drift keeps catching real attacks but privilege gradient keeps producing false positives, scope drift tightens while privilege gradient loosens. Global adjustment would punish accurate signals for the noise of inaccurate ones.

Oscillation Prevention. Two mechanisms prevent threshold oscillation. First, a minimum incident count (default 3) must accumulate before any threshold moves. One incident is noise; three is a pattern. Second, exponential smoothing ($\alpha = 0.3$) dampens each adjustment so that the effective change blends 30% of the proposed new value with 70% of the current value. A single batch of false positives cannot drastically loosen a threshold that weeks of confirmed attacks had tightened.

Hard Bounds. Every threshold has a floor and a ceiling. The floor is the GOPEL mode value (the tightest the system should ever run): scope drift 0.15, coherence decay 0.05, privilege gradient 0.03. The ceiling is twice the default (the loosest it should ever get): scope drift 0.60, coherence decay 0.30, privilege gradient 0.20. The system cannot calibrate itself blind in either direction.

Tamper Evidence. The CalibrationState is HMAC-SHA256 signed on every write and verified on every read. If an attacker compromises Overwatch and tries to loosen thresholds directly (pushing them to the ceiling to blind the gatekeeper), the integrity check fails and the system falls back to configuration defaults. The calibration mechanism cannot become an attack surface because tampering is detected before tampered values are used. This addresses the critical requirement Kimi's review identified: without tamper evidence, the calibration loop itself becomes a vulnerability.

The calibration loop plugs into the existing Factics Engine. When a human confirms or clears a trajectory-triggered escalation, the Factics Engine records per-signal incidents from the TrajectoryAlert, applies the smoothed asymmetric adjustment when the minimum incident count is reached, clamps to bounds, re-signs the state, and pushes the calibrated thresholds to the Intent Analyzer before the next evaluation. Every adjustment is recorded in a capped audit history for governance review.

7. How Overwatch Protects GOPEL

7.1 Prompt Injection Defense

The Intent Analyzer and Context Inspector scan incoming prompts for injection patterns, delimiter attacks, role manipulation, and obfuscated payloads before they reach the model. GOPEL itself is immune because it contains no cognition. But the AI platforms GOPEL governs are not immune, and Overwatch provides the cognitive defense layer that GOPEL, by design, cannot.

7.2 Supply Chain Protection

The Structural Verifier computes SHA-256 hashes of every GOPEL source file and compares them against signed deployment manifests. Verification runs from outside GOPEL's trust boundary, so a compromised GOPEL cannot hide its own compromise.

7.3 Behavioral Drift Detection

The Output State Evaluator catches behavioral drift by comparing model responses against established baselines and role-behavior envelopes. The Escalation Engine uses advisory accumulation to make sure warning states do not get cleared prematurely.

7.4 Semantic Manipulation (v2.3/v2.4)

The Trajectory Gatekeeper catches conversations that are individually clean but collectively drifting toward manipulation. It routes to human review before the damage compounds. The v2.4 Factics Calibration closes the feedback loop: human decisions on trajectory escalations automatically adjust per-signal thresholds through tamper-evident, asymmetrically smoothed calibration with hard floor and ceiling bounds. The gatekeeper gets more accurate with every encounter it processes.

7.5 Operational Resilience

Random audits create deterrence because operators never know which transaction will get a deep look. The Factics Engine turns every confirmed attack into a new detection rule. The Channel Manager's dead-man's switch means that silence itself becomes an alert.

8. Relationship to GOPEL

Property	GOPEL	Overwatch
Nature	Non-cognitive. Deterministic.	Cognitive. Pattern analysis and trajectory monitoring.
Function	Enforces governance rules.	Monitors for threats rules cannot anticipate.
Action authority	Enforce, hash, sign, gate.	Stop pipe, route to human. Nothing else.
Compromise impact	Enforcement stops.	Detection stops. GOPEL continues.
Cross-imports	Zero.	Zero. Verified by static analysis.

The relationship is asymmetric by design. GOPEL does not depend on Overwatch. Overwatch reads GOPEL records through a read-only observer interface but cannot modify them. Neither system can suppress the other's operations. Zero cross-imports across 27 total source modules.

9. Human Checkpoints and Overwatch

Overwatch does not replace human judgment. It amplifies human attention. When Overwatch escalates from RAI to AIG mode (whether by severity threshold or trajectory convergence), it activates GOPEL's evidence gates where human judgment is mandatory. The human receives the full analysis: which findings triggered the escalation, what trajectory signals converged, what behavioral patterns were detected. The human then exercises the judgment that neither GOPEL nor Overwatch can replicate.

This creates a three-layer protection model. GOPEL enforces deterministic rules. Overwatch detects cognitive threats and trajectory anomalies. Humans arbitrate at critical decision points. Each layer covers threats the other two cannot. That is the design: the machine does what machines do well (compute, verify, monitor), and the human does what humans do well (judge, contextualize, decide).

10. The Review Cycle That Built Overwatch

Version	Focus	Tests
v1.0	Core: Intent Analyzer, Context Inspector, Output State Evaluator, Structural Verifier, Escalation Engine, Random Audit, Factics, Pipeline	59
v1.1.0	Cross-platform rebuild: 5 new modules. ChatGPT 15 fixes, MiniMax 10 fixes.	113
v1.1.1	Grok red-team: flush_stale fix, crypto.py, GOPEL mode, resource guards	19
v1.2	GOPEL hardening: Factics JSONL persistence, 10K txn performance	8
v1.3	Obfuscation: base64/hex decode-rescan, cross-operator correlation	9
v1.4–1.9	Grok spec: supply chain, trust boundary, chaos, drift, governance, scale	44
v2.0	Consolidated hardening (Gemini+Kimi+Claude): TIER_UNTRUSTED, thread safety, bounds	38
v2.1	Structured logging: centralized JSON Lines, sanitization consolidation	39
v2.2	__slots__ optimization, type hints, latent bug fix	0
v2.3	Trajectory Gatekeeper: semantic manipulation detection via soft-signal convergence	17
v2.4	Factics Calibration: tamper-evident CalibrationState, asymmetric smoothed threshold adjustment	44

Total at v2.4: 390 automated tests across 15 test suites, 100% passing. Seven independent AI platforms contributed findings.

11. Adaptability

11.1 The Factics Loop

Every confirmed attack produces a Fact (structural signature of what happened), a Tactic (detection rule to catch it next time), and a KPI (measurable improvement metric). Every confirmed false positive produces a pattern correction. The loop runs continuously with persistent JSONL logs that survive pipeline restarts.

11.2 The Trajectory Feedback Loop

When a human confirms semantic manipulation at a trajectory-triggered CBG checkpoint, the CalibrationState tightens the relevant signal thresholds by 10%. When a human clears a trajectory false positive, it loosens them by 5%. The asymmetric rate keeps the system biased toward safety. Exponential smoothing ($\alpha = 0.3$) prevents abrupt swings, a minimum incident count of 3 prevents premature adjustment, and hard floor and ceiling bounds keep calibration from pushing the gatekeeper into either hypersensitivity or blindness. The entire CalibrationState is HMAC-signed on every write and verified on every read. If tampering is detected, the system falls back to configuration defaults. See Section 6.6 for the complete specification.

11.3 Chain Signature Library

The Intent Analyzer's chain signature library grows with every confirmed attack. Detection capability increases over time and does not decrease, because confirmed attack signatures are never removed from the library.

12. Proof of Concept Status

This is a working proof of concept. It is not a finished product.

Metric	Value
Overwatch version	v2.4 (Factics calibration)
Source modules	16
Test count	390 across 15 test suites
Test pass rate	100%
Cross-imports (Overwatch → GOPEL)	0
Review platforms	7 (ChatGPT, MiniMax, Grok, Gemini, Kimi, Claude, DeepSeek)
Review findings addressed	60+
Versioned releases	14 (v1.0 through v2.4)
GOPEL version (monitored)	v0.6.2 (11 security fixes, 218 tests)

The complete Overwatch source code is publicly available for review at:
https://github.com/basilpuglisi/HAIA/tree/basilpuglisi-overwatch/overwatch

13. Limitations and Future Work

Scale testing with real enterprise workloads. Real-time ML classifiers for more sophisticated injection detection. Distributed Overwatch across multiple GOPEL instances. Formal verification of isolation invariants. Deeper trajectory analysis, including cross-operator trajectory correlation for attacks that split across multiple operator sessions.

These are engineering effort applied to a validated foundation.

14. Conclusion

Deterministic enforcement and cognitive monitoring are complementary capabilities. GOPEL provides enforcement that cannot be manipulated. Overwatch provides detection that enforcement cannot. The Trajectory Gatekeeper extends that detection to semantic manipulation, the threat class that neither pattern matching nor deterministic rules can catch, by watching conversation trajectory and routing to human review when soft signals converge.

Overwatch has exactly one action: stop the pipe, route to human. That constraint is its security property. It cannot interfere with GOPEL other than pulling the brake when something looks wrong. The shield watches, the armor holds, and the human decides. Sixteen modules, 390 tests, zero cross-imports, fourteen versioned releases. The shield works. And v2.4 proved that the calibration loop converges rather than oscillates, which means the system gets more accurate the longer it runs.

15. References

[1] OWASP. (2026). OWASP Top 10 for Agentic Applications. Open Worldwide Application Security Project. https://owasp.org/www-project-top-10-for-large-language-model-applications/

[2] National Institute of Standards and Technology. (2026). Request for Information on AI Agent Security. Federal Register.

[3] National Institute of Standards and Technology. (2026). Composable Overlay for Secure AI Systems (COSAiS). NIST.

[4] MITRE. (2026). ATLAS: Adversarial Threat Landscape for AI Systems, Agentic Expansion. MITRE Corporation. https://atlas.mitre.org/

[5] MCPTox Benchmark. (2026). Tool-Description Poisoning Attack Success Rates Across LLM Agents. 72.8% success rate across 20 agents.

[6] Vassilev, A. (2026). Testimony on guardrail limitations and adversarial bypass gaps. National Institute of Standards and Technology.

[7] National Institute of Standards and Technology. (2023). NIST SP 800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations. https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

[8] European Parliament and Council. (2024). Regulation (EU) 2024/1689 (EU AI Act). Official Journal of the European Union.

[9] Puglisi, B. C. (2026). GOPEL Canonical Public Description v1.5. HAIA-RECCLIN Framework Series.

[10] Puglisi, B. C. (2026). HAIA-Overwatch Specification v1.0: Adaptive Security Shield for the HAIA Ecosystem. HAIA-RECCLIN Framework Series.

[11] Puglisi, B. C. (2026). GOPEL v0.6.1 Security Review Report: Pre-Lockdown Hardening Assessment. HAIA-RECCLIN Framework Series.

[12] Puglisi, B. C. (2026). GOPEL Specification Gap Analysis: v0.6.1 Reference Implementation vs. Canonical v1.4 + Extensions. HAIA-RECCLIN Framework Series.

[13] Puglisi, B. C. (2026). The First AI-Orchestrated Cyberattack and the Road We Chose Not to See. Analysis of agentic-attack threat landscape and governance gap.

[14] Puglisi, B. C. (2026). Council for Humanity Proposal v1.4. HAIA-RECCLIN Framework Series.

[15] Puglisi, B. C. (2026). HAIA-RECCLIN Agent Architecture Specification (EU Compliance Version). HAIA-RECCLIN Framework Series.

[16] Puglisi, B. C. (2026). GOPEL Post-Quantum Cryptographic Agility Amendment v1.2. HAIA-RECCLIN Framework Series.

[17] Puglisi, B. C. (2026). HAIA-Overwatch Code Review and Analysis Report. MiniMax Agent, May 2026.

[18] National Institute of Standards and Technology. (2023). AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1. https://www.nist.gov/artificial-intelligence/ai-risk-management-framework

[19] Meta AI. (2025, April 29). LlamaFirewall: An Open Source Guardrail System. https://ai.meta.com/research/publications/llamafirewall/

[20] Microsoft Open Source. (2026, April 2). Introducing the Agent Governance Toolkit. https://opensource.microsoft.com/blog/2026/04/02/introducing-the-agent-governance-toolkit/

[21] Bessemer Venture Partners. (2026, March). Securing AI Agents: The Defining Cybersecurity Challenge of 2026.

[22] Foundation for Defense of Democracies. (2026, March 9). Regarding Security Considerations for Artificial Intelligence Agents. https://fdd.org/analysis/2026/03/09/regarding-security-considerations-for-artificial-intelligence-agents/

[23] Proofpoint. (2026, March 17). Proofpoint Unveils Industry's Newest Intent-Based AI Security Solution. https://proofpoint.com/us/newsroom/press-releases/

[24] Zenity/MITRE ATLAS. (2026, January). AI Security and Agentic Threats 2026 Update. https://zenity.io/blog/current-events/mitre-atlas-ai-security

[25] Coalfire. (2026, January). AI Agent Security in 2026: A Practitioner Perspective.

[26] Puglisi, B. C. (2026). The First AI-Orchestrated Cyberattack and the Road We Chose Not to See. Analysis of GTG-1002 and agentic-attack threat landscape.