Agent Governance & Compliance for the Enterprise

OWASP, NIST, ISO 42001, and the architectural controls that actually matter

The Agile Monkeys

March 2026

The Risk Landscape

In December 2025, the OWASP Foundation released the Top 10 for Agentic Applications — a risk taxonomy developed by over 100 industry experts, researchers, and practitioners. It was overdue. By the time it published, 80% of organizations were already reporting agents acting outside expected behavior, yet only 21% had any visibility into what their agents could access (Help Net Security, 2026).

The OWASP Agentic Top 10 identifies ten risk categories specific to autonomous AI agents, from Agent Goal Hijack (ASI01) — representing total loss of control — through Tool Misuse, Identity and Privilege Abuse, to Rogue Agents (ASI10) — compromised or misaligned agents diverging from intended behavior. Real-world incidents have already been mapped to multiple categories, including prompt injection attacks through MCP servers, goal hijacking via manipulated tool outputs, and identity abuse through over-permissioned service accounts.

What makes agent risk fundamentally different from traditional application security is the combination of autonomy, tool access, and nondeterminism. Agents chain API calls dynamically. They generate and execute code. They create sub-processes. And they do all of this with behavior that varies from run to run. Traditional security models — static rules, pattern matching, perimeter defenses — were not designed for actors like these.

Shadow AI compounds the problem. The critical distinction: shadow IT accesses data; shadow AI takes actions. An unsanctioned agent with access to corporate email doesn't just read messages — it can send them, forward them, summarize them to external services, and trigger downstream workflows. The blast radius of an ungoverned agent is categorically larger than an ungoverned SaaS subscription.

Only 19% of organizations monitor agents in production (IBM, 2026). The gap between deployment velocity and governance maturity is where incidents happen.

What's Genuinely New About Agent Security

If your organization already runs a mature DevSecOps practice, much of agent governance applies established patterns to a new actor type: VPC isolation, API gateways, short-lived credentials, mutual TLS, audit logging, SIEM integration. These are necessary, non-negotiable, and

familiar.

What is genuinely new — what has no microservices equivalent — is where the real risk lies:

Semantic Data Transformation

Traditional DLP uses pattern matching. It detects credit card numbers (16-digit sequences), Social Security numbers (XXX-XX-XXXX), and keywords from classification lists. This works when data retains its format through a pipeline.

Agents break this assumption. When an agent summarizes a confidential report, paraphrases a customer complaint, or translates internal communications, the sensitive information is preserved in meaning but transformed in form. The credit card number becomes "the customer's payment method on file." The patient diagnosis becomes "the condition discussed in the consultation." Regex sees nothing. The data has leaked.

Semantic DLP — systems that analyze meaning rather than patterns — is the required replacement. Microsoft Purview now extends DLP and Information Protection policies to autonomous agents, allowing them to inherit the same protections as human users: blocking access to labeled files, preventing external transmission of classified content. Proofpoint's framework addresses four threat vectors: protecting agents from targeted attacks, controlling data loss by both people and agents, governing agent actions, and using agents to automate security itself.

The market is moving toward what some vendors describe as "Unified Agentic Defense Platforms" (UADP) — integrating Data Security Posture Management, adaptive DLP, and AI Runtime Security into a single layer. This is the successor to pattern-based DLP for the agentic era.

Layered Output Enforcement

Raw LLM output is nondeterministic: the same prompt may produce different outputs on consecutive runs. You cannot trust a system prompt — no matter how carefully crafted — to enforce policy consistently. "Never access data outside your scope" is a guideline, not a control.

The good news is that modern agent SDKs provide multiple layers to enforce deterministic behavior where it matters. The architectural response combines three layers, each adding a stronger guarantee, and aligns well with SOC 2 compliance requirements (PolicyLayer, 2025):

Layer 1: Structured Output Enforcement. Agents can be constrained to produce validated JSON outputs conforming to strict schemas. The LLM interprets the user request and generates a structured intent — a machine-readable description of what it wants to do. "Process a refund of

$500 for order #1234 to the customer's original payment method." Schema enforcement guarantees the intent is well-formed before any downstream processing sees it.

Layer 2: Business Logic Hooks. Deterministic pre- and post-execution hooks validate the intent against business rules. These are regular code, not LLM calls. If validation fails, the agent can be instructed to retry with the error context, correcting its output until it passes. This turns nondeterminism into a bounded, self-correcting loop.

Layer 3: Deterministic Policy Validation. A non-LLM policy engine — a rules engine, OPA, or deterministic code path — makes the final permit/deny decision against organizational policy. Refund amount within policy limits? Customer identity verified? Order eligible for refund? No fraud flags?

Only if Layer 3 approves does the action execute. The LLM never directly triggers consequential operations. This layered validation — schema enforcement, business logic hooks, and policy gates — is what makes agent behavior auditable and compliant. For SOC 2 processing integrity, the pattern enables intent fingerprinting — cryptographic hashing of the validated intent — creating an immutable record of what was authorized and executed.

Scoped Tool Access for Sub-Agents

When an agent delegates tasks to sub-agents, how should identity and permissions flow? The answer from production agent systems is clearer than the theoretical debate suggests: each sub-agent should be provisioned at design time with exactly the specific tools it needs — no more, no less. This is the same principle as microservice architecture. A payment processing sub-agent gets access to the payment API and nothing else. A fraud-checking sub-agent gets read access to transaction history but no write capabilities. A summarization sub-agent gets read access to the documents it needs to summarize but no network access at all.

This design-time scoping is preferable to runtime delegation schemes where a parent agent dynamically passes a subset of its permissions to children. Runtime delegation creates three problems: it makes it hard to audit what any given sub-agent can actually do at any given moment; it creates incentive to grant broad permissions to the orchestrator "just in case" a sub-agent might need them; and it complicates revocation, because permissions live ephemerally in the delegation chain rather than in the identity system.

That said, the identity and lifecycle challenges remain real:

Ephemeral sub-agent identities. An orchestrator agent may create dozens of short-lived worker agents per hour. Each worker needs its own identity — not for the sake of isolation alone, but so its actions are distinctly auditable. This requires provisioning and deprovisioning infrastructure that operates at machine speed, which is where patterns like SPIFFE workload identity federation become essential.

Delegation chain accountability. When a personal agent delegates to a service agent that delegates to a data access agent, and the data access agent exfiltrates data, the audit trail must preserve the full delegation chain — not just the immediate actor. Correlation IDs that flow through every call in the chain are the standard mechanism.

Identity proliferation. ISACA warns that agents creating sub-agents with untracked identities is already creating gaps in provisioning, ownership, and deprovisioning. Non-human identities already outnumber humans 50:1 in average enterprises (Silverfort, 2025). Agents will accelerate this ratio significantly, which is why treating agent identities as first-class principals in the existing IAM infrastructure (rather than a parallel registry) is essential.

NIST launched its AI Agent Standards Initiative in February 2026, with agent identity and authorization as a top-tier focus area. The open question: whether existing identity standards (OAuth, SAML, SPIFFE) are sufficient with extensions, or whether fundamentally new agent identity protocols are needed.

Agents as First-Class Principals

There is a temptation to manage agent identities through a separate system — a dedicated "agent registry" disconnected from the organization's IAM. This is the wrong approach. Cloud infrastructure has already solved this problem: Google Cloud IAM treats service accounts as members of the same policy system as human users. AWS IRSA lets Kubernetes pods assume IAM roles through OIDC federation, using the same policy language and audit trails as human users. SPIFFE (the CNCF standard for workload identity) provides a universal identity framework where non-human workloads get the same lifecycle management as human identities — provisioning, authentication, authorization, rotation, and deprovisioning.

The principle: agents should be first-class principals in the organizational identity system, not second-class entities governed by a parallel system. Same policy engine, same audit trails, same governance tools. This is not theoretical — it's how every major cloud provider manages non-human identities at scale.

But agents are not users. The distinction matters:

Property	Shared with users	Different from users
Identity	Unique, auditable, lifecycle-managed	No legal standing, no privacy rights
Permissions	Same policy engine, same least-privilege	Task-scoped, automatically revoked
Audit trail	Same format, same retention	Higher volume (50:1 NHI ratio)
Accountability	Traceable delegation chain	No personal judgment or liability
Memory	Persistent context (like personal KB)	Organization-owned, auto-decommissioned

The framing is agents-as-principals, not agents-as-users. A principal is any entity that can be authenticated, authorized, and audited — without implying the legal, regulatory, and social properties that "user" carries. This avoids GDPR absurdities (agents don't have data subject rights) while preserving the governance benefits of a unified identity model.

Agent-specific extensions beyond traditional workload identity are required because agents differ from conventional workloads in four ways: non-deterministic behavior (traditional workloads are predictable; agents may take unexpected actions within their permissions), memory persistence (traditional workloads are stateless; agents accumulate knowledge that must be governed), dynamic tool selection (traditional workloads call fixed APIs; agents choose dynamically), and sub-agent spawning (traditional workloads don't create new workloads autonomously). These differences mean existing workload identity infrastructure is necessary but not sufficient — the identity model works, but the governance layer needs behavioral monitoring, memory governance, and tool-use authorization extensions.

Access Control Architecture

Choosing an Access Control Model

How to implement access control for agents — whether through RBAC, PBAC, ABAC, ReBAC, task-scoped JIT permissions, or some combination — is an engineering decision that depends on the use case, the existing infrastructure, and the nature of the agent's work. There is no universally correct answer. What matters is that agents, like any other service, are treated as principals in the organization's existing permission enforcement infrastructure.

That said, the tradeoffs between models are worth understanding:

RBAC assigns permissions to roles, and roles to users or services. It works well when permissions are stable and the number of distinct roles is manageable. Agents with narrowly scoped, well-defined tasks may fit this model cleanly — a "refund-processor" agent with a single role is no different from a "refund-processor" microservice.

PBAC (Policy-Based Access Control) replaces static role assignments with a centralized policy engine that evaluates real-time signals: what is the agent trying to do, on whose behalf, with what context, at what risk level? The policy engine makes a permit/deny decision based on composable rules. PBAC is useful when agent behavior varies significantly by context, when real-time risk signals (anomaly scores, volume thresholds, time-of-day rules) should influence decisions, or when policies need to change without redeploying agents. Obsidian Security recommends PBAC for agents operating in high-variance contexts specifically because policies can incorporate these real-time signals.

ABAC layers attribute evaluation on top of either RBAC or PBAC, useful when access depends on properties of the resource being accessed (sensitivity labels, data classification, purpose tags) rather than just the requester's identity.

ReBAC (Relationship-Based Access Control) models permissions as paths through a graph of relationships — particularly well-suited to knowledge bases where access follows organizational relationships (see Pillar 3).

The pragmatic pattern most teams converge on: use the organization's existing access control infrastructure (whatever it is) as the foundation, and add policy-based layers only where the existing model genuinely fails for agent-specific requirements. Don't rebuild your access control stack to accommodate agents; extend it.

Short-Lived, Task-Scoped Credentials

Long-lived API tokens are among the most common identity compromise vectors for agents. The recommended pattern:

- Short token lifespans (e.g., 1-2 hours) with regular rotation
- Task-scoped authorization: Just-in-Time (JIT) permissions granted for the duration of a specific task, automatically revoked on completion
- On-Behalf-Of (OBO) delegation flows: agents act within the delegating user's permission boundary, never exceeding it
- Mutual TLS for all agent-to-service communication
- Workload identity federation (no stored credentials in agent environments)

Progressive Autonomy

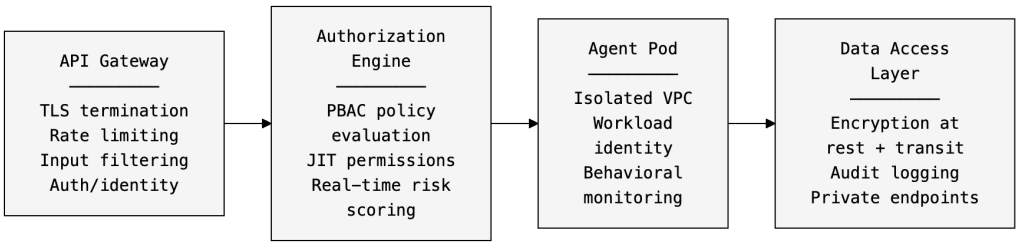
The Cloud Security Alliance's Agentic Trust Framework introduces a pattern the industry is converging on: organizations "progress agents from initial deployment through increasing levels of autonomy, with clear criteria and controls at each stage" (CSA, 2026).

A new agent starts with human-in-the-loop approval for every action. As behavioral baselines are established and no anomalies are detected, low-risk actions become autonomous while high-risk actions retain human gates. The autonomy level is continuously recalibrated based on behavioral monitoring — a single anomaly can trigger regression to full oversight.

This mirrors how organizations onboard human employees: new hires have restricted access, earn broader permissions over time, and can have access revoked immediately if something goes wrong.

Reference Deployment Architecture

A common reference architecture for autonomous agent services uses four layers:



Each layer enforces defense-in-depth. The API Gateway handles perimeter security. The Authorization Engine makes dynamic access decisions. The Agent Pod runs in network-isolated infrastructure with its own workload identity. The Data Access Layer encrypts everything and logs all access.

Critically, this architecture is fail-closed: any infrastructure failure causes agents to lose access, not gain it.

Compliance-to-Control Mapping

HIPAA

Requirement	Agent-Specific Control
Protected Health Information (PHI) safeguards	Auto-redaction of PHI in agent prompts and responses before processing

Business Associate Agreements	BAAs required with all LLM providers and agent hosting services
Unique user identification	Unique agent IDs with audit logs capturing all PHI access
Encryption	ePHI encrypted at rest and in transit, including in agent memory and context
Audit controls	Trace-level logging of all agent reasoning chains involving PHI
Automatic logoff	Agent sessions expire; no persistent PHI in agent memory after task completion
Retention	6-year log retention for all agent activity involving PHI

Key architectural pattern: Intelligent routing directs healthcare data only to HIPAA-compliant models and infrastructure. Non-compliant models never see PHI.

PCI-DSS

Requirement	Agent-Specific Control
Cardholder data protection	Prevent storage or disclosure of PANs in agent memory, outputs, or logs
Network segmentation	Payment-processing agents isolated in dedicated network segments
Data tokenization	Cardholder data tokenized before agent access; agents work with tokens, never raw PANs
Access control	Scoped credentials per agent; quarterly key rotation
Monitoring	Real-time pattern detection for card data in agent outputs

SOC 2

Trust Criteria	Agent-Specific Control
Security	Scoped API credentials per agent; agent-to-policy mapping reviews; continuous monitoring
Availability	SLA chains through agent orchestration; fail-closed design
Processing Integrity	Two-gate enforcement (LLM intent → deterministic validator); intent fingerprinting via cryptographic hashing

Confidentiality	TLS 1.3; private keys never transmitted to policy services; memory isolation between agents
Privacy	Pseudonymized transaction data; data minimization in agent prompts

Audit log requirements: Every log entry must capture event ID, timestamp, agent identifier, decision outcome, policy version applied, intent fingerprint, real-time cost counters, and source IP.

GDPR

Requirement	Agent-Specific Control
Legal basis	Documented legal basis per agent processing activity
Data minimization	Agents receive only the data necessary for the current task, not full records
Purpose limitation	Each agent's scope is documented and enforced; no repurposing data across agents
DPIAs	Data Protection Impact Assessments required for high-risk agent processing
Breach notification	72-hour notification capability; automated detection of agent-mediated breaches
Right to erasure	Tombstone events in event streams; summary regeneration excluding erased data; cryptographic erasure via per-user key destruction
Data residency	Intelligent routing: EU data processed only on GDPR-compliant infrastructure

The erasure challenge: LLMs complicate the right to be forgotten because training data cannot be selectively removed, and agent memory may contain summarized user data at multiple levels. The combination of tombstone events, cryptographic erasure, and summary regeneration (discussed in the knowledge hierarchy deep dive) represents the current best practice, but this remains an evolving area.

ISO 42001

ISO/IEC 42001 is the world's first AI management system standard, with 38 controls across 9 objectives. It is rapidly moving from "nice to have" to procurement-required, with a growing number of enterprise RFPs in the EU, US, and UK specifically requiring or scoring ISO 42001 status (Deloitte, 2025).

The controls most relevant to agent governance: risk and impact assessments specific to agent autonomy, training data governance and

provenance, model lifecycle management (versioning, deployment, retirement), bias monitoring and fairness documentation, and transparency requirements for AI decision-making.

The Dual Certification Baseline

Neither SOC 2 nor ISO 42001 alone is sufficient:

- SOC 2 covers foundational cybersecurity (access controls, encryption, availability) but lacks controls for model governance, training data, or agent autonomy
- ISO 42001 covers AI-specific risks but doesn't address foundational security infrastructure

The emerging enterprise consensus: pursue both. SOC 2 provides the security foundation; ISO 42001 provides the AI-specific governance layer. Together, they cover the full spectrum of agent risk.

Agent Testing and Validation

Governance without verification is policy fiction. The industry's biggest gap is not in defining what agents should do, but in testing that they actually do it.

Red Teaming Against OWASP Agentic Top 10

Systematic adversarial testing should map to the OWASP risk taxonomy:

- ASI01 (Agent Goal Hijack): Attempt to redirect agent behavior through manipulated inputs, tool outputs, and context injection
- ASI03 (Identity and Privilege Abuse): Test whether agents can escalate privileges, access unauthorized resources, or impersonate other agents
- ASI05 (Unexpected Code Execution): Validate sandboxing and containment of agent-generated code, prevent unauthorized code execution
- ASI10 (Rogue Agents): Test detection of agents that deviate from expected behavior; validate kill switches, circuit breakers, and isolation mechanisms

Red teaming should be continuous, not periodic — model updates can change behavior in ways that are difficult to anticipate.

Behavioral Baselining and Anomaly Detection

Establish expected behavioral patterns for each agent type: typical API call frequencies, data access patterns, response latencies, tool usage sequences. ML-based anomaly detection flags deviations in real time.

Target monitoring metrics from Obsidian Security:

- Mean Time to Detect (MTTD): < 5 minutes
- Mean Time to Respond (MTTR): < 15 minutes
- False Positive Rate: < 5%
- Agent Coverage: 100% (no unmonitored agents)

Governance Verification Automation

Automated tests that confirm policy enforcement:

- Can an agent access resources outside its declared scope? (Should fail)
- Does JIT permission revocation work after task completion? (Should succeed)
- Does two-gate enforcement block unauthorized intents? (Should block)
- Do delegation chains preserve access boundaries? (Should enforce ceiling)

These tests should run on every deployment and after every model update.

Model Update Regression

Model updates — whether version upgrades, fine-tuning, or provider changes — can alter agent behavior in ways that break governance assumptions. Every model update should trigger:

- Full behavioral test suite re-execution
- Comparison of output distributions against baseline
- Red team re-engagement for high-risk agent types
- Progressive rollout with increased monitoring

What's Still Debated

Not everything is settled. The industry is actively debating several foundational questions:

Agent identity standards. NIST is investigating whether existing identity standards are sufficient for agent identity or whether purpose-built protocols are needed. Standards like OAuth were designed for human-delegated access with session-based lifecycles, not for autonomous actors that operate continuously and spawn sub-processes. The answer will

shape agent IAM architecture for the next decade.

Authorization model. PBAC, enhanced ABAC, and dynamic RBAC all have advocates. The optimal model likely depends on organizational complexity, existing infrastructure, and agent autonomy levels. There is no universal answer yet.

Recursive sub-agent governance. When an agent creates sub-agents autonomously, how deep can the chain go? Should sub-agents inherit the parent's identity or get independent credentials? Should there be a maximum delegation depth? No consensus exists.

GDPR erasure with persistent memory. If an agent has learned from user interactions and that knowledge has been summarized, shared across levels, and incorporated into model fine-tuning, what does "erasure" mean? The legal and technical communities are not yet aligned.

Global taxonomy harmonization. IMDA (Singapore), NIST (US), the EU AI Act, UC Berkeley, and multiple vendor frameworks all use different risk categorizations. A CISO operating across jurisdictions must map between incompatible taxonomies. Harmonization efforts are underway but remain in early stages.

Governance vs. performance. Every governance layer adds latency: policy evaluation, two-gate validation, audit logging, encryption. How much governance overhead is acceptable before agents become too slow to be useful? The answer varies by use case, and the industry hasn't established benchmarks.

Architectural vs. infrastructural enforcement. This framework advocates building governance into the agent architecture — identity, access control, and policy enforcement as structural properties of the system. A complementary approach treats governance as middleware: a proxy layer between all data sources and all AI consumers that enforces consent, redaction, and data-use preferences (such as "Do Not Train") at the boundary, regardless of what the agent architecture looks like internally. Both approaches have merit, and production systems will likely use both — architectural controls for agent-specific governance, infrastructure middleware for data-level compliance.

"Do Not Train" as a data right. Beyond GDPR erasure, a growing pattern gives users the ability to exclude their data from AI training pipelines entirely. When an agent's knowledge base is built from organizational data, honoring "Do Not Train" preferences requires enforcement at the data pipeline layer — excluding opted-out data before it enters the event stream, not after it's been summarized into knowledge nodes. This is an emerging area with vendors like Transcend operationalizing it at scale.

These are not reasons to delay governance. They are reasons to build governance architectures that are modular and adaptable — capable of absorbing new standards and requirements as the landscape matures.

References

- OWASP. "Top 10 for Agentic Applications." December 2025. owasp.org
- NIST. "AI Agent Standards Initiative." February 2026. nist.gov
- Cloud Security Alliance. "Agentic Trust Framework." 2026. cloudsecurityalliance.org
- Microsoft. "Architecting Trust: NIST-Based Security Governance Framework for AI Agents." January 2026. techcommunity.microsoft.com
- Obsidian Security. "Security for AI Agents." 2025. obsidiansecurity.com
- ISACA. "The Growing Challenge of Auditing Agentic AI." 2025. isaca.org
- Help Net Security. "Enterprise AI Agent Security 2026." 2026. helpnetsecurity.com
- Proofpoint. "Proofpoint Secures Collaboration and Data in the Agentic Workspace." 2025. proofpoint.com
- Lakera. "Data Loss Prevention: A Complete Guide for the GenAI Era." 2025. lakera.ai
- Requesty. "Security Compliance Checklist for LLM Gateways." 2025. requesty.ai
- PolicyLayer. "SOC2 Compliance for AI Agents." 2025. policylayer.com
- Silverfort. "Insecurity in the Shadows: Why Non-Human Identities Are a Cybersecurity Priority." 2025. silverfort.com
- Deloitte. "ISO 42001 Standard: AI Governance & Risk Management." 2025. deloitte.com
- IBM. "The Year Companies Stop Building AI Agents and Start Running Them." 2026. ibm.com
- SPIFFE. "Secure Production Identity Framework for Everyone." CNCF, 2024. spiffe.io
- Google Cloud. "Best Practices for Managing Service Accounts." 2025. cloud.google.com
- AWS. "IAM Roles for Service Accounts (IRSA)." 2025. docs.aws.amazon.com