TALONYX

ORILink

Bidirectional Pre/Post-Inference Enforcement for AI Agents

Architectural Whitepaper
June 2026 | Version 1.4

100%	0%	800+	~1ms
Block rate	False positives	Total test cases	Avg gate latency

Validated across 6 LLM architectures including open-source, commercial, and hardened models | talonyx.ai

Abstract

ORILink is a bidirectional enforcement architecture for AI agents. It operates below the model layer — before inference on the inbound side, before execution on the outbound side — to intercept and block prompt injection attacks and unauthorized agent actions without requiring changes to the underlying model.

This whitepaper describes the threat model ORILink addresses, the multi-point enforcement architecture, the agent-to-agent (A2A) provenance system, observed validation results, and the deployment model. It is written for developers and security engineers evaluating agent infrastructure security.

ORILink does not patch the model. It enforces at the infrastructure layer — making security unconditional, model-agnostic, and independent of prompt engineering or fine-tuning.

1. The Problem

1.1 Prompt Injection Is Unsolvable at the Inference Layer

Prompt injection — the injection of malicious instructions into an agent's input context — is the fundamental attack surface of deployed AI agents. An agent that reads email, browses the web, processes documents, or receives messages from other agents is continuously exposed to content that may contain adversarial instructions disguised as legitimate input.

OpenAI, Google DeepMind, and Anthropic have each independently acknowledged that prompt injection cannot be reliably prevented at the inference layer. The model processes language; it cannot natively distinguish a trusted instruction from a malicious one embedded in external content. This is not a model quality issue — it is a structural limitation of how large language models work.

> *The model cannot solve this problem. Security must be enforced at the infrastructure layer, before the model sees the input.*

1.2 Agents Are Compliant by Design

AI agents are built to execute instructions. That compliance is a feature — it is what makes them useful. But it also makes them exploitable. An agent that can be instructed to read files, call APIs, send messages, or interact with other agents can be instructed to do those things maliciously, if the instruction source is compromised.

The threat has two directions:

- Inbound: malicious content in the agent's input context attempts to hijack the agent's behavior.
- Outbound: a compromised operator, a poisoned configuration, or a weaponized instruction source directs the agent to take harmful action against external systems or people.

Existing security tools address neither direction at the agent layer. Perimeter tools (WAFs, network firewalls) operate at the transport level and have no visibility into agent reasoning or intent. Identity tools (Okta, OAuth) verify who the agent is — not what it is about to do.

1.3 The Multi-Agent Contagion Problem

Modern agent deployments increasingly involve multiple agents communicating with each other — coordinating tasks, sharing context, delegating subtasks. This introduces a new attack surface: agent-to-agent (A2A) contagion.

A compromised agent does not stay compromised in isolation. It acts as a payload carrier. If Agent B receives malicious content from the web and forwards it to Agent A through a trusted channel, Agent A has no way to know the content originated outside the trust boundary. Standard A2A communication protocols carry no provenance information — the receiving agent sees a message from a trusted peer, not a message from the external web.

> *Trust cannot be conferred by forwarding. Content that entered the system as untrusted remains untrusted regardless of which agent forwarded it.*

2. The ORILink Architecture

ORILink is deployed as standalone middleware between the agent framework and the model. It requires no changes to the model, no changes to the agent's application logic, and no integration with the model provider. It operates unconditionally — no configuration can disable enforcement in ORILink Standard.

ORILink enforces at four points in the agent's request lifecycle, each addressing a distinct attack surface:

- Before your agent reads anything — content is checked for origin and trust level [Gate 1]
- Before your agent reads anything — data structure and encoding is validated [Gate 1.5]
- Before your agent acts on anything — outbound intent is evaluated against prohibited categories [Gate 2]
- Before your agent sends anything — output is scanned for sensitive data leakage [Gate 2.5]

2.1 Inbound Content Filtering

[Gate 1 — Pre-Inference]

Before your agent reads anything, ORILink checks where it came from. Every input token is annotated with a trust level based on its origin. Instructions from your own system carry full trust. Content from external websites, documents, tool responses, or other agents carries lower trust calibrated to its origin.

Anything carrying an injection signature — an attempt to override the agent's instructions, escalate privileges, or redirect behavior — is blocked before the model sees it.

Characteristic	Description
Pre-inference	The model never receives blocked content
Origin-aware	Trust level is determined by where content came from, not what it says
Token-level	Granular classification, not binary allow/block
Unconditional	Cannot be disabled by operator configuration

2.2 Structured Input Validation

[Gate 1.5 — Pre-Inference]

Before your agent reads anything, ORILink also validates the structure and encoding of incoming data. This enforcement point addresses attack surfaces that origin-checking alone does not close: injection payloads hidden inside structured data fields, and encoding-based evasion techniques designed to disguise malicious instructions.

Encoding and obfuscation attacks caught include:

- Base64, URL, and Hex-encoded payloads — decoded and rescanned before reaching the model
- Unicode homograph substitution — visually similar characters used to evade pattern detection
- Zero-width character insertion — invisible characters used to break up recognized attack strings
- Comment injection and schema mutation — malicious instructions embedded in structured data fields

Characteristic	Description
Multi-stage pipeline	Schema validation, signature detection, encoding detection, and obfuscation normalization operate in sequence
Encoding-aware	Encoded payloads are decoded and rescanned — encoding does not hide origin or intent
Normalization	Obfuscation techniques are normalized to canonical form before pattern matching
Unconditional	Cannot be disabled by operator configuration

2.3 Outbound Action Enforcement

[Gate 2 — Pre-Execution]

Before your agent acts on anything, ORILink evaluates what it is actually about to do. Not what the instruction calls it — what it would actually accomplish. The pending action is resolved into its concrete execution steps and evaluated against prohibited intent categories.

The critical design principle: framing attacks are caught because what the action does is evaluated independently of what it is called. An agent instructed to "research competitor infrastructure" that would result in network scanning is blocked — not because of the word "research," but because of what the resulting action sequence would actually do.

Characteristic	Description
Post-reasoning, pre-execution	The agent reasons normally, but cannot act on prohibited intent
Intent-based classification	Action label and framing are irrelevant — execution steps are the unit of analysis
Agent continuity	A blocked action does not crash or freeze the agent — it receives a structured refusal and continues operating
Full audit trail	Every block logged with timestamp, intent category, and the instruction chain that produced it
Operator notification	Blocks reported to the configured operator channel in real time

2.4 Output Leakage Detection

[Gate 2.5 — Post-Generation]

Before your agent sends anything, ORILink scans the output for sensitive data that should not leave the system. This enforcement point catches accidental or malicious exfiltration of credentials, keys, and private configuration in agent outputs before they reach any recipient.

Output leakage detection covers:

- API keys and tokens — OpenAI, Anthropic, AWS, GitHub, Google, and generic high-entropy patterns
- Credentials and passwords
- Private keys and certificates
- System prompt reconstruction and leakage
- Sensitive configuration data

2.5 Deployment Architecture

ORILink is deployed as a middleware layer between the agent framework and the model endpoint. The request lifecycle is:

- Agent framework sends request to ORILink
- Inbound content filtering [Gate 1] classifies tokens by origin and trust level
- Structured input validation [Gate 1.5] validates encoding and data structure
- Clean input passes to the model
- Outbound action enforcement [Gate 2] evaluates the model's intended action
- Output leakage detection [Gate 2.5] scans the output before delivery
- Action executes or is blocked permanently

> *Agent framework → ORILink (inbound filtering + structured validation + outbound enforcement + output leakage detection) → Model. The model sees only trusted, validated input. Actions execute only after outbound intent clearance. Output is scanned before delivery.*

ORILink is framework-agnostic. It does not require integration with any specific agent framework, model provider, or orchestration layer. Deployment is a configuration change, not a code change for the agent application.

---

3. A2A Provenance Enforcement

ORILink tracks content provenance end-to-end through agent-to-agent communication chains. Every piece of content that enters the system through inbound content filtering [Gate 1] is annotated with its origin and trust level. That annotation travels with the content — it cannot be elevated by a forwarding agent.

When Agent A forwards content to Agent B, Agent B's inbound filtering receives not just the content but its provenance record — where the content originally entered the system and what trust level it was assigned at

entry. Agent B's enforcement decisions reflect the content's actual origin, not Agent A's identity.

Messages that arrive without a valid provenance record are rejected. An agent cannot claim that content is trusted by forwarding it without provenance metadata. This closes the contagion attack surface: a compromised agent forwarding malicious content to peers cannot elevate the trust level of that content by virtue of being a trusted sender.

> *ORILink tracks where content came from — not just who forwarded it. Trust cannot be laundered through agent-to-agent forwarding.*

4. Blocked Intent Categories

The following intent categories are permanently blocked in ORILink Standard. They cannot be disabled by operator configuration. They represent the enforcement floor — the guarantee that ORILink cannot be used to weaponize an agent against external systems or people.

Category	Description
Unauthorized reconnaissance	Scanning, probing, or mapping external systems not in the agent's authorized scope
Attack payload generation	Constructing prompt injection payloads, exploit scripts, or jailbreak sequences
Unauthorized data exfiltration	Bulk read of sensitive data followed by transmission to external endpoints
Agent impersonation	Presenting as a different agent, system, or authority to gain elevated trust
Lateral movement	Accessing systems or data stores outside the agent's authorized scope
Denial of service patterns	Request volumes or patterns designed to degrade external systems
A2A contagion propagation	Forwarding unverified instructions to peer agents through trusted channels
Social engineering	Deceptive content designed to manipulate a human against their own interests
Output leakage	Transmission of credentials, keys, or sensitive configuration in agent outputs

5. Validation Results

ORILink was validated across three independent test phases covering 800+ test cases and 6 distinct LLM architectures — open-source, commercial, and hardened models. No simulated or mocked inference was used.

Internal Validation	Live Ollama Endpoints	Combined Results
Llama 3 | Mistral 7B | Gemma 2 | GPT-4o | GPT-4o Mini | Claude Haiku **100% block rate | 0 false positives**	LLaMA 3 8B | Mistral 7B v0.3 | Gemma 3 4B (Apple Silicon) | 87 live inference calls **100% block rate | 0 false positives**	6 LLM architectures | 800+ test cases | Open-source, commercial, and hardened models **100% block rate | 0 false positives**

Test coverage included:

- Framing attacks — harmful actions described using legitimate-sounding instruction language
- Multi-step attack chains — prohibited actions spread across multiple sequential steps
- A2A contagion vectors — malicious instructions forwarded through agent-to-agent channels
- Encoding and obfuscation attacks — injections disguised through encoding transformations
- Knowledge base poisoning — false authority instructions embedded in RAG sources
- Supply chain injection — malicious instructions embedded in external library responses
- Technical obfuscation attacks — character substitution, Unicode homograph substitution, zero-width character insertion, and comment injection
- False positive validation — 7 categories of legitimate agent actions confirmed unblocked

Inbound filtering [Gate 1] latency: ~1ms average. Outbound enforcement [Gate 2] latency: sub-millisecond for scope and deception checks; up to 3ms for full intent classification. Both enforcement points operate within latency budgets compatible with real-time agent deployments.

6. Product Tiers

	ORILink Standard	ORILink Professional (Planned)
Inbound content filtering [Gate 1]	Full enforcement	Full enforcement
Structured input validation [Gate 1.5]	Full enforcement	Full enforcement
Outbound action enforcement [Gate 2]	Hard block, unconditional	Tiered: permanent floor + authorized override

Output leakage detection [Gate 2.5]	Full enforcement	Full enforcement
ORIGuard autonomous monitoring	Included — all Business and Enterprise tiers	Included
Override mechanism	None	Explicit operator authorization with full audit trail
Target deployment	Developers, small teams, general agent deployments	Penetration testing firms, enterprise red teams, authorized security researchers
Status	Available now — Individual and Business tiers	Design and legal review in progress — no timeline set

> *ORILink Professional is not available today. The authorized override framework requires careful legal and architectural design before any build begins. The hard enforcement floor of ORILink Standard applies unconditionally in both tiers — ORILink Professional extends above the floor, never below it.*

7. How ORILink Fits the Security Stack

ORILink is not a replacement for perimeter security or identity management. It operates at the agent layer, where those tools have no visibility.

Layer	Tool Category	What It Provides	What It Cannot Do
Network	Perimeter / WAF	Transport-level filtering, rate limiting	Understand agent intent or semantic content
Identity	Okta, OAuth	Verify who the agent is	Know what the agent is about to do
Agent	ORILink	Inbound content filtering, structured input validation, outbound action enforcement, output leakage detection, A2A provenance tracking	Replace network or identity controls

> *Identity tells you who. ORILink tells you what — and stops it if it shouldn't happen.*

8. Deployment Independence: Why the Harness Is Optional

In 2026, the frontier AI labs converged on a new product category: the agent harness. Anthropic launched Managed Agents. OpenAI shipped a model-native harness. Google packaged the layer into Vertex Agent Engine. Microsoft shipped Foundry Agent Service. Each provides session management, tool orchestration, sandboxed execution, error recovery, and observability.

These are useful workflow tools. They solve real operational problems for teams running long-horizon autonomous agents. But they do not solve the security problem that ORILink addresses.

8.1 The Harness Layer vs. the Enforcement Layer

A harness manages how an agent does its work across time and tools. It handles session state, tool routing, sandbox lifecycle, and error recovery. These are orchestration concerns.

ORILink manages what the agent is allowed to understand and do at the language level. Inbound content filtering [Gate 1] classifies tokens by origin before inference. Outbound action enforcement [Gate 2] evaluates intent against prohibited categories before execution. Output leakage detection [Gate 2.5] scans output before delivery. These are enforcement concerns.

> *The harness manages the workflow. ORILink protects what flows through the agent. These are architecturally distinct responsibilities operating at different layers of the stack.*

8.2 Framework-Agnostic, Harness-Optional

ORILink operates at the model communication layer, not the orchestration layer. Its architectural position is defined by the flow of data through the agent: the point where content enters the context window and the point where the agent's decisions become actions against external systems. These two points exist in every agent deployment, regardless of what infrastructure surrounds them.

Deployment Scenario	Harness / Framework	ORILink Behavior
Bare script calling a model API	None	Full enforcement at all four points. ~1ms latency.
Python agent with tool calls	None	Identical enforcement. Classification and intent blocking operate independently of application logic.
LangChain / CrewAI agent	Framework orchestration	ORILink wraps the model call. Framework handles workflow. Enforcement is unchanged.
OpenAI Agents SDK	Open-source harness	ORILink sits inside the model invocation. SDK handles sessions, tools, sandboxing. Enforcement is unchanged.

Anthropic Managed Agents	Hosted managed harness	ORILink protects the agent inside the managed environment. Enforcement is unchanged.
Google Vertex Agent Engine	Cloud-managed harness	ORILink operates at the model layer within the managed runtime. Enforcement is unchanged.
Custom enterprise platform	Internal harness	ORILink integrates at the model communication layer. No dependency on platform architecture. Enforcement is unchanged.

8.3 What This Means

ORILink is complementary to every harness provider rather than competitive with any of them. A team that deploys Anthropic's Managed Agents gains workflow orchestration from Anthropic and token-level enforcement from ORILink. A solo developer who needs no harness at all can protect their agent with ORILink directly. All receive identical enforcement.

> *An agent can run without a harness. It should not run without enforcement at the token level. The harness is optional infrastructure. ORILink is not.*

9. Conclusion

Prompt injection is a structural problem. It cannot be solved by asking models to be more careful, by refining system prompts, or by adding detection layers that operate after inference. Security for AI agents requires enforcement that operates before the model sees malicious input and before the agent acts on compromised instructions.

ORILink enforces at both boundaries — inbound and outbound — as unconditional, model-agnostic middleware. It has been validated against a broad attack surface across multiple open-source and commercial models, with 800+ test cases, zero false positives, and sub-millisecond enforcement latency.

ORILink Standard is available now. Individual SDK and Business SDK tiers are available at talonyx.ai. Developers building on agentic frameworks can integrate ORILink without changes to their model, their agent logic, or their existing security infrastructure.