AKEYLESS

AI AGENT IDENTITY SECURITY

The 2026 Deployment Guide

AI Agent Identity Security: The 2026 Deployment Guide

Introduction:

The Year AI Agents Began to Act

Across engineering, operations, and security teams, a quiet shift has been underway. Systems that once assisted humans now make modifications on their own: pulling data from production stores, updating records, triggering workflows, and interacting with external services. These actions rarely arrive as large programs. These changes rarely appear as major initiatives. Instead, they emerge incrementally, folded into existing processes and justified as efficiency gains rather than architectural change.

When software can decide and act inside live environments, the consequences of its decisions matter as much as the quality of its outputs. Trust and authority now matter as much as capability. The question stops being whether an agent produces the right answer and becomes whether it should have been allowed to act at all.

Analyst firm Gartner predicts that 2026 will usher in "a new wave of turbulence [in] the form of AI agent sprawl, and internal development projects aiming at transforming custom-built AI chatbots into AI agents."[1] The chaos has already started. One global survey found that 80% of organizations using AI agents admitted their agents took unintended actions, including unauthorized system access and data sharing.[2]

We have moved beyond theory into real consequences. Recent industry data revealed that one in five organizations has experienced at least one AI agent–related security incident.[3] As autonomous behavior expands, incidents tied to unclear authority and mis-scoped access will only grow both in frequency and impact.

How AI agents are identified, authorized, and governed when they act inside real enterprise systems has become critical to get right.

Here you'll find a clear, practical path from principle to deployment.

Chapter 1:

Why AI Agent Deployment Is an Identity Problem

From Automation to Agency

In earlier eras of software automation, actions were predictable. Execution paths were predefined, permissions were static, and systems authenticated once before running continuously. The model was imperfect and often over-privileged, but it was legible. Teams could generally explain what a system was allowed to do, even when they disagreed with how much access it had.

AI agents do not fit that mold. Their behavior is shaped by context rather than configuration alone. They select tools dynamically, move across environments in a single flow, and adapt based on inputs that are not fully predictable. Autonomous AI agents are now live in production environments, orchestrating workflows, accessing sensitive data, and making decisions that impact business outcomes in ways traditional security controls were never designed to address.

At that point, the issue ceases to be how a system authenticates and becomes a question of what authority the system is exercising. When a system can decide and act, it functions as an actor. Actors require identity, and identity brings access control and accountability into scope.

Where Agent Deployments Go Wrong

Breakdowns are not usually spectacular or abrupt. Instead, they emerge through a cascade of reasonable design decisions made under delivery pressure, often before agents are treated as long-lived production actors.

- Teams reuse existing credentials because they are already approved
- They widen permissions to avoid blocking workflows
- Secrets pass through agent execution paths and connected tools without clear ownership
- Access persists because cleanup is deferred

Each choice seems contained, but together they form a pattern that becomes hard to explain and harder to defend.

This dynamic mirrors broader identity trends. Across most enterprises, non-human identities already vastly outnumber human users. Service accounts, automation, and agent identities now exceed human identities by an estimated 144 to 1.[4] Many of these identities are created quickly during experimentation or vibe-coded development and persist far longer than intended, multiplying the number of access paths that must be managed and increasing the likelihood that some will be misused or forgotten.

As agent counts increase, uncertainty compounds. Security teams often cannot determine which agent performed an action, under what authority, or on whose behalf. Monitoring may confirm that something happened, but it rarely explains whether it should have happened.

Agent Access Reality

Every deployment is layered on top of existing identity systems and trust assumptions that were designed for humans and static machines. Agents invoke services owned by different teams, traverse cloud and SaaS environments, and interact with systems that were designed for static workloads or direct human control.

This inherited complexity becomes more visible with scale. Recent research shows that the majority of organizations name insecure identities and risky permissions as their top cloud security risk, yet they lack the structure or workflows necessary to address these issues at scale.[5]

Given this reality, it's critical to ensure that agent access is explicit, limited to the task, and removed when the task is complete. That requires identity to be treated as part of deployment rather than an afterthought, and behavior to be enforced directly rather than inferred from accounts or roles. Reducing reliance on embedded secrets is part of this shift, but most environments must move incrementally, balancing new controls with existing systems that cannot be replaced overnight.

The Convergence of Human, Machine, and Agent Identity

AI agents do not operate in isolation. They act alongside existing machine identities and human access, often within the same workflows and systems. As a result, enterprises increasingly face a single identity problem expressed across different actors.

Applying consistent authentication, authorization, and policy logic across humans, machines, and AI agents reduces fragmentation and makes access decisions easier to explain, enforce, and audit. Without this consistency, teams are forced to manage parallel identity models that break down as agents begin to act autonomously inside shared environments.

Chapter 2:

A Practical Model for AI Agent Identity Security

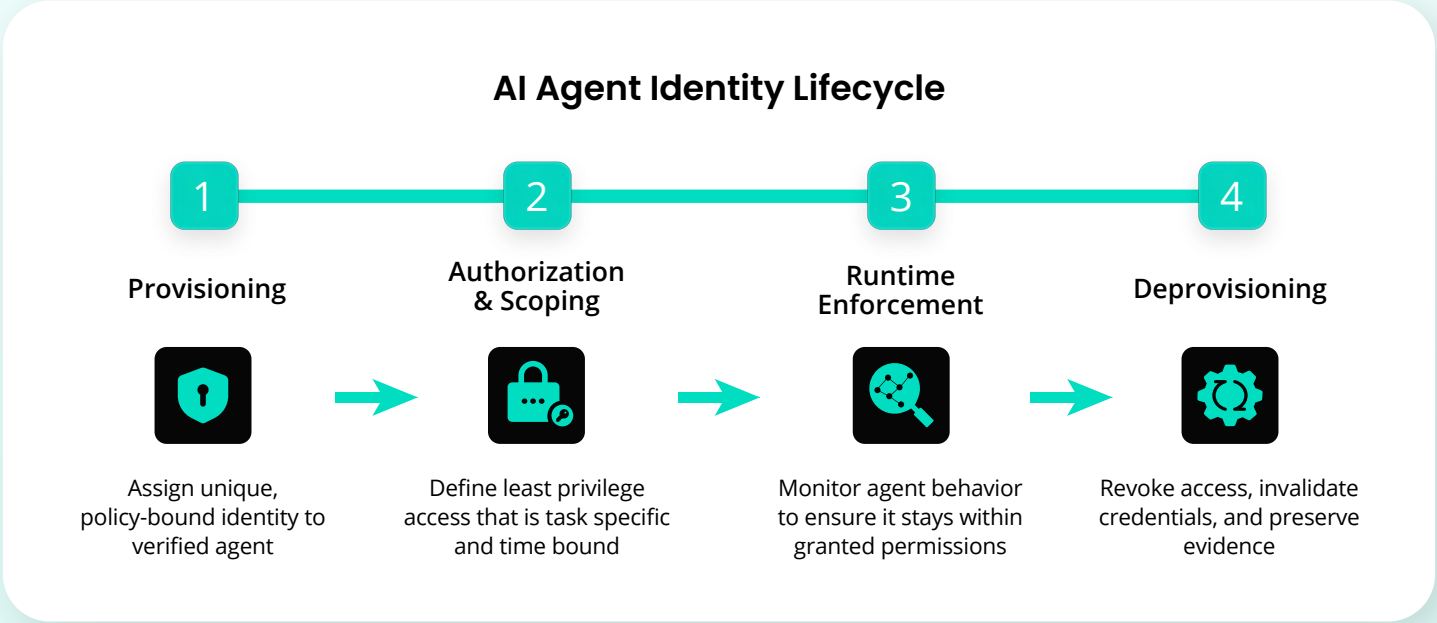
AI agent access is most secure when identity exists for a specific purpose and duration, not as a permanent object.

The AI Agent Identity Lifecycle

AI agent identity must be task- and time-bound. Treating identity as a lifecycle prevents standing access from persisting after work ends.

Most organizations have not adapted identity controls for autonomous agents. Research suggests that 95% of companies deploy agents without proper identity protection[6], leaving authentication and authorization boundaries weak or inconsistent. A lifecycle model makes access explicit, enforceable, and auditable as agents operate.

The AI agent identity lifecycle has four phases: provisioning, authorization and scoping, runtime enforcement, and deprovisioning.



Provisioning: identity issuance and policy binding

Provisioning happens when an agent is invoked to perform a task. Many teams shortcut this step by launching agents under existing service accounts or shared credentials and applying policy later, if at all.

Shared identities undermine accountability. During incidents, teams cannot tell whether an agent, a human, or another automation initiated a change. To contain risk, they often revoke access broadly, disrupting unrelated systems. Provisioning must establish three things before any work begins:

- A distinct identity that represents the agent as an actor
- Clear responsibility for what the agent may access
- Policy bound to that identity up front

Teams should establish identity using authentication mechanisms already present in the environment, such as federated identity, certificate-based authentication, or workload-native identity providers.

Authorization and task scoping

Authorization decides whether a specific agent request is allowed at that moment. For agents, access must align to the task being performed, not to static system roles.

Effective authorization is:

- Task-specific
- Time-bound
- Limited to the minimum required scope

Context determines whether access is appropriate. Environment, data sensitivity, and the tool or API being invoked all matter. Evaluating role-based and attribute-based policies dynamically, rather than at the system level, keeps permissions aligned with intent. Without this, access expands to cover edge cases and rarely contracts without breaking workflows.

Runtime enforcement and monitoring

Enforcement ensures agents stay within granted permissions even as execution paths change. Monitoring records what the agent attempted and what the system allowed.

Together, they provide clear answers to essential questions:

- What request did the agent make?
- What permissions applied at that moment?
- What outcome resulted?

This evidence should feed existing security operations systems, such as SIEM platforms, so teams can detect and investigate agent behavior as it happens, not after the fact.

Deprovisioning: access removal and evidence retention

When a task ends, identity and access must expire automatically. Persistent identities, credentials, and permissions accumulate risk.

Deprovisioning revokes access, invalidates temporary credentials, and preserves evidence for audit and investigation. Ephemeral identity reduces operational overhead by ensuring permissions exist only while work is active. When teams rely on manual cleanup, unused identities linger and surface later during audits or incidents.

Policy Design for AI Agents

Policy is what makes the agent identity lifecycle enforceable. For AI agents, policy cannot be expressed primarily in terms of systems or roles. Their behavior changes too quickly, and broad permissions granted for convenience tend to persist long after the original task is complete.

Task-scoped access

Policies for agents should answer one question at any given moment: what is this agent allowed to do right now?

Task-scoped policies grant access only for the duration of a specific task and only to the operations required to complete it. When the task ends, access expires automatically. This prevents agents from accumulating privileges that no longer reflect their purpose.

Task scope also makes policy easier to reason about. Instead of guessing which systems an agent might touch, teams define what outcome is permitted and constrain access accordingly.

Context-aware decisions

The same request can be acceptable in one situation and dangerous in another. Policy needs context to make that distinction reliably.

Environment, data sensitivity, and the tool or API being invoked all shape whether access should be allowed. Timing and approval state can matter just as much. Evaluating these inputs dynamically keeps policy precise without forcing teams to choose between overly broad access and brittle rules. When policy incorporates context directly, intent becomes explicit, enforcement becomes predictable, and audit trails are easier to interpret.

Guardrails for high-risk actions

Not all automated operations carry the same risk. Policies must differentiate routine tasks from sensitive operations such as modifying infrastructure, accessing production data, or changing access controls.

High-risk requests deserve tighter limits. This might mean narrower scopes, additional checks, or explicit approval before the agent proceeds. Without such guardrails, tools with broad access can amplify damage quickly. In the 2023 CircleCI breach, attackers exploited over-privileged credentials used by build and deployment automation to move laterally across customer environments and access sensitive systems.[7] The failure was not a missing patch or lack of detection, but insufficient constraints on what automated processes could do once authenticated.

Effective guardrails preserve autonomy where it is appropriate, while preventing routine automation from escalating into unintended privilege.

Getting to Secretless Agent Access

Secure AI adoption depends on establishing secretless, identity-based authentication, where agents gain access dynamically without storing credentials, reducing exposure while preserving agility. In a secretless model, agents authenticate using identity at the moment of access. Authority is granted for a specific task, for a limited time, and revoked automatically. Secrets and cryptography still exist, but they are kept out of agent logic entirely. Most environments reach this state incrementally.

The credential reality agents inherit

AI agents are typically deployed into environments built on long-lived credentials. API keys, tokens, and service account secrets remain embedded in pipelines, configuration, and integrations, often added during early experimentation or vibe coding and never fully removed. A 2025 report found nearly 24 million new leaked credentials on GitHub, with 70% of the secrets leaked in 2022 still valid today.[8]

This creates substantial risk. Secrets can be exposed within agent logic, execution paths, or intermediary tools.

From rotation to dynamic access

Secret rotation shortens exposure windows but leaves the access model unchanged. Credentials still live with the agent, and permissions persist beyond their intended use. Dynamic credentials issue permission only as needed and expire automatically, enforcing zero standing privileges.

Secretless, identity-based access

With identity-based access, agents no longer retrieve or store credentials. Identity is verified when permission is requested, and access is granted by policy for a limited purpose and duration. Credentials are issued and revoked transparently, keeping secrets out of agent logic while preserving compatibility with existing systems.

Why this matters for agents

Autonomy and long-lived secrets do not coexist well. Removing credentials from agent logic reduces exposure. Binding access to identity and task makes authority explicit. Automatic expiration removes cleanup as a manual process.

1st Stage

Where most organizations are

Static Secrets

Secrets stored in source code, configuration files and in basic vault stores

2nd Stage

Auto-Rotation

Automate periodic rotation of credentials & API Kevs

3rd Stage

Dynamic Identities

DyZero Standing Privileges (SP)

Auto creation & deletion of temporary machine identities only when required

4th Stage

Secretless

0Auth, OIDC, SPIFEE & ZSP

Using Advanced Authentication methods to enable the future of "SSO for Machines”

Cross-Domain Trust and Federation

A single AI agent task may require access to cloud services, SaaS APIs, internal platforms, and on-prem infrastructure, each governed by different identity and access models.

Shared credentials are often used to bridge these systems because they work across boundaries. For agents, this approach introduces persistent risk. Credentials must be distributed and stored in multiple environments, revocation is slow and difficult to coordinate, and permissions are rarely limited to a specific task or time window.
Federation offers a cleaner alternative. Instead of carrying credentials between systems, the agent proves its identity whenever permission is requested. Each environment independently verifies that identity and evaluates policy locally, based on task and context.

In practice, cross-domain access works as follows:

- The agent authenticates using its identity when permission is requested
- Each environment verifies the identity using its own trust anchors
- Authorization is evaluated locally, based on task scope and context

No credentials are shared between systems, and agents do not retain long-lived access. Trust is established on demand and expires automatically. This allows agents to operate across cloud, SaaS, and on-prem environments without redesigning existing systems, while keeping access decisions explicit and auditable.

Chapter 3:

Operationalizing Identity and Access for AI Agents

Access Control Does Not Belong Inside the Agent

Teams can quickly lose control when they embed access rules directly into agent logic. If credentials, approvals, or authorization checks live inside the agent, every policy change turns into a code change. Updating access means redeploying agents instead of adjusting policy.

Over time, different versions of the same agent enforce different rules. No one can say which controls are active, or where. When something goes wrong, teams scramble to trace behavior across deployments. Embedding access also magnifies failure. If an agent is misconfigured or compromised, it carries its permissions wherever it runs. Revoking access means finding every instance and redeploying under pressure.

Effective identity design separates responsibility. Agents decide what to attempt. Identity and access systems decide what is allowed. That separation lets teams change access without rewriting agents and correct mistakes quickly.

Control Plane and Enforcement, in Practice

Once access moves out of agent logic, a practical question follows: where do identity and access decisions live?

In production systems, teams split responsibility across two layers. The control plane issues identities, defines policy, and evaluates access decisions. Centralization keeps rules consistent, auditable, and manageable across teams and environments.

Enforcement happens where requests occur. When an agent tries to read data, invoke a tool, or modify a system, the platform evaluates that request against policy immediately and allows or blocks it. Teams adjust access without redeploying agents or embedding credentials.

This separation preserves speed without sacrificing control. Engineers ship agents independently. Security teams adjust access centrally. The same model already secures human and machine access at scale by pairing centralized policy with enforcement at the point of decision.

Designing for Failure and Intervention

AI agents will behave in unexpected ways. Policies will lag behind new use cases. Context will be misread. What matters is whether teams can detect drift early and respond before it spreads.

Strong access design makes that possible. When identity remains intact and enforcement happens at decision time, teams observe agent behavior as it unfolds instead of reconstructing it later.

- Effective systems answer three questions quickly:
- What request did the agent make?
- Under what identity, authority, and context?
- Did that request align with policy at that moment?

Clear answers enable intervention. Teams can pause execution, narrow scope, or revoke access immediately without changing agent logic. Controls apply at the access layer, not inside the agent.

Some operations still require human judgment. Changes to production data, infrastructure, or access controls often demand approval or escalation. These safeguards work best when tied to identity and policy rather than embedded in agent workflows.

Intermediary Layers and MCP

Many agent architectures rely on intermediary layers to handle external calls, tool invocation, or system access. These layers provide a natural place to apply identity verification and access policy without embedding logic into the agent itself.

The Model Context Protocol (MCP) standardizes how agents pass context and requests to external services. MCP servers act as mediation points where identity can be verified, policy evaluated, and short-lived access granted. This approach supports consistent, federated enforcement across cloud, SaaS, and on-prem environments while keeping credentials and policy logic outside agent code.

Zero-Knowledge Trust Architecture

As agent access expands, trust assumptions become increasingly important. Identity systems should minimize what any single component can observe or reconstruct, including infrastructure providers and intermediary services. Zero-knowledge principles reduce unnecessary exposure by ensuring credentials, keys, and sensitive material are never fully visible to any one party. Applied to agent access, this limits the consequences of compromise, supports separation of duties, and helps organizations meet regulatory and compliance requirements.

The Salesloft–Drift Attack: OAuth Tokens as the Weak Link

In 2025, attackers used stolen OAuth tokens from the Drift–Salesloft integration to gain unauthorized access to customer Salesforce environments, resulting in data exposure across multiple organizations.

How the attack unfolded

1 Token compromise

OAuth access and refresh tokens issued for the Drift–Salesloft integration were stolen. These tokens were long-lived and broadly trusted.

2 Valid authentication

Requests made with the stolen tokens authenticated successfully to Salesforce. The activity appeared legitimate at the protocol level.

3 Cross-system access

The tokens enabled access to Salesforce data across customer environments, allowing records to be queried and extracted.

4 Late detection

Abuse was detected only after unusual access patterns emerged, not at the moment of misuse.

5 Manual containment

Response required revoking tokens, rotating credentials, and auditing Salesforce access after the fact.

The incident underscores a persistent weakness in automated integrations: long-lived tokens function as standing access. Once exposed, they can be replayed across systems with little friction.

Chapter 4:

Evaluating Platforms for AI Agent Identity Security

When evaluating platforms for AI agent deployments, focus on whether they can issue identity, enforce access, and produce accountability as agents act across systems. This includes the ability to correlate agent activity over time to detect access drift and anomalous behavior early.

In practice, these capabilities are easiest to apply and operate when they are delivered through a unified control plane rather than stitched together from separate identity, secrets, and access tools. Fragmented tooling often recreates the same gaps agents exploit.

What to Evaluate for AI Agent Identity Security

Agent Identity Issuance

If identity is reused or shared, downstream controls will be ineffective

- Can the platform issue a distinct identity to an agent at runtime?
- Can that identity be bound to ownership and policy before the agent acts?
- Can identities be removed automatically when work ends?

Task-Scoped Authorization

Static roles and long-lived permissions do not hold up under autonomous behavior

- Can access be granted for a specific action rather than a broad system role?
- Can permissions be time-bound and expire automatically?
- Can authorization reflect task context such as environment, tool, or data sensitivity?

Runtime Enforcement

If enforcement depends on embedded credentials or agent code, control will drift

- Are access decisions enforced as actions take place?
- Can policy changes take effect immediately without redeploying agents?
- Does enforcement sit outside agent logic?

Secretless Progression

Secretless is a path, not a prerequisite, but the path must exist

- Can the platform reduce reliance on long-lived credentials?
- Does it support dynamic or identity-based access at the time of access?
- Can agents operate without directly handling secrets?

Cross-Environment Operation

Agents rarely stay inside a single trust boundary

- Can identities and policies span cloud, SaaS, and on-prem systems?
- Is federation supported without shared secrets as integration glue?

Evidence And Accountability

Without oversight, failures are hard to explain and harder to contain.

- Can you determine which agent acted, under what authority, and with what result?
- Is evidence available in real time, not reconstructed later?

Compliance and Audit Alignment

Controls that cannot be mapped to regulatory frameworks increase audit friction and operational risk.

- Do access controls map to standards such as SOC 2, ISO 27001, PCI DSS, and NIST SP 800-207?
- Can audit evidence be produced directly, without manual correlation?
- Can controls be applied consistently across regulated environments?

Secrets and Certificate Discovery

Unmanaged secrets and certificates remain a common source of unintended access.

- Does it use AI-driven discovery to identify secrets and certificates across code, pipelines, and infrastructure?
- Can findings be linked to ownership and usage context?
- Can rotation or revocation be triggered automatically when issues are found?

Platforms that meet these criteria enable secure AI agent access as deployments grow.

Platforms that do not will force teams to trade speed for control, or control for visibility.

Why Common Approaches to Securing AI Agents Fall Short

Most failures in securing AI agents stem from applying controls designed for visibility, storage, or static workloads to systems that act autonomously across multiple trust boundaries.

Gartner analysts warn that more than 40 % of agentic AI projects are expected to be cancelled by 2027 due to high costs, unclear business value, and insufficient risk controls,[9] a signal that foundational issues like identity, access, and accountability remain unresolved.

Visibility vs. control

Many approaches to securing AI agents emphasize visibility. Dashboards show which agents are active, which tools they call, and which systems they touch. Telemetry and alerts provide useful signals after the fact. What's often missing is the ability to constrain actions as they take place. Knowing that an agent accessed a system is not the same as being able to prevent or limit that access in real time.

Secrets management vs. agent security

Secrets management is often treated as the foundation of agent security because it centralizes credentials, enforces rotation, and supports auditing. Many approaches stop once a secret is retrieved, leaving how that credential is used outside the security boundary. Agents then authenticate with long-lived API keys, OAuth tokens, or other reusable credentials injected at execution time, giving them broad and persistent access. Once credentials enter agent logic, they can leak through execution traces, memory, or downstream tools, increasing exposure. Securing agents requires changing how access is granted, not just where secrets are stored.

System-level roles vs. task-level authorization

Many access models rely on system-level roles and permissions. This works for static workloads and human users. AI agents behave differently. A single task may span multiple systems, tools, and data sources in one flow. Granting broad roles to avoid friction creates persistent over-permission that is easy to miss during evaluation and hard to unwind in production. Approaches that support task-scoped, time-bound authorization align better with autonomous behavior.

Observation vs. identity issuance

Some approaches detect or classify agents without issuing a distinct identity. Agent activity is represented through shared service accounts, integrations, or tokens. As deployments scale, attribution erodes. Activity becomes harder to trace to a specific agent, task, or owner, even when records exist. Observing agents is not the same as establishing a verified identity.

Audit readiness vs. operational readiness

Audit artifacts are often treated as a proxy for security. Reports are complete, logs are retained, and evidence can be assembled after an incident. Operational readiness depends on understanding and responding as events unfold. Which agent is acting now? Under what authority? Can access be adjusted immediately?

For AI agents, behavior needs to be quickly understood and constrained.

AI Agent Identity Security Evaluation Checklist

This checklist can be used as a platform evaluation artifact, procurement input, or internal readiness assessment for organizations deploying AI agents in production environments.

Agent Identity Issuance

- [ ] Issues a distinct identity for each AI agent
- [ ] Supports on-demand identity issuance tied to agent execution
- [ ] Binds agent identity to policy before access is granted
- [ ] Supports automatic identity expiration or removal when work completes

Authentication and Federation

- [ ] Authenticates AI agents using identity-based mechanisms
- [ ] Supports federated authentication across cloud, SaaS, and on-prem environments
- [ ] Supports standards-based workload and service identity federation
- [ ] Eliminates the need for static credentials embedded in agent logic

Task-Scoped Authorization

- [ ] Grants access at the level of task, operation, or request
- [ ] Supports time-bound permissions with automatic expiration
- [ ] Evaluates authorization using contextual inputs (environment, tool, data sensitivity)
- [ ] Supports fine-grained authorization across multiple target systems

Task-Scoped Authorization

- [ ] Enforces access decisions at the point access is requested
- [ ] Applies policy changes immediately without agent redeployment
- [ ] Separates enforcement logic from agent code

Secrets and Certificate Handling

- [ ] Supports dynamic credential issuance
- [ ] Uses AI-driven discovery to continuously identify secrets and certificates across code, pipelines, and infrastructure
- [ ] Applies AI-based analysis to identify unmanaged, orphaned, or misconfigured credentials and certificates
- [ ] Supports incremental adoption of secretless access patterns
- [ ] Prevents exposure of credentials within agent execution paths

Cross-Environment Operation

- [ ] Maintains consistent identity and access controls across cloud, SaaS, and on-prem systems
- [ ] Supports hybrid and multi-cloud deployments
- [ ] Centralizes policy definition with distributed enforcement

Visibility, Evidence, and Accountability

- [ ] Attributes activity to a specific AI agent identity
- [ ] Records access events with associated identity, policy, and context
- [ ] Captures access events in real time
- [ ] Supports investigation and audit without post hoc reconstruction

Compliance and Architectural Alignment

- [ ] Aligns identity, access, logging, and encryption controls with major frameworks (e.g., SOC 2, ISO 27001, PCI DSS, NIST SP 800-207, DORA)
- [ ] Integrates with existing IAM, security operations, and compliance tooling
- [ ] Supports coexistence with existing secrets management systems

Chapter 5:

From Identity Model to Deployment

Moving from principles to deployment requires more than selecting tools. It requires understanding the environment AI agents will operate inside, the constraints that shape access decisions, and the maturity of existing identity and security controls

In practice, many organizations find that applying these principles consistently becomes difficult when identity, secrets, and access controls are split across disconnected systems. Unified platforms can reduce this friction by centralizing policy, identity, and enforcement while still integrating with existing infrastructure.

It also requires accounting for current and emerging security risks. Frameworks such as the OWASP Agentic AI Top 10 underscore the threats targeting autonomous AI systems, including identity and privilege abuse, tool misuse, and goal-hijacking exploits.[10]

There is no single starting point. Organizations differ widely in architecture, regulatory pressure, and operational complexity. The access model described in this guide is designed to adapt to those realities, not override them.

Understanding Your Deployment Context

Before deploying AI agents into production systems, teams need to understand the environment those agents will inherit. Some organizations operate largely within a single cloud ecosystem, with a dominant identity provider and relatively consistent tooling. Others span multiple cloud providers, SaaS platforms, legacy data centers, and custom internal systems, each with different trust assumptions and access models.

These differences affect how agent identity is established, where enforcement can occur, and how access decisions propagate across systems. Regulatory requirements and data sensitivity further shape what “acceptable” agent behavior looks like, particularly where attribution, evidence retention, and separation of duties are required. Most environments also sit somewhere along a spectrum of secrets maturity, from static credentials embedded in applications to more dynamic, identity-based access. None of this changes the access model itself, but it determines how that model must be applied in practice.

How Akeyless Helps Secure AI Agent Deployments

The Akeyless platform provides a unified way to establish agent identity, broker access, and control sensitive actions without embedding credentials or policy logic into agents themselves.

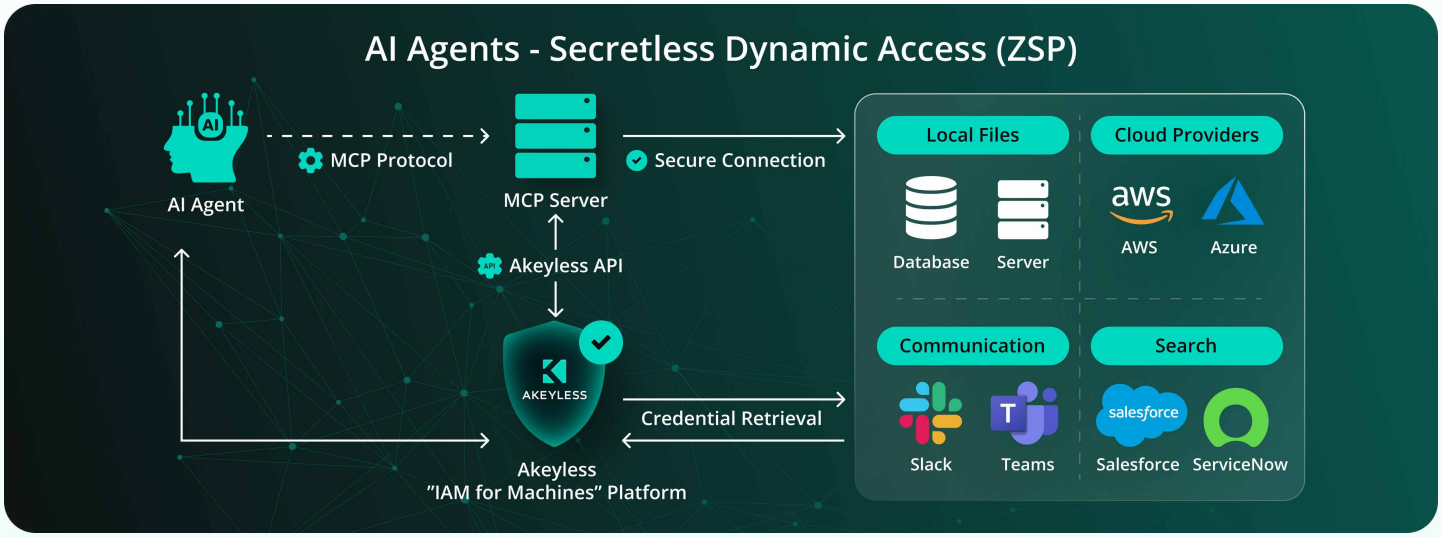
Akeyless issues just-in-time, verifiable identities to AI agents, allowing them to authenticate across cloud, SaaS, and on-prem environments. Access is granted based on policy and expires automatically, keeping authority explicit and time-bound as agents operate autonomously.

The access path adapts to the target system. Where modern federation is available, identity-based, ephemeral access is issued directly. For legacy databases and on-prem systems, access is brokered through local gateways that enforce the same policies without exposing long-lived credentials. For sensitive or privileged operations, Akeyless applies policy-based controls and monitoring so agent activity remains visible and attributable without slowing deployment. Across all cases, secrets stay out of agent logic and supporting toolchains.

A detailed mapping of these capabilities to the access requirements defined earlier is provided in the Appendix.

Example Execution Flow

The execution flow below illustrates one common pattern using a mediation layer such as an MCP server:



1 AI Agent → MCP Server

The agent initiates a task and sends a request through a Model Context Protocol (MCP) server.

2 MCP Server → Akeyless API

The MCP server authenticates to Akeyless using its existing infrastructure identity, such as a cloud IAM role, Kubernetes service account, or GitHub-issued JWT.

3 Akeyless → Target System

Akeyless evaluates policy based on identity, task, and context, then connects to the target system (database, SaaS application, or cloud service) and provisions a short-lived, task-scoped credential.

4 MCP Server → AI Agent

The credential is used to establish a controlled session that allows the agent to complete its task. The credential expires automatically when the task ends.

Advanced capabilities for secure agent deployment

As AI agents take on broader responsibilities and operate across more systems, access security must extend beyond basic authentication and authorization. At scale, autonomous behavior demands stronger cryptographic guarantees and deeper operational insight.

Akeyless applies zero-knowledge design principles and cryptographic controls that prevent any single system or operator from reconstructing sensitive material, while providing post quantum protection. AI-assisted analysis helps security teams detect anomalies, investigate agent behavior, and understand emerging access patterns, enabling faster response and more effective policy tuning.

Key Deployment Considerations Moving Forward

As AI agents move from pilots into sustained production use, the most important decisions are architectural rather than procedural.

Anchor deployment decisions in identity

Treat agents as actors with authority. Clear identity is what makes access decisions explainable and enforceable as deployments expand.

Decouple agent behavior from access control

Keeping enforcement outside agent logic allows access to evolve independently of how agents reason, plan, or execute tasks.

Reduce credential exposure over time

Progress toward secretless access by shrinking where credentials exist and how long they live, rather than attempting to eliminate them all at once.

Design for mixed environments

Expect agents to cross boundaries between cloud services, SaaS platforms, and legacy systems, and plan for consistent access control across them.

Build attribution in from the start

The ability to trace actions back to a specific agent and authorization context is what makes incidents containable and audits survivable.

Platforms like Akeyless are designed to help teams apply these principles in practice. To learn more about securing AI agent deployment and access, or to see how Akeyless does it, schedule some time with one of our experts at akeyless.io/demo.

Resources

[1] Predicts 2026: Secure AI Agents to Avoid Ungoverned Sprawl and Abuses, Gartner, December 2025
[2] AI agents: The new attack surface, SailPoint and Dimensional Research, May 2025
[3] The State of AI Agent Security 2026, Neural Trust, November 2025
[4] The NHI & Secrets Risk report H1 2025, Entro Labs
[5] The State of Cloud and AI Security 2025, Cloud Security Alliance, September 2025
[6] https://www.csoonline.com/article/4109999/agentic-ai-already-hinting-at-cybersecuritys-pending-identity-crisis.html
[7] https://nhimg.org/circleci-breach
[8] The State of Secrets Sprawl 2025, GitGuardian, April 2025
[9] https://www.csoonline.com/article/4109999/agentic-ai-already-hinting-at-cybersecuritys-pending-identity-crisis.html
[10] https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

Appendix

The table below shows how Akeyless supports the recommended evaluation criteria and architectural requirements described in this guide.

Evaluation area	How Akeyless implements it
Agent identity	Issues short-lived, verifiable identities to AI agents using native workload attestation, binding identity to policy and ownership without pre-provisioned accounts or bootstrap secrets
Authentication and federation	Authenticates agents and intermediaries using cloud IAM roles, Kubernetes service accounts, certificates, or federated identities across cloud, SaaS, and on-prem environments
Authorization and access scope	Evaluates policy per task and context, translating authorization decisions into time-bound entitlements that expire automatically
Runtime enforcement	Enforces access outside agent logic through just-in-time credential issuance and controlled sessions, with policy changes taking effect immediately without redeploying agents
Secrets handling and path to secretless	Provisions short-lived credentials at execution time and keeps secrets out of agent inputs, execution paths, memory, logs, and toolchains. Supports incremental adoption of secretless access while minimizing exposure when credentials are required.
Cross-environment operation	Combines native federation with Akeyless Gateways that broker access locally to databases, servers, and legacy systems, enforcing the same policies across cloud, SaaS, and on-prem environments
Privileged and sensitive agent actions	Applies additional policy controls, session visibility, and approval hooks for high-risk agent operations involving production data, infrastructure, or code
Visibility, monitoring, and evidence	Records identity issuance, access events, and policy decisions with agent-level attribution; Akeyless Jarvis™ analyzes identity and access behavior to surface risk, anomalies, and misuse across agents and non-human identities
Cryptographic trust and future resilience	Applies patented Distributed Fragments Cryptography™ (DFC™), so no single system or operator can reconstruct sensitive key material. Uses hybrid post-quantum transport security (Hybrid TLS 1.3 with ML-KEM768) to protect identity, secrets, and access channels against current and future cryptographic threats.
Unified identity and access platform	Centralizes secrets, certificates, workload identity, and policy enforcement in a single control plane, enabling consistent access control and auditing for human, machine, and AI agent access without relying on multiple disconnected systems
Agent frameworks and tooling integrations	Integrates with common agent frameworks and automation tools such as Cursor, n8n, and CI/CD systems, allowing agents to request access through identity-based flows rather than embedded credentials

AKEYLESS

About Akeyless

Akeyless is the identity security platform built for machines, AI agents, and the people who depend on them. The platform unifies secrets management, certificate lifecycle automation (PKI), privileged access (PAM), identity governance, and identity visibility and intelligence, to improve efficiency while securely accelerating modernization.

Built on a patented zero-knowledge, post-quantum–ready cryptographic foundation, Akeyless reduces risk, improves visibility, and lowers operational costs. Enterprises choose Akeyless to manage the entire lifecycle of their secrets and identities, including discovery, governance, management, protection, and secure access.

Take control of your identities and secrets.

Request a demo to see how at akeyless.io/demo

Akeyless.io