Bounded by Design

The Architecture of Agentic Security and What the Terminology Obscures

Michael J Martin

Outlan.net

May 2026

Keywords: agentic AI, security operations, SIEM, SOAR, event-driven architecture, distributed systems, large language models, orchestration, retrieval-augmented generation, failure domains

Abstract

Security products described as "agentic," "autonomous," or "self-directed" often combine established telemetry, analytics, workflow, and policy infrastructure with newer model-driven planning and language interfaces. The established components do not make the resulting system non-agentic: an agent can select and sequence actions inside a bounded architecture. The more consequential question is whether claims about autonomy, generality, persistence, independent planning, and authority match the product's execution semantics. This paper offers a systems-engineering framework for answering that question. It distinguishes agenticity from autonomy, capability, open-endedness, safety, and architectural novelty; traces the mixed lineage from intrusion detection and log management through SIEM, SOAR, graph analytics, retrieval, and model-mediated orchestration; and analyzes the control boundary between probabilistic decisions and consequential actions. It proposes a multidimensional maturity model rather than a binary test, expands the threat model for retrieval and tool use, and specifies lifecycle controls for traceability, least privilege, validation, blast-radius management, incident response, and human oversight. Large language models provide genuine value in summarization, retrieval, explanation, and analyst acceleration. That value is strongest when evidence provenance and authority remain explicit. "Agentic" is therefore best treated as a description of degree and behavior, not as a substitute for architecture.

1. INTRODUCTION

Security operations are constrained by scale, incomplete observation, and time. Defenders must turn heterogeneous telemetry into evidence, relate weak indicators across entities and time, decide whether the resulting pattern warrants intervention, and act without creating a larger incident. The problem predates cloud computing and generative AI. What has changed is the volume and diversity of evidence, the number of systems that can be queried or changed, and the availability of models that can interpret language and choose among tools.

The architecture of a current platform may include collectors, durable event streams, parsers, identity resolution, entity state, graph and time-series stores, detection engines, vector indexes, case systems, language models, policy decision points, workflow engines, and response connectors. Some of these elements are deterministic; others are probabilistic; all can fail. Their composition—not the label on the product—determines who or what forms goals, how actions are selected, what authority is exercised, what evidence supports a conclusion, and how failure propagates.

The thesis of this paper is deliberately narrower than "bounded systems are not agents." Boundedness is compatible with agentic behavior and is usually desirable in security. An agent that observes results, updates a plan, and chooses its next tool call can be meaningfully agentic even when every tool and permission is predefined. The concern is overstatement: products may imply broader autonomy, persistence, generality, learning, independent planning, or authority than their execution model provides. The proper evaluation unit is the end-to-end system, including its state, policies, connectors, humans, and operating environment.

Scope and method

This is an architecture position paper, not a product census, controlled benchmark, or empirical estimate of market prevalence. It synthesizes foundational research, public technical documentation, and current risk-management and application-security guidance, including NIST's AI Risk Management Framework and Generative AI Profile, MITRE ATT&CK and ATLAS, OWASP guidance, and original work on ReAct and retrieval-augmented generation. It uses those sources to derive design questions and control requirements; it does not claim that every vendor uses the same architecture or that any architecture is safe merely because it implements a listed control. NIST frames AI risk management as contextual and lifecycle-oriented, which is consistent with the paper's emphasis on system boundaries and operating conditions [1][2].

2. TERMS THAT SHOULD NOT BE COLLAPSED

The vocabulary is useful only if its dimensions remain separate.

Agenticity is the degree to which a system pursues goals through a feedback loop: it observes, maintains task-relevant state, selects and sequences actions, evaluates results, and revises behavior. Agenticity can be narrow, temporary, and policy-bounded.

Autonomy is the degree of discretion the system may exercise without contemporaneous human authorization. It concerns authority, duration, scope, and reversibility. A system can be highly agentic in investigation while having little autonomous remediation authority.

Capability is the demonstrated ability to perform a task under stated conditions. A capable model is not necessarily authorized, persistent, safe, or agentic. Conversely, a simple system can exercise dangerous authority despite modest reasoning capability.

Open-endedness is the extent to which goals, plans, action types, environments, and stopping conditions are not pre-enumerated. It is not a synonym for intelligence. In production security, constrained action spaces often reflect sound engineering.

Safety is the control of unacceptable risk over the lifecycle, including security, privacy, reliability, human factors, and organizational impact. Safety is not the absence of nondeterminism; it is evidence that hazards are identified, bounded, monitored, and recoverable.

Architectural novelty concerns whether a system introduces materially different execution, state, learning, or control primitives. It is independent of user-visible capability. A new semantic interface over an established workflow engine may be valuable without being a new control-plane architecture. A model-driven action selector may be architecturally consequential even if the final action is executed by an old connector.

These dimensions prevent two symmetric errors. The first is to treat every tool-using assistant as broadly autonomous. The second is to deny agenticity whenever tools, permissions, or goals are bounded. A defensible description should state where the system sits on each dimension and under what conditions.

3. A COMPLEX LINEAGE: IDS, LOG MANAGEMENT, SIEM, SOAR, AND AGENTS

Modern security platforms did not descend through a single, clean sequence. Network intrusion detection contributed packet inspection, signatures, alerts, and sensor operations; Snort's 1999 paper is an influential example [3]. Host and application logs contributed records of identity, administration, and business activity. Log-management platforms contributed centralized collection, retention, search, integrity practices, and compliance reporting. Security information management and security event management emphasized, in different proportions, longer-term reporting and near-real-time monitoring. SIEM joined and commercialized parts of those traditions while expanding to multiple telemetry sources, normalization, correlation, alerting, investigation, and reporting. NIST's log-management guidance describes generation, transmission, storage, analysis, and disposal as an infrastructure-wide discipline and identifies centralized SIEM capabilities across varied log types [4].

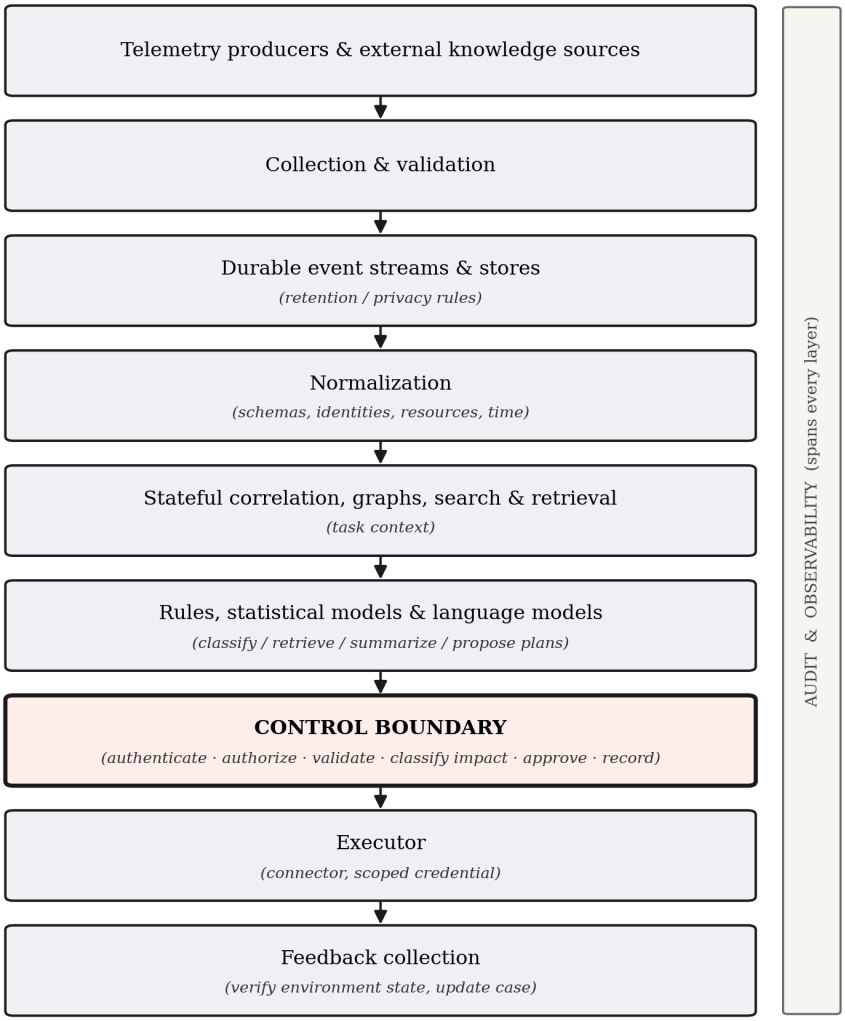
SOAR addressed another bottleneck: repeatable investigation and response across products. A playbook can gather identity and asset context, query endpoints, branch on evidence, request approval, invoke a containment API, update a case, and verify completion. This is distributed workflow engineering: state transitions, retries, timeouts, authorization, idempotency, compensating actions, and audit.

Current agentic designs add model-mediated interpretation and action selection to this lineage. ReAct is an influential framework because it interleaves language-model reasoning traces with actions and observations, allowing a model to update a plan as information arrives [5]. It did not originate or formalize all planner–executor architecture; planning, workflow, control, and feedback long predate it. Its importance here is narrower: it demonstrates a practical model-mediated loop that can choose among tools rather than merely fill a fixed template.

Continuity does not imply identity. A model that dynamically decides which evidence to seek, changes a hypothesis after tool output, and chooses a different next action is behaviorally different from a static playbook. At the same time, the agent's effectiveness still depends on telemetry quality, state, permissions, and connector behavior. The useful historical conclusion is therefore not that the architecture "was already complete," but that newer agentic behavior is composed with an extensive operational substrate whose semantics remain decisive.

4. REFERENCE ARCHITECTURE: EVIDENCE, DECISIONS, AND CONTROL

Figure 1. Layered reference architecture for AI-augmented security operations.



The reference architecture separates evidence, interpretation, decision, control, action, and assurance. Telemetry producers and external knowledge sources enter through collection and validation. Durable streams and stores preserve evidence according to retention and privacy rules. Normalization resolves schemas, identities, resources, and time semantics. Stateful correlation, graphs, search, and retrieval construct task context. Rules, statistical models, and language models then classify, retrieve, summarize, or propose plans.

The control boundary sits between a proposal and a consequential side effect. It authenticates the calling workload and user, authorizes the requested operation, validates the target and parameters, classifies impact, checks budgets and rate limits, determines whether approval is required, and records the decision. The executor invokes a connector with scoped credentials. Feedback collection verifies the resulting environment state and updates the case, while audit and observability span every layer.

This separation is analytical rather than absolute. A normalization model may be probabilistic. A policy can contain ambiguous or stale data. A connector may return success before the target converges. A conventional distributed workflow may exhibit nondeterminism, partial failure, and weak observability. "Deterministic" in this paper therefore means that a transition is governed by explicit code and policy for a given recorded input and state—not that the entire distributed system is perfectly predictable or inspectable.

5. TELEMETRY INTEGRITY AND TEMPORAL SEMANTICS

The first limit on any investigation is the evidence that arrived. Missing endpoint events, delayed audit feeds, duplicate records, parser failures, schema drift, sampling, clock error, and unauthorized source changes create uncertainty that fluent output can hide. The pipeline should expose coverage, freshness, parser health, deduplication, source authenticity, and data-quality indicators to downstream reasoning and to analysts. Uncertainty about evidence is itself evidence.

Durable, replayable logs are valuable because they decouple producers from consumers, absorb bursts, and permit later analytics to reprocess retained records. Kafka, for example, implements ordered partitions rather than one global order; consumer position is represented by offsets in each partition [6]. A security design must therefore state the partition key and the ordering property it actually needs. Ordering events for one endpoint may be tractable when keyed by endpoint, while ordering actions across endpoint, identity, cloud, and network sources requires correlation under uncertainty.

At least three clocks matter. **Event time** is when the source says an event occurred. **Ingestion time** is when the platform accepted it. **Processing time** is when a computation handled it. These values diverge under buffering, clock skew, disconnected agents, retries, and backfill. Stream processors use windows, watermarks, triggers, and allowed-lateness policies to trade latency against completeness; late data may revise an earlier result instead of fitting a final global sequence. Apache Beam's programming model and the Dataflow model make these tradeoffs explicit for unbounded, out-of-order streams [7][8].

Security narratives should preserve those semantics. "A token was issued before a process connected" may mean event-time order within an uncertainty interval, not a globally established causal sequence. A model should not silently convert ingestion order into causality. Correlation results need timestamps, clock source, skew assumptions, window definitions, late-arrival status, and revision history where those facts affect the conclusion.

Replay also needs precise language. A retained telemetry stream supports reprocessing; it is not necessarily **event sourcing**. Event sourcing captures every change to an application's authoritative state as a sequence of events from which that state can be reconstructed [9]. A security data lake may retain observations without making them the source of truth for the case system, entity graph, policy state, or external environment. Operational replayability therefore depends on what was captured and which states can be reconstructed, not merely on the existence of a message log.

6. NORMALIZATION, STATEFUL CORRELATION, AND GRAPHS

Normalization is an epistemic boundary. Parsers decide that fields from different products refer to the same concept; identity resolution decides that names refer to the same principal; asset resolution decides that cloud IDs, hostnames, and addresses refer to the same resource. These mappings are versioned interpretations, not clerical transformations. Their confidence, source, and effective time should be retained.

Stateful correlation constructs meaning from sequence and context. A failed login has different significance when related to privilege, device posture, geography, prior behavior, and subsequent access. The relevant "memory" may be an entity store, time-series database, graph, vector index, case record, model context, or durable log. Those stores have different consistency, retention, access-control, and causal properties. Calling all of them "agent memory" obscures which source is authoritative and how it can be corrected.

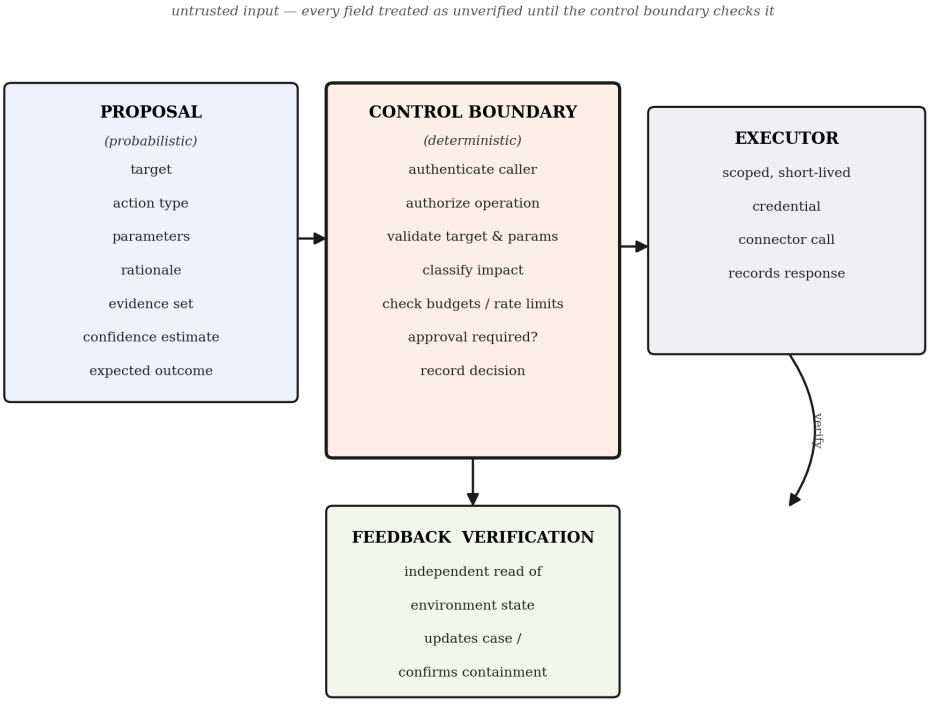
Graphs are useful because identities, resources, processes, sessions, permissions, and communications are relational. Paths can expose privilege chains, lateral movement hypotheses, dependencies, or likely blast radius. Yet graph limitations depend on representation and query design. A property graph can model intervals, versions, and provenance; a temporal graph database can support time-aware traversal; a snapshot graph may erase change. The relevant questions are whether edges carry direction, source, confidence, and valid time; whether queries distinguish observation from inference; and whether deleted or superseded relationships remain available for reconstruction.

A graph path is not automatically a causal proof. A vector neighbor is not automatically a precedent. A case summary is not automatically ground truth. The system should preserve the transformation chain from source event to normalized record, entity resolution, derived edge, retrieved context, model claim, analyst decision, and action. Provenance makes uncertainty inspectable instead of allowing polished language to convert an inference into an apparent fact.

7. PLANNING, ORCHESTRATION, AND THE CONTROL BOUNDARY

An agentic security workflow can be bounded and still be agentic. Within an allowed catalog, a model may choose which query to run, interpret the result, update a hypothesis, select the next action, and stop when evidence is sufficient. The meaningful boundary is not "model versus no model." It is where a model-generated output becomes an input to authority.

Figure 2. Probabilistic-to-deterministic control boundary.



The proposal side may generate a target, action type, parameters, rationale, evidence set, confidence estimate, and expected outcome. The control side should treat each as untrusted input. It resolves identities and targets, validates the request against a strict schema, evaluates policy, checks separation of duties and approval requirements, computes blast radius, enforces rate and spend limits, and issues a short-lived credential scoped to the approved operation. The executor then performs the call and records the response. A verifier checks the environment, not just the HTTP status, before the case is marked contained.

A deterministic policy engine does not make the end-to-end system deterministic when model output materially influences which action is selected, which target is chosen, or when execution occurs. It makes the final gate explicit and testable. This distinction matters for assurance: policy tests can prove that forbidden operations are denied under modeled conditions, but they do not prove that allowed model-selected operations are correct.

Likewise, identical-trajectory reproducibility is generally the wrong target. Models, search indexes, external APIs, and concurrent systems can change or behave nondeterministically. The operational target is **traceability and replayability**: retain model and provider version, inference settings, system and user prompts, retrieved passages with identifiers and hashes, tool definitions, tool calls and outputs, policy versions and decisions, human approvals, generated outputs, state transitions, errors, and timestamps. With these artifacts, investigators can reconstruct why a decision was made, replay deterministic portions, compare a model or policy revision, and identify where divergence occurred without promising bit-for-bit repetition.

8. GENUINE LLM VALUE—AND ITS LIMITS

Language models provide real leverage where security work is dominated by unstructured evidence and communication. They can summarize long incidents, extract entities from prose, translate

commands or alerts into an analyst-readable explanation, draft case notes, map natural-language questions to searches, and compare current evidence with prior cases. Retrieval-augmented generation combines parametric generation with retrieved non-parametric context, making organization-specific material available at inference time [10].

These capabilities can accelerate analysts, but performance must be demonstrated on representative tasks. A useful evaluation set distinguishes evidence extraction, grounded summarization, query generation, triage ranking, plan proposal, and action selection. Each requires different metrics and error tolerances. The evaluation should include normal operations, rare but high-impact cases, adversarial content, missing data, stale context, connector errors, and cases where abstention is the correct behavior.

Model self-confidence is not a sufficient control. A generated confidence phrase is not an externally validated probability, and even learned predictive scores can be miscalibrated; calibration measures whether confidence corresponds to observed correctness on a defined population [11]. Production decisions should combine task-specific calibration measured on held-out and monitored data, evidentiary checks, explicit policy constraints, independent validation, and abstention or escalation rules. A low-risk summary may proceed with source links and a warning. A high-impact remediation should require stronger evidence, a separate validator or policy check, and human approval unless the action class has been specifically authorized.

Human oversight is not automatically effective. Reviewers can accept plausible output without checking evidence, especially under time pressure. Approval interfaces should surface the proposed action, target, authority used, evidence and counterevidence, uncertainty, expected blast radius, rollback or compensation plan, and material changes since the previous review. Approval quality should be measured through sampling, reversals, near misses, response time, and reviewer disagreement—not merely by counting approvals.

9. RETRIEVAL IS A SECURITY BOUNDARY

RAG changes the threat model because external and internal content becomes instruction-adjacent context. **Direct prompt injection** enters through a user message. **Indirect prompt injection** is embedded in retrieved email, web content, tickets, documents, code, tool output, or other data the agent reads. Either can attempt to override instructions, disclose secrets, or trigger tools. OWASP identifies prompt injection, excessive agency, sensitive-information exposure, vector and embedding weaknesses, and related risks in LLM applications; its agent guidance also addresses tool abuse, memory poisoning, goal hijacking, and high-impact actions [12][13].

Poisoned retrieval content can bias many future investigations. The content may be malicious at ingestion or later compromised. **Access-filtering errors** can retrieve material the requesting principal or agent is not allowed to see. In shared infrastructure, incorrect metadata, namespaces, cache keys, or post-retrieval filters can cause **cross-tenant leakage**. Authorization should therefore occur before or during retrieval using the requesting identity and purpose; asking the model not to repeat forbidden content is not access control.

Embedding and index drift arise when embedding models, preprocessing, distance metrics, corpora, or index parameters change. The same query may retrieve different context after an update. **Chunking errors** can separate a condition from its exception, a command from its target, or a finding from its date. **Stale context** can present revoked permissions, obsolete runbooks, superseded indicators, or

closed incidents as current. Index lineage should include source version, ingestion time, chunking configuration, embedding model, access labels, deletion state, and rebuild history.

Citation faithfulness is separate from answer plausibility. A response can cite a real document that does not support the associated claim. Systems should retain exact passages, stable source identifiers, and offsets; validators should test whether each consequential claim is entailed by its cited evidence. Analysts need a direct path from prose to source data. Retrieval should fail closed for authorization errors, identify stale or low-quality context, and support abstention when sufficient evidence is unavailable.

Tool output should be treated like retrieved content. A compromised connector, malicious webpage, or poisoned ticket can return instructions disguised as data. The agent runtime should distinguish data from control, constrain parsers, validate structured outputs, isolate untrusted rendering, and prevent tool responses from silently expanding authority. Sandboxing and egress controls reduce the consequences when interpretation fails.

10. FAILURE DOMAINS AND CONTROL OBJECTIVES

Table 1. Failure domains, propagation paths, and principal controls.

Failure domain	Propagation and operational consequence	Principal controls
Telemetry and time	Loss, delay, duplicates, clock error, or parser drift create an incomplete or false sequence.	Coverage/freshness metrics; source authentication; replay; event/ingestion/processing timestamps; lateness and revision policy.
Normalization and state	Identity or asset misresolution changes risk, target, or blast-radius classification.	Versioned mappings; provenance; confidence; effective-time semantics; reconciliation; state-health alarms.
Graph and retrieval	Stale, poisoned, unauthorized, or weakly related context becomes persuasive evidence.	Pre-retrieval authorization; tenant isolation; index lineage; passage-level citations; poisoning tests; stale-data flags.
Model and planning	Unsupported claims, poor calibration, or goal hijacking select the wrong investigation or action.	Task-specific evaluation; evidentiary checks; independent validation; policy constraints; abstention and escalation.
Authority and tools	Overbroad credentials, ambiguous targets, or confused-deputy behavior amplify a bad proposal.	Least privilege; short-lived scoped credentials; separation of duties; canonical targets; strict schemas; approval gates.
Execution and feedback	Retries duplicate side effects; success responses mask failure; compensation is unavailable.	Idempotency keys; rate limits; circuit breakers; independent readback; canaries; rollback or compensating actions.
Human oversight	Automation bias or poor approval context turns nominal review into rubber-stamping.	Evidence-centered review UI; reviewer training; dual control; sampled audits;

		override, reversal, and disagreement metrics.
Multi-agent and supply chain	Privileges compose across agents; compromised connectors, packages, indexes, or policies alter behavior.	Delegation policy; message authenticity; shared-memory isolation; component provenance; signed releases; quarantine and kill switches.

Failures compose. Telemetry loss can produce a weak hypothesis; stale identity state can direct it at the wrong account; an overbroad connector can amplify it; a success response without verification can conceal that the intended containment never occurred. The purpose of a failure-domain analysis is not to assign every error to "the model" or "the platform," but to identify where uncertainty enters, how it propagates, and which independent control can stop it.

Blast radius has several dimensions: number of identities or assets affected, business criticality, duration, reversibility, geographic or tenant scope, data exposure, cost, and external visibility. A low-count action can still be catastrophic if it affects a signing key or safety system. The action classifier should use maintained asset and identity context, not only the model's description of impact.

Feedback validation closes the loop. A connector acknowledgment proves that a request was accepted, not that the intended state exists. Verification should query an independent read path when feasible, account for eventual consistency, distinguish pending from failed, and avoid retries that duplicate non-idempotent actions. If validation remains inconclusive, the system should reduce authority, preserve the ambiguous state, and escalate.

11. SAFETY AND LIFECYCLE ENGINEERING

The safety architecture should be organized across identity, execution, information, resilience, lifecycle, assurance, and response. NIST SP 800-53 provides a broad catalog of security and privacy controls, including access control, audit, least privilege, separation of duties, system integrity, and supply-chain risk management [14]. NIST's Generative AI Profile adds lifecycle considerations for generative systems, while OWASP's agent guidance focuses on the application-specific hazards created by tools, memory, and delegated action [2][13].

Identity, authority, and secrets

Apply least privilege to the agent, orchestrator, connector, model service, user, and workload independently. Use short-lived, audience-restricted credentials; bind authorization to the approved target and action; centralize secret management; prevent prompts and model-visible logs from containing reusable secrets; and rotate or revoke credentials after incidents. Enforce separation of duties for policy changes, connector registration, high-impact approvals, and audit administration. Prevent confused-deputy behavior by carrying the initiating principal, tenant, purpose, and authorization context through each delegation rather than allowing a privileged connector to act solely on the model's assertion.

Multi-agent systems require authority composition, not just per-agent permissions. If one agent retrieves data, another proposes a plan, and a third executes it, the combined workflow must not acquire the union of privileges without an explicit policy decision. Delegation depth, transitive tool access,

shared memory, message authenticity, and responsibility for final approval should be defined. No agent should be able to manufacture approval by asking a peer that has a different trust context.

Input, tool, and execution controls

Validate every tool call against a closed schema: operation, target type, parameter ranges, allowed values, preconditions, idempotency key, and expected side effects. Resolve targets to canonical identifiers and reject ambiguous names. Validate tool output before it updates memory or policy-relevant state. Run code, file conversion, browsing, and untrusted parsers in sandboxes with filesystem, network, process, and time limits. Use destination allowlists, protocol restrictions, data-loss prevention, and egress inspection where appropriate.

Set rate limits, concurrency limits, token and monetary budgets, action quotas, and per-incident ceilings. Circuit breakers should stop repeated or anomalous calls. Kill switches should revoke authority at the policy or credential layer rather than depending on the agent to obey a stop message. Dry runs should show intended changes and required authority. Canaries should expose a change to a limited scope. Rollback is preferred when the prior state can be restored; compensating actions are required when a side effect is irreversible or rollback is unsafe.

Versioning, testing, and change management

Version models, providers, inference settings, prompts, tool schemas, policies, normalization rules, retrieval corpora, chunking strategies, embedding models, indexes, and evaluation datasets. A release manifest should identify the compatible set. Changes to any of these components can alter behavior even when application code is unchanged.

Testing should include unit and policy tests, connector contract tests, deterministic replay where possible, regression suites, adversarial evaluation, red teaming, and staged production canaries. MITRE ATLAS offers a living knowledge base of tactics and techniques involving attacks on or misuse of AI-enabled systems, while MITRE ATT&CK provides a common language for adversary goals and techniques in enterprise environments [15][16]. These resources can inform scenarios, but control tests should also cover organization-specific assets, permissions, workflows, and failure histories.

Calibration should be measured per task, action class, environment, and material model revision. Monitor retrieval quality, citation support, abstention rate, false positive and false negative outcomes, override rate, policy denials, tool errors, verification failures, and the distribution of actions and targets. Drift monitoring should detect changes in telemetry schemas, incident mix, embedding behavior, model output, connector responses, and reviewer behavior. Thresholds should trigger reduced authority or rollback to a validated configuration.

Privacy, retention, tenancy, and supply chain

Minimize prompts and retained traces to the data needed for the task. Define retention separately for raw telemetry, prompts, model outputs, embeddings, case records, and audit evidence. Support deletion obligations without silently leaving content in derived indexes. Enforce tenant isolation at storage, retrieval, cache, model-routing, connector, and observability layers. Test access filters before and after index rebuilds.

Threat models should include compromise of connectors, model endpoints, packages, prompt repositories, policy bundles, evaluation data, and update channels. Record component provenance and protect build and deployment workflows. NIST's Secure Software Development Framework emphasizes integrating secure practices and provenance-oriented evidence into the software lifecycle [17].

Human factors and incident response

Automation bias can convert an advisory system into de facto autonomy when reviewers routinely accept its recommendation. Interface design, staffing, incentives, and review time are therefore part of the control plane. Assign accountability for approving actions, maintaining evidence quality, tuning thresholds, and accepting residual risk. Periodically test whether reviewers detect unsupported claims and whether escalation routes work under load.

Prepare an agent-specific incident response plan. It should cover credential revocation, connector isolation, model or prompt rollback, index quarantine, memory invalidation, preservation of traces, notification, and restoration of a known-good configuration. Forensic reconstruction requires immutable or tamper-evident records of prompts, context, policy decisions, calls, outputs, approvals, and state transitions, with synchronized time and access controls. NIST SP 800-61 Rev. 3 places incident response within broader cybersecurity risk management and emphasizes preparation, detection, response, recovery, and continuous improvement [18].

12. ECONOMIC AND ORGANIZATIONAL INCENTIVES

"Agentic" terminology spreads through multiple, nonuniform incentives. Vendors may use it to communicate a real shift from static assistance to model-selected actions, to align with buyer vocabulary, to differentiate in a crowded category, or to make an interface legible in a short demonstration. Some claims are precise; others compress distinctions among capability, autonomy, and architecture. Without a defined corpus of products and claims, it is not responsible to infer a uniform motive or prevalence.

Buyers also vary. Security leaders may seek faster triage, staffing leverage, tool consolidation, improved consistency, or evidence of modernization. Procurement and executive audiences often need short categories, while operators need details about state, authority, failure, and recovery. The resulting mismatch can reward a broad label even when the implementation is carefully bounded. Conversely, an architecture team can dismiss useful model-mediated planning merely because the underlying tools are familiar.

Analysts, boards, investors, journalists, regulators, and practitioners operate under different information and time constraints. Public demonstrations reveal interface behavior more readily than backpressure, tenant isolation, calibration, or compensating actions. Technical diligence should correct this observability imbalance without attributing bad faith. Requests for architecture diagrams, action inventories, evaluation results, incident procedures, and control evidence convert a branding debate into verifiable questions.

The measured position is therefore evidence-conscious: treat "agentic" as a hypothesis about behavior. Ask which goals persist, which actions are selected dynamically, what authority is delegated, what feedback changes the plan, what is learned or merely retrieved, and which evidence supports safety claims. Market language is context; execution traces and control artifacts are evidence.

13. AGENTICITY AND AUTONOMY AS A MATURITY PROFILE

A single threshold for "genuine agenticity" hides important variation. The maturity matrix below uses four illustrative operating profiles across nine dimensions. It is not a certification scale, and higher is not always better. A Level 1 system may be appropriate for privileged infrastructure; a Level 3 system may be justified for reversible, low-impact actions in a well-instrumented environment. Systems can occupy different levels by action class.

Table 2. Agenticity and autonomy maturity matrix.

Dimension	Level 0 — Assistive	Level 1 — Procedural	Level 2 — Adaptive agent	Level 3 — Delegated autonomous
Goal formation	Human supplies each task.	Human goal mapped to a fixed procedure.	Delegated goal is decomposed and revised within policy.	System refines subgoals within a standing mandate.
Planning horizon	One response or query.	Short workflow with known end state.	Multi-step episode; pauses and resumes.	Long-running, cross-session pursuit with renewal rules.
Dynamic action selection	No action or human-selected tool.	Predefined branch or playbook.	Model selects and sequences from a governed catalog.	Composes actions through governed interfaces within a mandate.
Environmental feedback	Displayed to a human.	Feedback gates the next fixed step.	Observed effects update the plan.	Continuous closed loop with degraded modes.
Adaptation	No autonomous revision.	Chooses among predefined branches.	Revises hypothesis and strategy during the task.	Reprioritizes across tasks under policy and budgets.
Memory	Transient prompt context.	Workflow and case state.	Persistent case memory with provenance and correction.	Cross-session/cross-case memory with retention and isolation controls.
Authority	Advice or read-only access.	Preapproved low-risk actions or per-step approval.	Bounded reversible authority by action class.	Broader preauthorization with limits, canaries, and revocation.
Learning	None at runtime.	Human updates rules or prompts.	Retrieval and state update; model changes are offline.	Controlled skill/policy updates validated before wider use.
Human oversight	Human reviews every output.	Human approves specified steps.	Risk-based approvals plus active monitoring.	Exception-based and retrospective review with accountable owners.

The dimensions should be assessed separately:

- **Goal formation** asks whether goals are supplied, decomposed, revised, or originated within a mandate.
- **Planning horizon** asks how many steps and how much elapsed time the system can manage before renewal.
- **Dynamic action selection** asks whether the sequence is fixed, selected from a bounded catalog, or composed more openly.
- **Environmental feedback** asks whether observed effects can change the plan.
- **Adaptation** asks whether the system revises a hypothesis or strategy within the task.
- **Memory** asks what persists, for how long, under what provenance and correction rules.
- **Authority** asks which side effects can occur without contemporaneous approval.
- **Learning** distinguishes retrieval or state update from parameter, policy, or skill change.
- **Human oversight** asks whether review is continuous, exception-based, retrospective, or absent—and whether it is effective.

No weighted total can substitute for the profile. Authority and blast radius may dominate safety even when all other dimensions are low. Persistent memory can create privacy and poisoning risk without increasing action authority. Dynamic planning can improve investigations while all remediation remains approval-gated. The system description should publish the profile by deployment mode and name the evidence used to assign it.

14. PRACTICAL ARCHITECTURE EVALUATION CHECKLIST

An evaluator should be able to answer the following questions with diagrams, configurations, tests, and traces—not only narrative assurances.

Evidence and time

- Which telemetry sources are authoritative, and how are loss, delay, duplication, parser failure, schema drift, and source authenticity measured?
- Which decisions use event time, ingestion time, or processing time? What are the partition keys, lateness policy, revision semantics, and clock assumptions?
- Can a claim be traced through normalization, entity resolution, graph or retrieval context, model output, policy decision, and action?

State and retrieval

- What does "memory" mean in this deployment? Which stores are durable, tenant-isolated, correctable, retained, and replayable?
- How are graph edges and retrieved chunks versioned with source, confidence, valid time, access labels, and deletion state?
- How are direct and indirect prompt injection, poisoning, stale content, chunking errors, embedding drift, cross-tenant leakage, and citation faithfulness tested?

Agent behavior and authority

- What goals can the system form or revise, how long can they persist, and which actions are dynamically selected?

- Which model outputs materially affect action type, target, timing, or parameters? Where is the probabilistic-to-deterministic boundary?
- What is the effective authority after composing users, agents, orchestrators, connectors, credentials, and peer agents?
- Which actions require approval, dry run, canary, blast-radius analysis, rollback, or compensating action?

Assurance and operations

- Are model, prompt, policy, tool, index, and schema versions recorded for every consequential trace?
- How are task-specific calibration, evidence sufficiency, independent validation, abstention, and escalation evaluated?
- What budgets, rate limits, circuit breakers, kill switches, egress controls, and degraded modes contain failure?
- How is outcome verification independent of the action request, and how are ambiguous or eventually consistent results handled?
- Can the organization revoke authority, quarantine poisoned context, reconstruct an incident, restore a known-good release, and measure approval quality?

15. CONCLUSION

Agentic behavior can exist inside a bounded architecture. A system that observes, plans, selects tools, incorporates feedback, and revises its approach is meaningfully agentic even when its permissions and action catalog are constrained. Those constraints are not evidence against agenticity; they are often the conditions that make it deployable.

The architectural mistake is to let one word imply more than the system delivers. Agenticity does not establish broad autonomy, general capability, open-endedness, persistent learning, architectural novelty, or safety. Those properties must be evaluated independently. In security operations, the decisive facts remain execution semantics, state, authority, provenance, temporal behavior, feedback validation, and failure containment.

Language models can materially improve retrieval, summarization, explanation, investigation planning, and analyst throughput. When their outputs influence action selection, however, the end-to-end system remains probabilistic even if an explicit policy engine performs the final call. Safe deployment therefore requires an evidence-preserving pipeline, calibrated and independently checked decisions, least-privilege execution, bounded blast radius, verified outcomes, effective human oversight, and a lifecycle capable of detecting drift and reconstructing incidents.

The practical question is not whether a platform deserves the "agentic" label. It is whether its claimed degree of agenticity and autonomy is supported by traces, tests, controls, and operating evidence—and whether the organization can recover when any layer is wrong.

REFERENCES

[1] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0).* NIST AI 100-1, 2023. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf

[2] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile.* NIST AI 600-1, 2024. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

[3] M. Roesch. "Snort—Lightweight Intrusion Detection for Networks." 13th USENIX Systems Administration Conference (LISA '99), 1999. https://www.usenix.org/legacyurl/usenix-technical-program-abstract-13th-systems-administration-conference-lisa-99-12

[4] K. Kent and M. Souppaya. *Guide to Computer Security Log Management.* NIST SP 800-92, 2006. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf

[5] S. Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models." ICLR, 2023. https://arxiv.org/abs/2210.03629

[6] Apache Software Foundation. "Apache Kafka Design." https://kafka.apache.org/41/design/design/

[7] Apache Software Foundation. "Apache Beam Programming Guide." https://beam.apache.org/documentation/programming-guide/

[8] T. Akidau et al. "The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing." *Proceedings of the VLDB Endowment,* 2015. https://doi.org/10.14778/2824032.2824076

[9] M. Fowler. "Event Sourcing." 2005. https://martinfowler.com/eaaDev/EventSourcing.html

[10] P. Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." NeurIPS, 2020. https://arxiv.org/abs/2005.11401

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. "On Calibration of Modern Neural Networks." ICML, 2017. https://proceedings.mlr.press/v70/guo17a.html

[12] OWASP Foundation. *OWASP Top 10 for LLM Applications 2025.* 2024. https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf

[13] OWASP Foundation. "AI Agent Security Cheat Sheet." https://cheatsheetseries.owasp.org/cheatsheets/AI_Agent_Security_Cheat_Sheet.html

[14] National Institute of Standards and Technology. *Security and Privacy Controls for Information Systems and Organizations.* NIST SP 800-53 Rev. 5, 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

[15] MITRE. "Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS)." https://atlas.mitre.org/

[16] MITRE. "Enterprise Tactics—MITRE ATT&CK." https://attack.mitre.org/tactics/

[17] National Institute of Standards and Technology. *Secure Software Development Framework (SSDF) Version 1.1.* NIST SP 800-218, 2022. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf

[18] National Institute of Standards and Technology. *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile.* NIST SP 800-61 Rev. 3, 2025. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r3.pdf