CONCEPT WHITE PAPER / GOVERNED AGENTIC SYSTEMS

SHADOWCORE.AI

A Sovereign Control Plane for Governed Agentic Systems

Andrey Zaykovskiy | ShadowCore AI | Vancouver, Canada

Concept White Paper | Version 0.9 | July 2026

Control before autonomy. Evidence before trust.

Status and intended use

This document defines a target architecture and product thesis for ShadowCore AI. It is a concept white paper, not a claim of production readiness, regulatory compliance, certification, patent issuance, or completed enterprise validation. Framework references describe design alignment and intended evidence outputs; an implementation would still require control testing, independent review, legal analysis, and deployment-specific threat modelling.

ShadowCore AI is intended to make tool-using AI systems easier to constrain, inspect, approve, verify, and stop. It does not attempt to make an unreliable model reliable by assertion. It assumes model outputs, retrieved content, tool descriptions, agent messages, and remembered context may all be wrong or adversarial.

Abstract

AI agents change the unit of risk. A conventional model produces content; an agent can select tools, call APIs, modify files, send messages, deploy code, move money, or coordinate other agents. The critical question is therefore no longer only whether a model generated a correct answer. It is whether a proposed action was authorized, bounded, observable, reversible, and supported by sufficient evidence.

ShadowCore is a sovereign control plane for agentic systems. It sits between intent and side effects. It converts a user goal into a signed intent capsule; evaluates the proposed plan against policy, identity, data, cost, and risk constraints; routes consequential actions through explicit approval; issues short-lived capabilities to isolated workers; verifies post-conditions; and records an evidence chain that can support incident response, audit, and governance.

The design is model-agnostic, tool-agnostic, and deployment-flexible. It can govern local models, hosted models, Model Context Protocol servers, direct APIs, code sandboxes, and multi-agent workflows. A local-first deployment can keep sensitive content inside the user’s environment. Enterprise and regulated deployments can centralize policy without centralizing raw data.

The proposition is deliberately narrow: autonomy should scale only as fast as control, evidence, and recovery mechanisms scale with it.

Contents

Executive summary

Agentic adoption is moving faster than enterprise trust. Docker's 2026 survey of 805 technical practitioners and decision-influencers reported that 60 percent of represented organizations already had agents in production and 94 percent treated agent building as a strategic priority. The same report found security and compliance concerns to be the leading barrier, cited by 40 percent; 48 percent identified the operational complexity of coordinating multiple components as the leading orchestration challenge; and 79 percent operated agents across two or more environments. These are leading-edge, self-reported results rather than a census of the broader market, but the pattern is clear: capability is being distributed before governance is consolidated. [1]

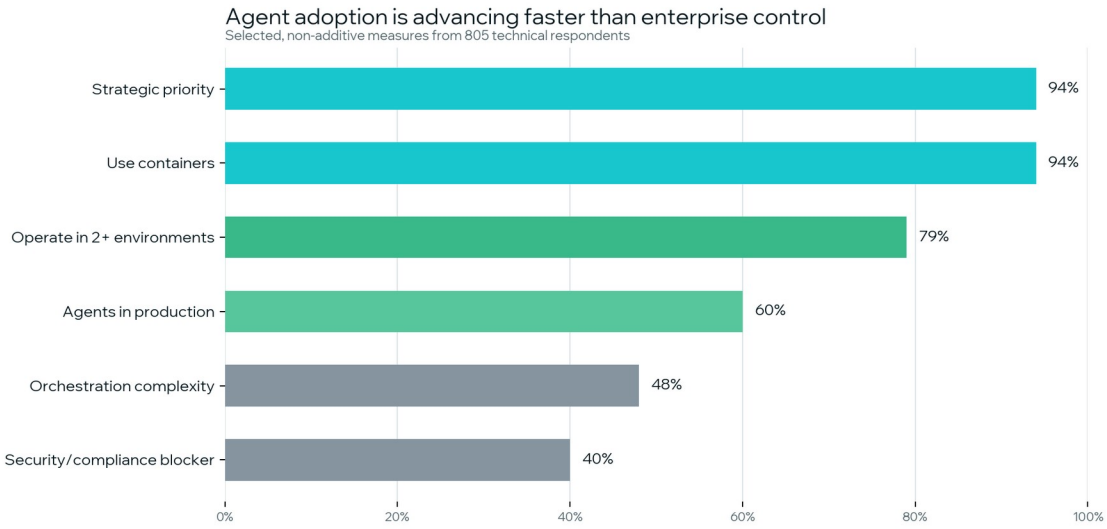

Figure 1. Adoption is advancing faster than enterprise control. Percentages are distinct survey measures and should not be summed. Source: Docker, 2026.

Existing agent frameworks are optimized to help a system plan and act. Security tools are generally optimized to protect hosts, networks, identities, software supply chains, or data. Governance programs define responsibilities and controls, but frequently operate outside the execution path. ShadowCore connects these layers at the moment a proposed action becomes a real side effect.

The architecture has four planes:

- **Control plane:** policy, identity, tool registration, risk scoring, approval, budgets, and kill switches.
- **Execution plane:** isolated workers, local or cloud model adapters, MCP gateways, direct connectors, network policy, and ephemeral credentials.
- **Assurance plane:** plan inspection, adversarial review, simulation, runtime monitoring, post-condition verification, and rollback decisions.
- **Evidence plane:** append-only event records, artifact hashes, policy decisions, approvals, tool receipts, verification results, and exportable control evidence.

Each action travels through a controlled lifecycle: declare intent, create a plan, inspect it, classify risk, approve when required, issue a scoped capability, execute in isolation, verify the result, and record or roll back. An agent never receives ambient authority merely

because the user is authorized. The authority is narrowed to the specific action, resource, time window, and expected post-condition.

The first commercial wedge is an **agent execution gateway** for internal coding, security, and operational agents. It does not replace existing agent frameworks. It wraps their tool calls and provides policy enforcement, approval, isolation, evidence, and recovery. This lowers integration cost and makes a pilot measurable.

The minimum credible product is not a dashboard alone. It must demonstrate five properties end to end:

1. a consequential tool call cannot bypass the policy gate;
2. an approval is specific enough for a human to understand the side effect;
3. credentials are temporary and narrower than the user's standing privilege;
4. the outcome is independently verified against declared post-conditions; and
5. the resulting evidence can reconstruct what happened without relying on the model's narrative.

1. From model risk to action risk

1.1 The new unit of consequence

A language model response can be inaccurate, unsafe, biased, or misleading. An agent inherits all of those failure modes and adds new ones: it can chain errors across steps, select an unsafe tool, inherit excessive privilege, consume unbounded resources, trust poisoned memory, accept a forged inter-agent message, or continue acting after the original intent has drifted.

The danger does not require a malicious model. A helpful agent with a vague objective, broad credentials, and an ambiguous tool can produce the same practical damage as an attacker. The system must therefore distinguish four different questions:

- **Capability:** Can the model or tool perform the action?
- **Authorization:** Is this principal permitted to perform it?
- **Intent:** Is this action a faithful and necessary step toward the declared goal?
- **Assurance:** Did the action produce the expected result without violating constraints?

Most agent stacks answer the first question well. Traditional identity systems partially answer the second. The third and fourth remain fragmented across prompts, application code, manual reviews, logs, and after-the-fact incident response.

1.2 Why familiar controls are insufficient by themselves

Conventional controls remain necessary, but they are not sufficient when natural-language planning selects and sequences actions at runtime.

- Identity and access management can establish who or what is calling, but a valid identity can still misuse a valid tool.

- Endpoint security can observe processes, but a harmful workflow may consist entirely of trusted binaries and authorized APIs.
- Network controls can restrict destinations, but they do not prove that a permitted transfer is consistent with user intent.
- Application logs can show that a call occurred, but not necessarily which retrieved instruction influenced the decision, which policy version authorized it, or whether the result was verified.
- Human approval can reduce risk, but only if the reviewer sees a comprehensible diff, material side effects, data movement, cost, reversibility, and the exact authority being granted.

The missing element is an execution-time control layer that treats plans and model outputs as untrusted proposals rather than instructions.

1.3 Market signal: fragmentation is becoming architecture

The Docker survey found that 61 percent of respondents combined local and managed-cloud models, 98 percent used more than one model, and 79 percent ran agents across multiple environments. Ninety-four percent used containers in development or production. [1] These numbers reinforce two design requirements.

First, control cannot be coupled to one model provider or orchestration library. A policy boundary that disappears when the model changes is not a durable boundary. Second, local execution is not a fringe exception. Organizations combine local and hosted resources for privacy, control, latency, cost, and compliance. Governance must follow the workload across those boundaries.

2. The ShadowCore thesis

2.1 Core claim

Every material side effect initiated by an AI agent should be mediated by a policy-enforced, evidence-producing control loop.

The loop must remain separate from the model proposing the action. A model may explain why it believes an action is appropriate, but its explanation is not authorization and is not proof. The control plane evaluates structured facts: principal, resource, tool version, arguments, data classification, destination, budget, risk tier, approval status, policy version, execution environment, and expected post-conditions.

2.2 Design principles

Control before autonomy

Autonomy is a privilege granted to a bounded workflow, not a default property of an agent. New tools, broader scopes, higher budgets, new destinations, or persistent memory require explicit control changes.

Evidence before trust

Trust is earned through reproducible evidence: signed artifacts, verified tool identity, explicit decisions, immutable receipts, post-condition checks, and successful recovery tests. A polished natural-language explanation is not evidence.

Least agency, not only least privilege

An agent should receive no more autonomy than the task requires. Some workflows need a model to recommend; fewer need it to execute; fewer still need it to execute without confirmation. OWASP's 2026 guidance makes the same distinction by extending least privilege to least agency. [7]

Intent and authority travel together

Every execution request carries a machine-readable intent capsule containing the goal, constraints, authorized resources, prohibited outcomes, risk budget, expiry, and expected result. Delegation cannot expand the capsule.

Independent verification

The component that proposes an action should not be the sole component that declares success. Verification uses deterministic checks where possible and separate evaluators where judgment is unavoidable.

Reversibility by design

The platform prefers previews, transactions, branches, snapshots, staged deployment, delayed publication, soft deletion, and compensating actions. Irreversible operations receive higher risk tiers.

Sovereignty is a deployment property

Sensitive prompts, documents, embeddings, logs, and evidence can remain local. Central governance may distribute signed policy and receive minimal attestations without collecting raw content.

Fail closed on ambiguity

Unknown tools, unsigned versions, unresolved identities, stale approvals, schema mismatches, destination changes, and verifier disagreement cause pause or denial. Operational availability exceptions must be explicit and separately governed.

2.3 Scope and non-goals

ShadowCore governs tool-using AI workflows. It is not a foundation model, a replacement for endpoint protection, a compliance certificate, or a promise that agents cannot fail.

The initial product explicitly excludes:

- covert surveillance, hidden profiling, or non-consensual analysis of individuals;
- automated lie detection or high-stakes behavioral inference;
- manipulation designed to exploit psychological vulnerability;

- unrestricted offensive cyber operations;
- autonomous weapons targeting or kinetic control;
- unreviewed financial transfers, destructive infrastructure changes, or public communications; and
- collection of raw customer content when an attestation or local computation can satisfy the control objective.

ShadowCore originated in broader exploration of context-aware assistants, behavioral prediction, local AI, and multi-agent software generation. That work exposed a more defensible product need: before systems are given deeper context and greater agency, users need a sovereign mechanism to see, constrain, approve, and prove what those systems do.

3. Reference architecture

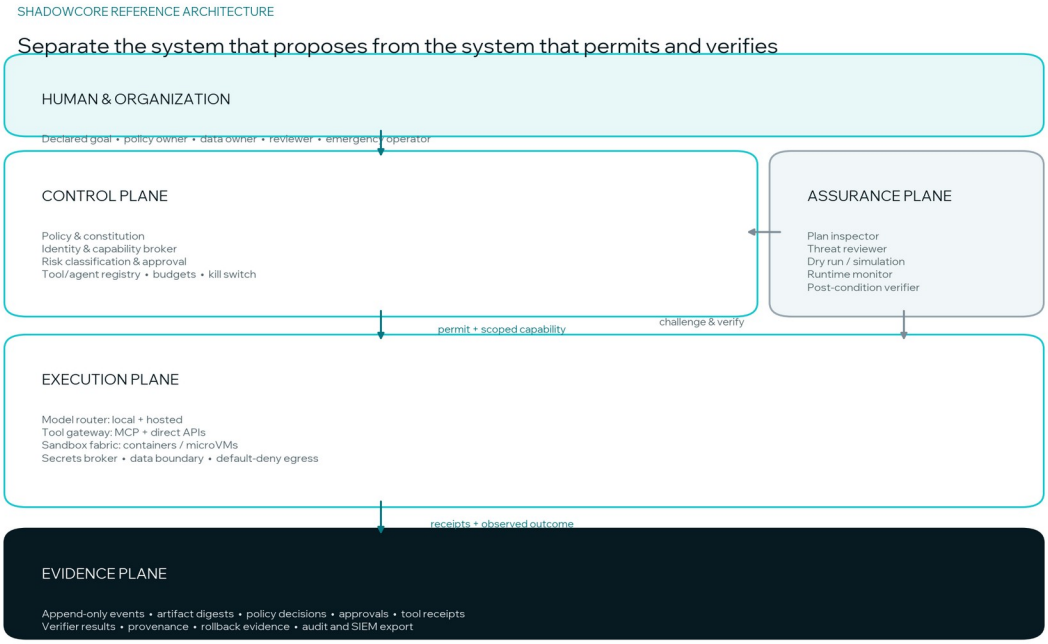

Figure 2. ShadowCore separates policy, execution, assurance, and evidence so that no model authorizes or validates its own side effects.

3.1 Control plane

The control plane is the policy decision and coordination layer. It should be small enough to audit and independent enough that an agent cannot rewrite its own constraints.

Policy and Constitution Service. Stores versioned organizational policy, workflow-specific invariants, prohibited actions, data handling rules, approval thresholds, and emergency controls. Policies compile to enforceable rules rather than remaining prose alone.

Identity and Capability Broker. Resolves the human principal, agent identity, service identity, device or workload identity, and delegation chain. It issues short-lived, action-

specific credentials only after a permit decision. NIST zero trust guidance rejects implicit trust based only on network location or ownership; ShadowCore applies the same principle to agents and tools. [4][5]

Tool and Agent Registry. Records tool identity, publisher, version, schema, permissions, network destinations, risk class, provenance, signature status, health, and allowed deployment zones. Tool descriptions supplied by an MCP server are treated as untrusted metadata until validated.

Risk Engine. Classifies each proposed action using impact, reversibility, data sensitivity, privilege, external exposure, novelty, cost, and confidence. Risk is evaluated per action, not only per agent.

Approval Service. Produces a human-readable approval packet: requested side effect, affected resources, data movement, before/after diff, estimated cost, recovery plan, expiry, and reason for escalation. Approval is cryptographically bound to the exact action request.

Budget and Circuit Breaker Service. Enforces time, token, API, financial, file, process, and network budgets. It can pause a workflow, revoke credentials, isolate a worker, or stop an entire agent class.

3.2 Execution plane

The execution plane performs permitted work inside defined boundaries.

Sandbox Fabric. Runs code and tools in ephemeral containers, microVMs, or equivalent isolation domains. Default-deny egress, read-only mounts, resource quotas, syscall restrictions, and disposable workspaces reduce blast radius. Containerization is a practical substrate, but isolation strength must match the threat model; a container alone is not automatically a security boundary.

Model Router. Chooses among local and hosted models based on data classification, capability, latency, cost, and policy. It can require local inference for restricted content and permit cloud inference only for redacted or approved context.

Tool Gateway. Normalizes direct APIs, MCP servers, command-line tools, and internal services behind a common invocation envelope. It validates schemas, fully qualifies tool identity, pins versions, checks destinations, and blocks token passthrough. Current MCP security guidance explicitly warns about confused-deputy flows, audience-invalid tokens, SSRF, session hijacking, compromised local servers, and overbroad scopes. [9]

Secrets Broker. Injects ephemeral credentials at execution time. Secrets do not enter model context, persistent memory, or general logs. Credentials are audience-bound, scope-bound, time-bound, and revoked after completion or policy drift.

Data Boundary Enforcement. Applies classification labels, redaction, field-level filtering, residency rules, egress allowlists, and retention limits before data enters a model, tool, or external destination.

3.3 Assurance plane

The assurance plane challenges the proposed workflow and independently checks the result.

Planner Inspector. Converts free-form plans into a typed action graph; rejects missing dependencies, unbounded loops, hidden side effects, inconsistent assumptions, and unsupported tool claims.

Threat Reviewer. Tests the plan against prompt injection, goal hijack, privilege escalation, tool poisoning, data exfiltration, cost amplification, memory contamination, inter-agent spoofing, and recovery failure.

Arbiter or Judge. Resolves disagreements among planner, inspector, policy engine, and verifier. Model-based arbitration can advise, but deterministic policy remains authoritative and material exceptions require a human.

Simulation and Dry Run. Executes against mocks, snapshots, staging systems, or read-only previews. The result becomes part of the approval packet.

Runtime Monitor. Watches tool sequences, destination changes, rate anomalies, policy drift, budget consumption, and declared invariants. It can suspend execution between atomic steps.

Post-condition Verifier. Checks the declared outcome using tests, API reads, checksums, diffs, deployment health, policy queries, or a separate evaluation model. If verification fails, the workflow enters containment or rollback rather than simply asking the planner whether it succeeded.

3.4 Evidence plane

The evidence plane reconstructs the execution independently of conversational memory.

Each event records:

- workflow, action, parent, principal, agent, and tool identifiers;
- intent capsule digest and policy version;
- model and prompt-template versions without requiring raw sensitive content;
- input and output artifact digests;
- risk classification and reasons;
- approval identity, scope, timestamp, and expiry;
- credential scope and lifetime, never the secret itself;
- sandbox image, configuration, and supply-chain provenance;
- tool request and response receipts, with governed redaction;
- verifier method, result, confidence, and exceptions; and
- rollback or containment actions.

Records are append-only and hash-linked. A Merkle-tree or transparency-log construction can provide efficient tamper evidence, but cryptography does not guarantee that the original event was truthful. Evidence quality still depends on trusted instrumentation, time sources, identity, and control coverage.

Software supply-chain controls can reuse established patterns. SLSA provenance records where, when, and how artifacts were built; in-toto records authorized steps and materials; Sigstore supports signing and transparency-log verification. [10][11][12] ShadowCore

applies analogous provenance to agent bundles, policy packages, prompts, tool manifests, and execution receipts.

4. Controlled execution lifecycle

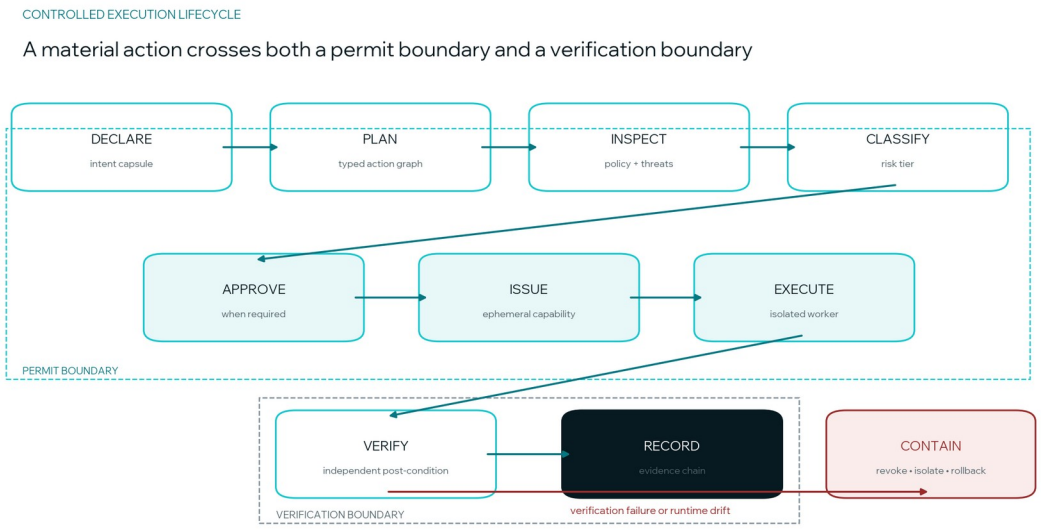

Figure 3. Every material action crosses a permit boundary and a verification boundary. Failure routes to pause, containment, or rollback.

4.1 The intent capsule

An intent capsule is a signed, typed envelope that binds purpose to authority. A minimal capsule contains:

```
{
       "goal":    "Update    dependency    X    and    open    a    draft    pull    request",
                                          "principal":                    "user:andrey",
                                            "agent":                   "agent:coder:v0.3",
                        "allowed_resources":            ["repo:shadowcore/control-plane"],
       "allowed_actions":    ["read",    "branch",    "edit",    "test",    "open_draft_pr"],
          "prohibited_actions":     ["merge",     "publish_package",     "modify_secrets"],
                                              "data_class":                      "internal",
         "budget":    {"wall_minutes":     20,     "api_usd":     3,     "egress_mb":     10},
                                     "approval_policy":                  "risk-tier-2",
      "expected_postconditions":    ["tests_pass",    "no_secret_diff",    "draft_pr_exists"],
                                       "expires_at":                 "2026-07-15T20:30:00Z"
}
```

The capsule prevents a common delegation failure: a parent agent may divide work, but a child cannot expand the original goal, privilege, destination, budget, or retention period. Changes create a new capsule and a new policy decision.

4.2 Risk tiers

Tier	Default handling	Typical examples	Required controls
R0 Observe	automatic	search approved data, summarize logs, inspect configuration	read-only identity, data boundary, logging
R1 Reversible	automatic within policy	create branch, generate draft, change disposable sandbox	scoped capability, budget, post-condition check, automatic cleanup
R2 Consequential	explicit human approval	send message, open external PR, modify production-like config, access restricted record	approval packet, step-up authentication, dry run, rollback plan
R3 Critical	two-person or specialist approval; often staged	production deployment, financial action, privileged identity change, destructive operation	separation of duties, maintenance window, verified backup, live monitoring, independent verification
R4 Prohibited	deny	covert profiling, unauthorized exfiltration, disallowed targeting, unbounded persistence	logged denial, alert, review of attempted goal drift

Risk tier is dynamic. A nominally read-only action can escalate when it crosses tenants, accesses sensitive data, invokes an unknown tool, creates a large cost, or follows untrusted retrieved content.

4.3 Permit, execute, verify

The action state machine is deliberately explicit:

1. **Declare.** Normalize the user goal and obtain confirmation when the scope is ambiguous.
2. **Plan.** Produce a typed graph of proposed actions, dependencies, tools, data, destinations, and post-conditions.
3. **Inspect.** Validate the graph and challenge it against policy and the threat model.
4. **Classify.** Assign a risk tier and determine required evidence, simulation, and approval.
5. **Approve.** Bind the human or automated decision to the exact action digest and expiry.
6. **Issue.** Mint a temporary capability for one tool, resource, scope, and time window.
7. **Execute.** Run inside the selected isolation and data boundary.
8. **Verify.** Test post-conditions through an independent path.
9. **Record.** Append receipts, artifacts, decisions, and verifier results to the evidence chain.
10. **Recover.** On failure or drift, revoke, isolate, roll back, compensate, or escalate.

Approval fatigue is a security failure. ShadowCore therefore automates low-risk, repeatable, reversible actions while making high-risk approvals materially informative. The objective is not to ask a human to click more often; it is to ask at the correct boundary with enough context to decide.

5. Security and safety model

5.1 Trust assumptions

ShadowCore assumes the following may be compromised or incorrect:

- user-supplied and retrieved natural language;
- model output and model self-reports;
- persistent memory, embeddings, and summaries;
- tool names, schemas, descriptions, and return values;
- external MCP servers and direct integrations;
- another agent's identity or message;
- container images, dependencies, and build pipelines;
- approval links or sessions presented outside the trusted interface; and
- logs generated only by the component being investigated.

The trusted computing base should be limited to the policy evaluator, identity and capability broker, approval binding, core gateway, evidence writer, key management, and selected isolation primitives. Every addition to that base increases the assurance burden.

5.2 Threat-control mapping

OWASP's 2026 Top 10 for Agentic Applications identifies goal hijacking, tool misuse, identity and privilege abuse, agentic supply-chain vulnerabilities, unexpected code execution, memory and context poisoning, insecure inter-agent communication, cascading failures, human-agent trust exploitation, and rogue agents. [7] ShadowCore maps these risks to execution-time controls rather than relying on prompt rules alone.

Threat	Primary ShadowCore controls	Residual risk
Goal hijack	signed intent capsule, untrusted-input labelling, goal-drift detection, re-approval	semantic equivalence is imperfect; subtle drift may pass
Tool misuse	typed schemas, least agency, per-call policy, egress and cost budgets, dry run	legitimate operations can still have unexpected domain effects
Identity abuse	distinct agent identity, delegation chain, JIT credentials, audience binding, TOCTOU re-check	upstream identity compromise remains possible
Supply-chain compromise	signed agent/tool bundles, pinned digests, provenance, registry policy, quarantine	trusted publisher or build system may be compromised
Unexpected code execution	sandboxing, command allowlists, no raw model-to-shell path, syscall and egress limits	isolation escape or unsafe interpreter behavior
Memory poisoning	provenance, tenant and task segmentation, expiry, write approval, retrieval trust labels	semantically poisoned but plausible content
Inter-agent spoofing	mutual authentication, signed envelopes, nonce and replay protection, scope revalidation	compromised authorized peer

Threat	Primary ShadowCore controls	Residual risk
Cascading failure	action graph limits, circuit breakers, budgets, rate limits, idempotency, blast-radius zones	correlated dependency or policy failure
Human trust exploitation	factual approval packets, source labels, no urgency manipulation, step-up authentication	reviewer error and automation bias
Rogue agent	invariant monitor, independent verifier, revocation, isolation, kill switch	novel behavior within permitted envelope

MITRE ATLAS provides a living knowledge base of adversary tactics and techniques for AI-enabled systems. [8] An implementation should maintain a deployment-specific ATLAS mapping and convert relevant techniques into red-team cases, detection rules, and recovery drills.

5.3 Privacy and human autonomy

Sovereign execution is valuable because the most intimate data can also be the most useful to an assistant. That creates a direct design tension: context improves utility while expanding inference and manipulation risk.

ShadowCore applies four boundaries:

1. **Consent must be specific and revocable.** General acceptance of a long terms-of-service document is not sufficient for sensitive inference, recording, or secondary use.
2. **Local processing is preferred for intimate content.** Raw conversations, voice, biometrics, and personal archives should not leave the user's environment by default.
3. **Assistance must not silently become influence.** Suggestions should identify uncertainty and avoid optimizing for dependency, spending, persuasion, or emotional compliance.
4. **High-stakes inference requires a separate governance case.** Employment, insurance, law enforcement, healthcare, credit, intimate relationships, and deception detection are outside the initial product scope.

This boundary is not an incidental ethics statement. It reduces product risk and prevents the governance platform from becoming the very surveillance or manipulation system it is intended to control.

6. Governance and evidence

6.1 From policy documents to executable controls

NIST AI RMF organizes AI risk management through Govern, Map, Measure, and Manage. [2] The Generative AI Profile extends that structure with actions tailored to generative systems. [3] ISO/IEC 42001 specifies an organizational management system for responsible development and use of AI, including policy, objectives, risk treatment, performance evaluation, and continual improvement. [6]

ShadowCore does not replace these frameworks. It provides an execution and evidence layer that can help operationalize selected outcomes:

Governance objective	ShadowCore mechanism	Example evidence
Accountability	named human, agent, tool, policy, and approver identities	signed decision record and delegation chain
Risk-based treatment	per-action classification and control requirements	risk factors, tier, exceptions, mitigations
Transparency	visible plan, data sources, tools, destinations, and limits	approval packet and action graph
Human oversight	escalation at consequential boundaries	approval receipt, expiry, step-up authentication
Security and resilience	isolation, least agency, budgets, rollback	sandbox profile, credential scope, recovery result
Measurement	verifier outcomes and operational metrics	pass/fail results, drift events, incident trends
Continual improvement	policy versioning and control effectiveness review	change history, failed controls, updated test cases

The EU AI Act establishes harmonized, risk-based requirements for AI systems in the European Union. [13] Applicability depends on system role, context, risk classification, and deployment facts. ShadowCore can produce technical records useful to a compliance program, but it cannot determine legal classification or make a system compliant by itself.

6.2 Evidence is a product surface

Evidence should be useful to four different readers:

- **Operator:** What is running, what is paused, and what requires action now?
- **Security reviewer:** Which identity, tool, data, and destination boundaries were crossed?
- **Auditor or governance lead:** Which control operated, under which policy version, with what result?
- **Engineer:** Can the execution be reproduced, tested, and debugged without exposing unnecessary data?

The evidence model separates content from proof. A restricted document may remain local while the central record stores its classification, digest, local verifier result, policy decision, and retention status. This supports federated governance without building a central collection of sensitive prompts and outputs.

6.3 Control effectiveness metrics

Useful metrics measure control performance, not the volume of agent activity:

- percent of side-effecting actions mediated by the gateway;
- percent of credentials that are ephemeral and action-scoped;

- policy-bypass attempts detected in testing and production;
- approval precision: proportion of escalations judged materially necessary;
- verifier coverage by risk tier;
- rollback success rate and median containment time;
- unregistered or unsigned tool invocation attempts;
- goal-drift and memory-poisoning detection rate in red-team suites;
- evidence completeness and reconstruction success; and
- false-denial rate and operator time lost to controls.

NIST's work on testing, evaluation, verification, and validation reinforces the need for defined methods, measurements, and context rather than one universal safety score. [14] ShadowCore should publish evaluation protocols and limitations for each control claim.

7. Sovereign deployment model

7.1 Local-first does not mean unmanaged

Running a model on a workstation can improve privacy and availability, but it does not automatically provide isolation, identity, safe tool access, evidence, or recovery. ShadowCore treats local infrastructure as a first-class deployment target with the same control lifecycle as cloud infrastructure.

Home or individual node. A single-machine appliance runs local inference, the control plane, isolated workers, and an encrypted evidence vault. Cloud model calls are optional and policy-routed. The default network stance is no inbound public service and deny-by-default outbound access for workers.

Team node. A small organization runs the gateway and evidence service on-premises or in a private cloud. Users receive distinct identities and approvals; policies and tool registries are shared; sensitive workloads can remain on local GPU nodes.

Federated enterprise. Central teams publish signed policy packages and registry decisions. Regional or business-unit nodes execute locally and return minimal attestations. Raw data stays within the applicable zone unless a policy-authorized transfer occurs.

Regulated or disconnected zone. Models, tools, policies, signatures, and revocation data are imported through a controlled release process. Evidence is exported through a separate review boundary. Offline operation never silently falls back to unverified artifacts or stale authorization.

7.2 Deployment invariants

Every topology should preserve these invariants:

- no unauthenticated control API is exposed to the public internet;
- no worker receives standing secrets through prompt context or image layers;
- tool and agent versions are pinned by digest;

- policy changes are signed, versioned, reviewed, and reversible;
- evidence storage is separate from the agent's writable workspace;
- an operator can revoke, pause, and isolate without cooperation from the model;
- backups and recovery are tested, not merely configured; and
- local execution does not bypass approval or audit requirements.

8. Product surface

8.1 Command Center

The Command Center is the human control surface, not a decorative dashboard. Its primary views are:

Now: active workflows, pending approvals, blocked actions, incidents, budgets, and kill-switch state.

Plan: the action graph, tools, data, destinations, expected changes, and risk tier before execution.

Approve: a concise diff and consequence statement, with approve once, approve bounded class, edit scope, deny, or escalate.

Evidence: a chronological reconstruction linked to artifacts, decisions, tool receipts, verifier results, and recovery actions.

Policy: readable policy alongside compiled enforcement rules, tests, owners, change history, and affected workflows.

Registry: trusted agents, models, tools, MCP servers, versions, provenance, permissions, risk class, and revocation status.

8.2 Approval packet design

A good approval packet answers seven questions without requiring the reviewer to inspect a chain-of-thought transcript:

1. What will change?
2. Why is it necessary for the declared goal?
3. Which resources, identities, data, and external parties are involved?
4. What is the maximum plausible impact?
5. Can the action be previewed or reversed?
6. Which checks have already passed or failed?
7. Exactly what authority and time window does approval grant?

Approval is invalidated when the tool version, destination, material arguments, data classification, cost bound, or post-conditions change.

9. Initial use cases

9.1 Governed coding agent

The agent may inspect a repository, create a branch, edit files, run tests, and open a draft pull request. It cannot merge, modify production secrets, publish packages, or change CI permissions without a new intent capsule and higher-tier approval.

Verification includes test results, secret scanning, dependency and license checks, diff constraints, code-owner policy, and confirmation that the remote object is a draft. Evidence links commit, build, tool, policy, and approval digests.

9.2 Security operations assistant

The agent may summarize Zeek, endpoint, identity, and cloud telemetry and propose containment. Read-only correlation is R0. Isolating a host or disabling a token is R2 or R3 depending on environment. The approval packet shows affected users, dependency risk, time window, and recovery command.

The system does not permit an injected log line or malicious ticket to become an administrative instruction. Retrieved operational content remains untrusted data.

9.3 Regulated document workflow

The agent may classify incoming documents locally, extract permitted fields, and prepare a draft. External transmission requires a data-boundary decision and approval. Central evidence can record the digest, classification, extractor version, human review, and destination without retaining the document.

9.4 Sovereign personal AI appliance

A one-click local appliance can provide private search, drafting, planning, and bounded automation without opening public inbound ports or sending personal archives to a cloud provider by default. The commercial promise is not that local AI is inherently safe; it is that safe defaults, visible permissions, updates, recovery, and support are packaged as a maintained product.

10. Implementation roadmap

Phase 0: Control kernel prototype (0–6 weeks)

Build one vertical slice around a coding workflow:

- intent capsule schema and signing;
- policy decision point and tool gateway;
- three risk tiers plus prohibited actions;
- sandboxed worker with default-deny egress;
- one local model adapter and one hosted model adapter;

- one MCP integration and one direct API integration;
- human approval with action-digest binding;
- post-condition verifier; and
- append-only evidence export.

Exit criteria: demonstrated prevention of gateway bypass in the scoped workflow; expired or altered approvals rejected; credentials absent from model context and persistent logs; verifier detects seeded false-success cases; complete incident reconstruction from evidence.

Phase 1: Design-partner pilot (6–12 weeks)

Add identity integration, signed registry, policy tests, budget controls, rollback automation, and an operator Command Center. Run with one design partner in an internal, non-customer-facing workflow.

Exit criteria: at least 95 percent of scoped side effects mediated; zero standing production credentials held by agents; measurable reduction in manual reconstruction time; approval false-positive rate low enough for sustained use; recovery drills pass.

Phase 2: Enterprise hardening (3–6 months)

Add high-availability control services, federated nodes, tamper-evident evidence, enterprise identity, separation of duties, policy-as-code CI, supply-chain attestations, expanded MCP controls, SIEM export, retention policy, and independent security testing.

Exit criteria: documented threat model; external penetration test; failover and disaster-recovery evidence; control mappings reviewed by qualified practitioners; red-team coverage for applicable OWASP Agentic Top 10 and MITRE ATLAS techniques.

Phase 3: Sovereign appliance and ecosystem

Package local inference, gateway, policy, registry, evidence vault, updates, and recovery into a supported deployment. Publish an agent bundle format that carries identity, requested capabilities, policy requirements, provenance, tests, and verifier definitions.

The ecosystem goal is portability without policy loss: an agent can move between supported environments, but its declared capabilities and evidence obligations travel with it.

11. Product strategy and defensibility

11.1 Wedge

The entry product is a control gateway for side-effecting agent actions. It integrates with existing frameworks rather than asking teams to rebuild agents. A successful pilot proves policy mediation, approval quality, evidence completeness, and recovery in one internal workflow.

11.2 Expansion

Expansion follows control adjacency:

- more tools and model providers;
- organization-wide policy and registry;
- federated sovereign nodes;
- compliance evidence packs;
- adversarial evaluation suites;
- managed updates and signed agent bundles; and
- sector-specific profiles for regulated environments.

11.3 Defensibility

The defensible asset is not a prompt library. It is the accumulated control system: policy semantics, tool risk metadata, approval UX, verifier patterns, execution receipts, recovery playbooks, red-team cases, framework mappings, and integration reliability across heterogeneous environments.

Defensibility must not depend on capturing customer content. Privacy-preserving control telemetry can improve policies and detectors without centralizing raw conversations, source code, or regulated documents.

12. Research agenda and open questions

ShadowCore should remain explicit about unresolved problems.

Semantic intent validation. Deterministic policy can validate types and scopes, but intent is partly semantic. How should disagreement among user, planner, inspector, and policy be calibrated without turning another model into an oracle?

Verifier independence. Different models can share training data and failure modes. What combinations of deterministic tests, diverse models, and human review produce meaningful independence?

Approval calibration. How can the system minimize dangerous approvals and approval fatigue simultaneously? Which consequence representations help humans detect manipulation or hidden scope?

Memory governance. What provenance, expiry, correction, and partitioning mechanisms prevent persistent context from becoming an invisible source of privilege or bias?

Agent identity. How should identity persist across model changes, forks, delegated tasks, and offline nodes without creating unaccountable synthetic principals?

Evidence minimization. What is the minimum evidence required to reconstruct and audit an action while protecting content, trade secrets, and personal data?

Recovery for external side effects. Some actions cannot be rolled back. How should compensating controls, delay windows, escrow, or multi-party approval be standardized?

Control-plane compromise. A centralized control layer is high value. Which components can be formally minimized, isolated, replicated, or independently verified?

Conclusion

Agentic systems will not become trustworthy because models learn to sound more certain. They become governable when authority is explicit, execution is bounded, side effects are mediated, outcomes are independently checked, and evidence survives the conversation that produced the action.

ShadowCore AI proposes a practical trust layer for that transition. It does not require one model, one cloud, or one agent framework. It requires a consistent invariant: no consequential action without scoped authority, policy evaluation, appropriate approval, isolated execution, post-condition verification, and an auditable record.

The immediate opportunity is narrow enough to build: wrap one internal agent workflow with a control gateway and prove that control loop end to end. The longer-term platform is broader: a sovereign operating layer through which people and organizations can grant AI systems useful autonomy without surrendering visibility, revocation, or accountability.

Control before autonomy. Evidence before trust.

Appendix A. Minimum control set for an MVP

Control domain	Minimum implementation	Test evidence
Identity	distinct user, agent, tool, and workload identities	forged or missing identity denied
Authorization	action-, resource-, audience-, and time-scoped capability	altered scope and expired token denied
Policy	versioned deny, allow, escalate, and budget rules	regression suite and signed policy digest
Tool trust	registered schema, pinned version, destination and permission declaration	unknown or changed tool quarantined
Isolation	disposable worker, restricted filesystem, quotas, default-deny egress	prohibited write and egress blocked
Secrets	JIT injection outside model context and logs	secret scanning and revocation test
Approval	consequence packet bound to action digest	material argument change invalidates approval
Verification	deterministic post-condition check for the pilot workflow	seeded false-success case detected
Evidence	append-only action chain with governed redaction	independent reconstruction succeeds
Recovery	revoke, stop, isolate, clean up, and restore path	timed recovery drill passes

Appendix B. Evidence event model

A canonical event envelope may contain the following fields:

Field group	Representative fields
Correlation	event_id, workflow_id, action_id, parent_action_id, trace_id
Time	proposed_at, permitted_at, started_at, ended_at, trusted_time_source
Identity	principal_id, agent_id, workload_id, tool_id, approver_id, delegation_digest
Intent	intent_digest, goal_class, constraints_digest, expected_postconditions
Policy	policy_bundle_digest, decision, rule_ids, risk_tier, exception_id
Execution	sandbox_digest, model_id, tool_version, destination, capability_id, budget_usage
Artifacts	input_digests, output_digests, provenance_refs, redaction_manifest
Verification	verifier_id, method, expected, observed, result, confidence, exception
Recovery	revocation_id, containment_action, rollback_ref, recovery_result
Integrity	previous_event_hash, event_hash, signer, signature, retention_class

Raw prompts, chain-of-thought, secrets, and sensitive payloads should not be stored by default. When content retention is justified, it should have a separate classification, access policy, expiry, and consent basis.

Appendix C. Framework alignment summary

Source	ShadowCore use	Limitation
NIST AI RMF 1.0	organize governance, context, measurement, and treatment outcomes	voluntary framework; mapping is not validation
NIST AI 600-1	generative-AI risk actions and documentation	profile is broader than the MVP
NIST SP 800-207/207A	identity-centric, per-request zero-trust design	agent semantics require additional controls
ISO/IEC 42001	management-system responsibilities and continual improvement	certification requires independent conformity assessment
OWASP Agentic Top 10	threat taxonomy and mitigations for agentic applications	risks evolve; deployment threat model remains necessary
MITRE ATLAS	adversary tactics and techniques for testing and monitoring	technique coverage does not prove safety
MCP specification and guidance	authorization, consent, token, session, SSRF, and local-server controls	MCP implementations and versions vary
SLSA, in-toto, Sigstore	provenance, authorized steps, signatures, and transparency patterns	software provenance does not validate model behavior
EU AI Act	risk-aware documentation and oversight context	legal applicability is fact-specific

References

1. Docker. *The State of Agentic AI Report*. 2026. Survey of 805 respondents. https://www.docker.com/resources/the-state-of-agentic-ai-white-paper/
2. National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1. 2023. https://doi.org/10.6028/NIST.AI.100-1
3. Autio, C., et al. *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*, NIST AI 600-1. 2024; NIST page updated April 8, 2026. https://doi.org/10.6028/NIST.AI.600-1
4. Rose, S., Borchert, O., Mitchell, S., and Connelly, S. *Zero Trust Architecture*, NIST SP 800-207. 2020. https://doi.org/10.6028/NIST.SP.800-207
5. Chandramouli, R., and Butcher, Z. *A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments*, NIST SP 800-207A. 2023. https://doi.org/10.6028/NIST.SP.800-207A
6. International Organization for Standardization. *ISO/IEC 42001:2023 — Information technology — Artificial intelligence — Management system*. https://www.iso.org/standard/42001
7. OWASP GenAI Security Project, Agentic Security Initiative. *OWASP Top 10 for Agentic Applications 2026*. December 2025. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/
8. MITRE. *MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems*. Accessed July 15, 2026. https://atlas.mitre.org/
9. Model Context Protocol. *Security Best Practices* and *Authorization Specification*. Accessed July 15, 2026. https://modelcontextprotocol.io/docs/tutorials/security/security_best_practices and https://modelcontextprotocol.io/specification/2025-11-25
10. SLSA. *Build Provenance, Specification v1.2*. Accessed July 15, 2026. https://slsa.dev/spec/v1.2/build-provenance
11. Sigstore. *Cosign Signing and Verification Documentation*. Accessed July 15, 2026. https://docs.sigstore.dev/cosign/
12. in-toto. *A Framework to Secure the Integrity of Software Supply Chains*. Accessed July 15, 2026. https://in-toto.io/
13. European Union. *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence*. 2024. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng
14. National Institute of Standards and Technology. *Outline: Proposed Zero Draft for a Standard on AI Testing, Evaluation, Verification and Validation*. 2025. https://www.nist.gov/document/outline-proposed-zero-draft-standard-ai-testing-evaluation-verification-and-validation

Author: Andrey Zaykovskiy
Organization: ShadowCore AI, Vancouver, Canada
Web: https://shadowcore.ai
Contact: siberianspirit@proton.me