Agentic AI for Autonomous Cyber Operations and Multi-Domain Intelligence Fusion

Zach Kelling — Hanzo Team

Version 1.0 — February 21, 2026 February 2026

Abstract

We present an agentic AI framework comprising 83 specialized agents and 15 multi-agent orchestrators for autonomous cyber operations, intelligence fusion, and tactical decision support. The framework operates across three tiers: (1) a Playground control plane providing Kubernetes-like orchestration for autonomous AI agents with W3C DID identity, verifiable credentials, and durable state; (2) an agent SDK with async-first execution, OpenAI-compatible APIs, and native Model Context Protocol (MCP) integration exposing 13 canonical tools; and (3) a ZAP-based consensus protocol enabling distributed agent voting with threshold agreement and post-quantum authentication. Agents execute 200–300 sequential tool calls for deep analysis, coordinate via zero-copy binary RPC at 4.2M calls/second, and operate fully disconnected on edge hardware using quantized Zen models (0.8B–27B parameters). The system supports automated red/blue team operations, vulnerability chain analysis, multi-source intelligence correlation, and course-of-action generation with explainable reasoning via chain-of-thought traces. A multi-channel bot platform unifies 20+ messaging channels under local-first privacy controls for secure operator interaction.

Executive Summary. This framework lets a team field a fleet of AI agents that investigate, defend, and analyze on their own—running on a laptop or a rugged edge device with no connection to the cloud. It is built for cyber-operations teams and intelligence analysts who must act at machine speed and cannot afford to ship sensitive data to an outside service. Every agent action is cryptographically signed, so there is a complete, tamper-proof record of what each agent did and why; and high-stakes calls require several independent agents to agree before anything is recommended, which guards against a single model's mistake. Operationally, this means continuous vulnerability hunting, automated red/blue exercises, and fused multi-source intelligence with a human always in the loop for final decisions—all owned and run by the operator, not rented from a remote provider. The result is autonomous capability without surrendering control, accountability, or the data itself.

Contents

1 Introduction

Modern naval operations generate intelligence from sensors, communications, cyber systems, and open sources at volumes exceeding human analytical capacity. Simultaneously, the cyber domain requires continuous vulnerability assessment, threat hunting, and incident response at machine speed. Traditional automation—rule-based scripts and playbooks—cannot adapt to novel threats or fuse information across domains.

Agentic AI addresses these challenges through autonomous agents that reason, plan, use tools, and collaborate. Unlike chatbots that respond to queries, agents take initiative: they decompose complex tasks, execute multi-step plans, invoke external tools, and coordinate with other agents to produce actionable intelligence.

1.1 Design Principles

1. **Autonomous but accountable**: Every agent action is signed with a W3C DID and produces a verifiable credential. Full audit trail, zero deniability.

2. **Disconnected-capable**: Agents run entirely on edge hardware with local models. No cloud dependency.

3. **Consensus-validated**: High-consequence decisions require threshold agreement among multiple agents, preventing single-agent hallucination.

4. **Tool-native**: Agents interact with systems through 13 canonical MCP tools, not bespoke integrations.

1.2 Strategic Context

This agent framework is one layer of an integrated stack built by Hanzo, an American applied-AI lab founded in 2014 (Techstars '17) whose founder works at the frontier of both AI/ML and cryptography. American frontier AI has consolidated into a closed monoculture—effectively two labs—while the only openly available frontier model weights today are largely foreign. Hanzo is the American open-source frontier alternative: it owns both the model (the Zen family, 0.8B–1T+ parameters, with owned weights) and the cryptography (a production post-quantum stack), so defense and regulated users get sovereign, open, auditable AI without renting from the closed duopoly or depending on foreign open weights. For autonomous cyber and intelligence work that requirement is acute—the agents here reason over the most sensitive data an organization holds, so the model performing that reasoning must be one the operator can own, inspect, and run disconnected, on their own hardware, where the data already lives. The same agentic stack is what makes this practical at scale: Hanzo used it to deliver a FINRA/SEC-approved, SOC2-compliant securities platform to production—a full ATS/BD/TA stack consolidated from 200+ microservices into 3 compiled binaries, in record time, with a small team—demonstrating that agent-driven engineering meets the bar of a heavily regulated, audited domain.

2 Agent Architecture

2.1 Specialization Taxonomy

Domain	Count	Capabilities
Security	12	Vulnerability assessment, pen testing, incident response, malware analysis, SIEM correlation
Infrastructure	11	K8s operations, network diagnostics, database administration, cloud provisioning
Data/ML	10	Data engineering, ML pipelines, model evaluation, feature engineering, dataset curation
Languages	15	Rust, Go, Python, TypeScript, C++, Java, Solidity, and 8 more
Architecture	8	System design, API design, microservices, distributed systems
QA/Testing	7	Unit testing, integration testing, load testing, security testing
Domain	20	Blockchain, DeFi, cryptography, NLP, computer vision, signal processing

Table 1: 83 specialized agents by domain.

2.2 Multi-Agent Orchestrators

Orchestrator	Workflow
Security Hardening	Red agent attacks → blue agent defends → red re-tests fixes
Incident Response	Triage → containment → eradication → recovery → lessons
ML Pipeline	Data prep → training → evaluation → deployment → monitoring
Full-Stack Dev	Architecture → backend → frontend → testing → deployment
Intelligence Fusion	Collection → processing → analysis → dissemination

Table 2: Selected multi-agent orchestrators (5 of 15).

2.3 Agent Execution Model

Algorithm 1 Agent Task Execution

Input: Task description T, available tools $\mathcal{F}$, context C
Initialize: Agent identity (DID), model (Zen), memory store
repeat
    Reason about current state and remaining subtasks
    Select tool $f \in \mathcal{F}$ and generate arguments
    Execute tool: $r \leftarrow f(\text{args})$
    Sign action: $\sigma \leftarrow \text{ML-DSA-65.Sign}(\text{sk}, T \| f \| r)$
    Update memory with observation r
    Append to audit trail: $(T, f, r, \sigma, \text{timestamp})$
until task complete or max steps reached (200–300 calls)
return (result, audit_trail, verifiable_credential)

3 Playground: Agent Control Plane

3.1 Architecture

Playground provides Kubernetes-like orchestration for autonomous AI agents:

- **Registration**: Agents register with capabilities, DID, and stake weight.
- **Discovery**: Other agents and services discover registered agents by capability.
- **Scheduling**: Tasks dispatched to agents based on capability match and availability.
- **Durable state**: Built-in persistent state and vector search (no external Redis or vector DB).
- **Lifecycle**: Health monitoring, graceful shutdown, automatic restart.

3.2 Identity and Accountability

Every agent in Playground possesses:

- **W3C DID**: Decentralized identifier derived from ML-DSA-65 public key (`did:key:z6Mk...`).
- **Verifiable Credentials**: Every action produces a DID-signed VC as a tamper-proof receipt.
- **Capability assertions**: Agent advertises what it can do; Playground verifies via credential chain.
- **Policy boundaries**: "Only agents with `security` credential can access vulnerability tools" — enforced at infrastructure level, not application level.

3.3 Performance

Runtime	Throughput	Overhead/Handler
Go	8.2M req/s	280 B
Python	6.7M req/s	512 B
TypeScript	4.0M req/s	384 B

Table 3: Playground control plane performance.

4 Model Context Protocol Integration

4.1 13 Canonical Tools

Agents interact with systems through the Model Context Protocol (MCP), providing a standardized tool interface:

Tool	Type	Capability
`fs`	Core	File system operations (read, write, search)
`exec`	Core	Shell command execution (sandboxed)
`code`	Core	AST analysis, symbol search, refactoring
`git`	Core	Version control operations
`fetch`	Core	HTTP requests, API calls
`workspace`	Core	Project context management
`ui`	Core	User interface interaction
`think`	Optional	Internal reasoning steps
`memory`	Optional	Persistent memory across sessions
`hanzo`	Optional	Platform API (IAM, KMS, PaaS)
`plan`	Optional	Task decomposition and planning
`tasks`	Optional	Task tracking and progress
`mode`	Optional	Execution mode control

Table 4: MCP canonical tool set (HIP-0300).

4.2 MCP-ZAP Bridge

The ZAP bridge provides 10–30× performance improvement over native MCP JSON-RPC:

- **Binary encoding**: Tool name (64 B fixed), arguments (length-prefixed JSON), result (zero-copy bytes).
- **Message types**: ToolList (100), ToolCall (101), ToolResult (102).
- **Auto-discovery**: Bridge queries MCP server capabilities and exposes via ZAP.

- **20-server aggregation**: Single ZAP endpoint fronts 20+ MCP servers with prefix namespacing.

4.3 Search Strategy

MCP implements multi-strategy code and data search:

Strategy	Priority	Engine	Triggers
Symbol	10	TreeSitter AST	`class`, `function`, `def`
AST	20	TreeSitter	Code-like patterns
Vector	30	LanceDB	Natural language queries
Text	40	Ripgrep	Default fallback
File	50	Glob	Paths, extensions

Table 5: MCP multi-strategy search with priority dispatch.

5 Distributed Agent Consensus

5.1 Threshold Voting Protocol

For high-consequence decisions, multiple agents must independently reach the same conclusion:

Algorithm 2 Agent Consensus for Threat Classification

Coordinator broadcasts sensor data D to n analyst agents
for each agent A_i in parallel **do**
    A_i independently analyzes D using assigned model
    A_i produces classification C_i with confidence p_i
    A_i signs: $\sigma_i \leftarrow \text{ML-DSA-65.Sign}(\text{sk}_i, D \| C_i \| p_i)$
    A_i returns $(C_i, p_i, \sigma_i, \text{DID}_i, \text{stake}_i)$
end for
Verify all signatures against registered DIDs
Compute weighted consensus: $\hat{C} = \arg\max_c \sum_{i:C_i=c} \text{stake}_i$
if $\sum_{i:C_i=\hat{C}} \text{stake}_i \geq \frac{2}{3} \sum_i \text{stake}_i$ **then**
    return (CONSENSUS, $\hat{C}$, aggregated evidence)
else
    return (NO_CONSENSUS, all (C_i, p_i), escalate to human)
end if

5.2 Defense Applications

- **Threat classification**: 3+ agents independently assess sensor contact; consensus prevents single-model hallucination from triggering false alarms.
- **Targeting**: Distributed analysis with $\frac{2}{3}$ threshold before engagement recommendation. Human remains in loop for final decision.

- **Intelligence corroboration**: Agents processing different SIGINT, IMINT, and OSINT sources; consensus identifies corroborated vs. single-source intelligence.
- **Cyber attribution**: Multiple agents analyze malware, network indicators, and TTPs independently; consensus reduces false attribution risk.

6 Autonomous Cyber Operations

6.1 Defensive Cyber Operations (DCO)

6.1.1 Continuous Vulnerability Assessment

1. Security agent scans codebase using MCP `code` tool (AST analysis, dependency audit).
2. Agent identifies vulnerability class (injection, XSS, auth bypass, crypto weakness).
3. Agent generates proof-of-concept exploit via `exec` tool (sandboxed).
4. Agent proposes fix, generates test case, and validates fix eliminates vulnerability.
5. Full workflow signed and recorded as verifiable credential.

6.1.2 Threat Hunting

1. O11y anomaly detection triggers agent investigation.
2. Agent queries distributed traces via MCP `fetch` tool.
3. Agent correlates indicators across logs, metrics, and traces.
4. Agent classifies threat using MITRE ATT&CK framework.
5. Multi-agent consensus validates classification before alert.

6.1.3 Incident Response Orchestrator

The 5-phase incident response orchestrator coordinates specialized agents:

Phase	Agent	Actions
Triage	Security analyst	Severity assessment, scope determination
Contain	Infrastructure	Network isolation, credential rotation
Eradicate	Malware analyst	Root cause removal, persistence check
Recover	DevOps	Service restoration, integrity verification
Lessons	Architecture	Post-incident report, control improvement

Table 6: Incident response orchestrator phases.

6.2 Offensive Cyber Operations (OCO)

6.2.1 Automated Penetration Testing

1. Reconnaissance agent maps attack surface via MCP `fetch` and `exec` tools.
2. Vulnerability agent identifies exploitable weaknesses in discovered services.
3. Exploitation agent chains vulnerabilities into attack paths.
4. Lateral movement agent assesses post-exploitation pivot opportunities.
5. Reporting agent produces findings with severity, evidence, and remediation.
6. **Operator identity**: Ring signatures (LatticeLSAG) protect pen tester identity during operations.

6.2.2 Red/Blue Team Automation

Adversarial pairs operate in continuous cycles:

Algorithm 3 Red/Blue Agent Cycle

repeat
  Red agent identifies attack vector
  Red agent develops and executes exploit (sandboxed)
  Blue agent receives attack indicators
  Blue agent develops detection rule and mitigation
  Red agent adapts attack to evade new detection
  Blue agent refines defense based on adapted attack
  Record cycle results as verifiable credentials
until convergence (red cannot bypass blue) or max iterations
return (attack report, defense posture, residual risk)

7 Intelligence Fusion

7.1 Multi-Source Correlation

Agents process intelligence from heterogeneous sources:

INT Type	Agent Capability	MCP Tools Used
SIGINT	NLP agent with 32+ language support	`code` (pattern matching), `fetch`
IMINT	Vision-language model (Zen VL)	`fs` (image access), `code` (analysis)
OSINT	Web search and document exploitation	`fetch` (HTTP), `fs` (documents)
CYBER	Network traffic analysis, malware RE	`exec` (tools), `code` (analysis)
HUMINT	Report processing and entity extraction	`code` (NER), `memory` (context)

Table 7: Intelligence source processing by agent type.

7.2 Graph-Based Fusion

The GraphVM provides in-memory graph analytics for intelligence correlation:

- **Entity resolution**: Link entities across SIGINT, IMINT, and OSINT sources.
- **Network analysis**: Identify communication patterns, social networks, and organizational hierarchies.
- **Temporal analysis**: Track entity behavior changes over time.
- **Geospatial correlation**: Associate entities with locations from multiple sources.
- **Merkle-verified**: Graph state is Merkle-proofed for integrity verification.

8 Multi-Channel Operator Interface

8.1 OpenClaw Bot Platform

OpenClaw provides a privacy-first, local-first multi-channel AI assistant:

- **20+ messaging channels**: WhatsApp, Telegram, Slack, Discord, Signal, Matrix, iMessage, and more.
- **Local-first**: All processing on operator's device. No cloud routing of messages.
- **Gateway control plane**: WebSocket-based session management, channel routing, tool dispatch.
- **Policy enforcement**: DM pairing, group routing rules, channel allowlists, mention gating.
- **Voice interface**: Wake word activation, Talk Mode (macOS/iOS/Android).
- **Skills platform**: Bundled, managed, and workspace-specific skill sets.

8.2 Defense Application

- Operators interact with AI agents via their preferred secure messaging app (Signal, Matrix).
- Agent responses routed through local gateway—never traverses external servers.
- Multi-channel unification: single AI assistant across all communication platforms.
- Group messaging with policy-enforced access controls per channel.

9 Explainable AI and Transparency

9.1 Chain-of-Thought as Explainability

Zen models with extended reasoning (96–128K thinking tokens) provide inherent explainability:

- **Reasoning traces**: Every conclusion preceded by visible reasoning steps.
- **Heavy Mode**: 8 parallel reasoning trajectories with explicit comparison.
- **Tool call logging**: Every external tool invocation recorded with inputs and outputs.
- **Memory traces**: Agent's retrieval of relevant context from vector memory is logged.

9.2 Verifiable Decision Records

- Every agent decision produces a verifiable credential (W3C VC).
- Credential contains: task, reasoning trace, tool calls, result, DID signature.
- Credentials form an immutable chain anchored to blockchain.
- Post-operation review can replay exact decision process.

9.3 Adaptability via LoRA

- Mission-specific LoRA adapters fine-tune agent behavior without full model retraining.
- Adapters swappable at runtime: switch from SIGINT to IMINT analysis in seconds.
- Quantized base model (4-bit) + unquantized adapter layers = edge-deployable specialization.

10 Reinforcement Learning from Operations

10.1 Operational Feedback Loop

Agent performance improves through operational feedback:

1. Agent executes task and produces result with confidence score.
2. Human operator validates or corrects result (accept/reject/modify).
3. Feedback recorded as preference pair: (agent output, human correction).
4. Preference pairs used for LoRA fine-tuning via RLHF/DPO.
5. Updated adapter deployed to operational agents.

10.2 Multi-Agent Learning

- **DeltaSoup**: Byzantine-robust aggregation of LoRA updates from multiple agents.
- **Contributor reputation**: Agents that produce higher-quality outputs gain reputation weight.
- **Differential privacy**: Individual training contributions protected from extraction.
- **GT-QLoRA**: Gate-Targeted LoRA for Mixture-of-Experts adaptation without full model access.

11 Edge Deployment

11.1 Disconnected Agent Operation

Platform	Model	Agents	Capability
MacBook M4	zen4-pro (27B)	5–10	Full reasoning, OCR
Jetson Orin	zen4 (9B)	3–5	Tactical analysis
Raspberry Pi 5	zen4-nano (0.8B)	1–2	Basic classification
Server (A100)	zen4-ultra (1T)	20+	Frontier reasoning

Table 8: Edge deployment configurations.

11.2 Agent Mesh over RNS

Agents on disconnected nodes coordinate via RNS mesh:

- Each node runs local agents with local models.
- Agent consensus messages traverse RNS mesh (LoRa, packet radio, serial).
- ZAP binary protocol minimizes bandwidth (256 B per call vs. 2,826 B JSON-RPC).
- Store-and-forward ensures eventual consensus delivery.
- Hybrid PQ handshake protects all agent communication.

12 AR/VR Integration

12.1 Vision-Language Models as AR Engine

Zen VL models provide real-time scene understanding for augmented reality overlays:

- **Object detection**: Identify and classify objects in camera feed.
- **Spatial reasoning**: Determine object positions, distances, and relationships.
- **OCR**: Extract text from signage, documents, and displays in 32 languages.
- **Temporal tracking**: Video comprehension for activity detection and tracking.

12.2 Mission Rehearsal in VR

- Digital twin environments (Netrunner) feed simulated scenarios to VR displays.
- Babylon.js 3D rendering engine provides immersive visualization.
- Agents provide real-time tactical analysis overlaid on VR environment.
- Network conditions (DDIL, jamming) simulated for realistic rehearsal.

13 Conclusion

We have presented a comprehensive agentic AI framework providing autonomous cyber operations, multi-domain intelligence fusion, and tactical decision support. The combination of 83 specialized agents, Playground control plane with verifiable credentials, distributed consensus with threshold voting, and multi-channel operator interface provides a complete system for naval information warfare operations.

The framework's key differentiators—post-quantum agent identity, consensus-validated decisions, edge-deployable inference, and explainable reasoning—address the specific requirements of naval operations where accountability, disconnected operation, and trust are paramount.

References

[1] Anthropic, "Model Context Protocol Specification," 2024.

[2] W3C, "Decentralized Identifiers (DIDs) v1.0," 2022.

[3] W3C, "Verifiable Credentials Data Model v2.0," 2024.

[4] NIST, "ML-DSA Standard," FIPS 204, 2024.

[5] MITRE, "ATT&CK Framework," 2024.

[6] E. Hu *et al.*, "LoRA: Low-Rank Adaptation," ICLR 2022.

[7] R. Rafailov *et al.*, "Direct Preference Optimization," NeurIPS 2023.

[8] L. Ouyang *et al.*, "Training Language Models to Follow Instructions with Human Feedback," NeurIPS 2022.

[9] T. Dao *et al.*, "FlashAttention-2," ICLR 2024.

[10] Hanzo Team, "ZAP: Zero-Copy Application Protocol," 2026.

[11] Hanzo Team, "Quasar Consensus," Hanzo, 2025.

[12] "Two-Round Threshold Signature from LWE," IACR ePrint 2024/1113.

[13] Zen LM, "Zen Model Family Technical Report," 2026.

[14] LanceDB, "LanceDB: Serverless Vector Database," 2024.

[15] M. Brunsfeld, "Tree-sitter: Incremental Parsing System," 2018.

[16] OpenTelemetry, "Observability Framework," CNCF, 2024.

Reproducibility

Source code. `github.com/hanzoai/agents` (commit `TBD`).
Build. `pnpm install && pnpm build` (Node 20+); SDK via `uv sync`.
Hardware tested. Apple M-series, x86_64 Linux + NVIDIA A100/H100.
Standards referenced. W3C DID, OpenAI tool-use API, MCP, OpenTelemetry.
Contact. research@hanzo.ai.