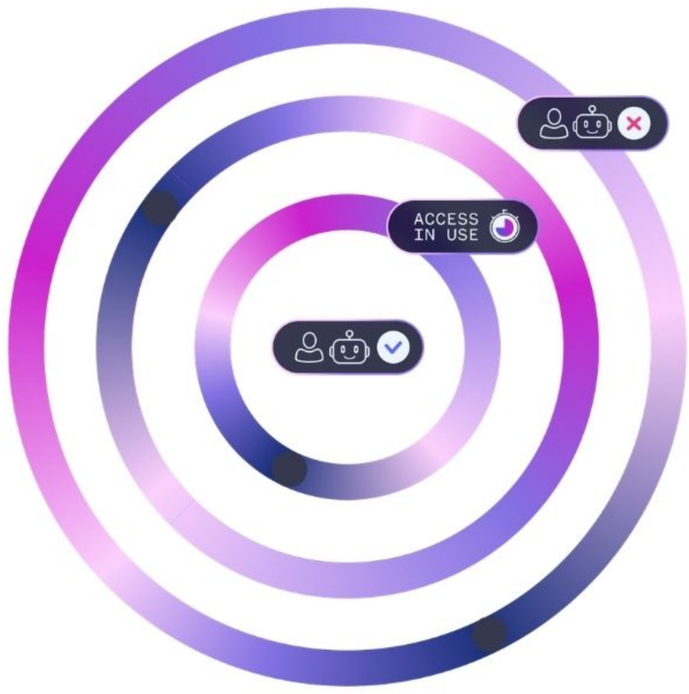
britive

The Definitive Guide to Zero Standing Privilege



EXECUTIVE OVERVIEW

THE END OF STANDING PRIVILEGE

For decades, privileged access management (PAM) has been built on a single, dangerous assumption: that privileges must exist somewhere to be used.
We created administrator accounts, service accounts, and standing roles. We secured them in vaults, rotated their passwords, and monitored their sessions. But fundamentally, the access remained. It sat dormant in the environment, waiting for a legitimate user, or a patient attacker.

This model of "managed privilege" worked when infrastructure was static and change was slow. But in the modern enterprise, it's quickly become a structural liability.

Cloud-native environments are ephemeral. DevOps pipelines deploy code hundreds of times a day. Non-human identities outnumber human users by orders of magnitude. And now, the rise and rapid adoption of AI agents introduces non-deterministic actors that have rapidly changing access needs at machine speed.

> IN THIS NEW REALITY, **STANDING PRIVILEGE IS A FAILURE MODE.**

Any permission that exists before it is needed contributes to an attack surface that is impossible to defend manually. It is also the last place implicit trust hides.

This guide introduces the architectural standard that replaces legacy management: Zero Standing Privilege (ZSP).

ZSP is not a theoretical ideal; it is a practical operating model that can remove the implicit trust that organizations are trying to remove, bringing them closer to achieving Zero Trust. It is the shift from managing secrets to removing them. By ensuring that no user, machine, or agent holds privilege by default, organizations can replace fear-driven security with a foundation of trust that enables innovation rather than slowing it down.

WHAT IS ZERO STANDING PRIVILEGE?

At its core, Zero Standing Privilege (ZSP) is an authorization state where **no identity, either human or machine, possesses inherent privileged access to any resource.**

In a ZSP architecture, the default state of every identity is zero trust, zero access. Privilege is not a property of the user; it is a temporary capability granted only at the moment of need.

The Core Definition

ZSP requries that access adheres to three strict conditions:

- **Minted on Demand**
  Permissions do not exist until they are requested.
- **Scoped to the Task**
  Access is limited to the specific resource and action required (adhering to the principle of least privilege).
- **Automatically Destroyed**
  Permissions expire and are revoked immediately upon task completion.

A CLEAR MENTAL MODEL FOR THE FUTURE OF ACCESS

How ZSP Differs from Traditional Models

vs. Traditional PAM

Legacy PAM relies on pre-provisioned admin accounts stored in a vault. The privilege exists 24/7; the vault just controls who can check out the keys. In ZSP, there are no keys to vault because the admin account does not exist until needed.

---

vs. Role-Based Access Control (RBAC)

RBAC relies on static assignments (e.g., "John is in the Admin Group"). This creates access creep as users accumulate roles over time. ZSP grants access based on context and policy, not just group membership or role.

---

vs. Static Admin Accounts

These are break-glass or just-in-case accounts. In a ZSP model, these are replaced by dynamic elevation, removing the risk of a dormant super-user account being compromised.

THE EVOLUTION OF PRIVILEGED ACCESS MODELS

AND WHY THE INDUSTRY IS SHIFTING FROM ACCESS MANAGEMENT TO SECURE ARCHITECTURE.

To understand why ZSP is necessary, we must look at how access models have failed to keep pace with infrastructure.

PHASE 1

Static Privilege - The Era of the Keys to the Kingdom

In the early days, we created root and admin accounts. These were shared, long-lived, and incredibly dangerous. Security relied entirely on trusting the humans who knew the password.

PHASE 2

Vault-Based PAM — The Managed Risk Era

As risks grew, we introduced vaults. We took those static keys, locked them away, limited access to them, and rotated them periodically.

While this improved hygiene, it did not solve the root problem.

- **Vaulting ≠ ZSP**

  A vaulted secret is still a standing privilege. If the vault is breached, or an endpoint is compromised after checkout, the risk is absolute.

- **Rotation ≠ ZSP**

  Rotating a password every 60 minutes still leaves a 59-minute window of vulnerability.

The Operational Cost

This era also introduced significant overhead. As organizations scale their cloud footprints, they often attempt to retrofit legacy, vault-based PAM into modern environments. This creates a massive architectural burden known as the integration tax.

Traditional PAM was built for static servers. To make it work in a dynamic cloud, vendors must stitch together disjointed modules:

Endpoint Agents	Requiring heavy software on every target resource, slowing down deployments and creating maintenance nightmares.
Proxies and Jump Hosts	Forcing developers to route their workflows through network bottlenecks, breaking native CLI tools and frustrating engineering teams.
Synchronization Delays	Relying on scheduled syncing between the vault and the cloud provider, meaning a newly created database might not be protected until the next sync cycle.

PHASE 3

Runtime Authorization & ZSP — The "Architected" Era

This is the modern standard. Instead of managing static credentials, we move to **runtime authorization**. It replaces stitched-together modules with a natively unified control plane. By leveraging API-driven runtime authorization, ZSP platforms interface directly with the cloud provider's native IAM structures (like AWS STS or GCP bindings).

There are no agents to install, no proxies to route through, and no synchronization delays. Security becomes decoupled from static infrastructure, allowing you to secure hybrid environments, human users, and agentic AI through a single common policy.

No Stored Credentials	We stop creating accounts just in case.
Ephemeral Access	We use cloud-native primitives (like AWS STS tokens or short-lived certificates) to grant access for minutes, not months.
Policy as Code	Decisions are made in real-time based on policy, not static assignments.

KEY INSIGHT

ZSP is not a single, standalone feature that you buy; it's an operating model you adopt. It moves security from securing the keys to changing the lock entirely.

THE TWO PLANES OF IDENTITY: ASSURANCE VS. AUTHORIZATION

A common point of confusion is where ZSP fits in the modern identity stack. It is helpful to distinguish between two distinct planes of operation:

PLANE 1

Identity Assurance (Authentication)

The Question	Who are you?
The Tools	SSO, MFA, IdP (e.g., Okta, Entra ID).
The Function	This is the front door. It verifies that the user is who they say they are.
The Limitation	Once the user walks through the door, authentication stops. It cannot control what the user does inside the cloud environment or for how long.

PLANE 2

Runtime Authorization (ZSP)

The Question	What can you do right now?
The Tools	Cloud Infrastructure Entitlement Management (CIEM), Next-Gen PAM, Britive.
The Function	This is the engine. It evaluates intent, context, and policy to grant specific permissions for a specific timeframe.
The Power	ZSP lives here. It picks up where authentication leaves off, ensuring that a verified identity doesn't become a runaway risk. In Zero Trust terms, authentication ensures no implicit trust, while runtime authorization ensures that access is always verified.

WHY STANDING PRIVILEGE IS A STRUCTURAL RISK

THE HIDDEN COST OF JUST-IN-CASE ACCESS.

In a modern environment, standing privilege isn't just bad hygiene; it's a structural flaw that attackers actively exploit.

The Growing Attack Surface

Every standing permission is a potential entry point.

- **Credential Theft**

  If an admin has permanent access to Production, a phishing attack on their laptop gives the attacker the keys to the kingdom.

- **Lateral Movement**

  Attackers look for over-privileged service accounts to jump from a compromised dev environment to production databases.

The Velocity Trap and Natural Bottleneck

Beyond security, standing privileges slow you down.

- **Role Explosion**

  As you add clouds and services, you add roles. Managing thousands of static roles becomes a manual burden that slows down onboarding and provisioning.

- **Audit Drag**

  Proving compliance requires reviewing every user's standing access. In a ZSP model, you audit activity, not possibility, making compliance faster and more accurate.

The AI Multiplier

This is the new urgency. AI Agents are **non-deterministic.** They figure out how to solve problems on the fly.

- You cannot pre-provision an AI agent without giving it broad, dangerous access.
- If you give an agent standing admin rights, a simple prompt injection could wipe your database.
- **ZSP is the only safe architecture for AI.** It allows the agent to request exactly what it needs, use it for milliseconds, and then lose it immediately.

THE PRINCIPLES OF A ZERO STANDING PRIVILEGE ARCHITECTURE

To achieve ZSP, organizations must adopt a new architectural mindset centered on **ephemeral access**.

PRINCIPLE 1

Identity as the New Perimeter

We stop relying on network boundaries. The identity is the control plane. Whether Human, Machine, or AI, every actor is treated with the same Zero Trust baseline.

PRINCIPLE 2

Minting Access on Demand

We move away from provisioning (which implies permanence) to **minting**.

What this means	Just as a cryptographic token is generated for a single session, permissions should be dynamically created at the moment of request.
Why it matters	This leverages native cloud capabilities (like tokenization and dynamic roles) rather than legacy account creation. It is faster, cleaner, and leaves no artifact behind to be stolen.

PRINCIPLE 3

Ephemeral by Design

Access must self-destruct. Whether it is a 1-hour window for a developer or a 300-millisecond token for a bot, the expiration is hard-coded into the credential itself. Revocation is not an afterthought; it is a default condition.

HOW ZSP WORKS IN PRACTICE

A DAY IN THE LIFE OF A ZSP ENVIRONMENT

Request	A Cloud Engineer (or a CI/CD pipeline) needs access to an S3 bucket to fix a production issue.
Verification	They authenticate via their SSO.
Authorization	The ZSP platform evaluates the request against **policy**: Is this user allowed? Is it during business hours? Is the risk score low?
Minting	**a.** If approved, the platform **mints** a short-lived, time-bound access token (e.g., an AWS STS token) with the exact permissions needed. **b. Crucially**: No agent was required on the target server. The access was provisioned via API.
Action	The engineer performs the fix.
Termination	The token expires automatically. The access vanishes. The admin identity ceases to exist.

Redefining the Audit: From Possibility to Reality

In a legacy environment, auditing privileged access is an exercise in auditing possibility. Because standing privileges exist 24/7, auditors must review massive spreadsheets of group memberships to determine who could have accessed a critical system.

When a breach occurs, security teams spend weeks correlating network logs with vault checkouts to reconstruct what actually happened.

Zero standing privilege changes the audit paradigm entirely. Because access is minted at the exact moment of execution and destroyed immediately after, the authorization log is the definitive truth.

Attribution

Every action is tied back to a specific identity, policy evaluation, and business justification.

Scope

The log proves exactly what permissions were granted and precisely when they were revoked.

Machine-Speed Compliance

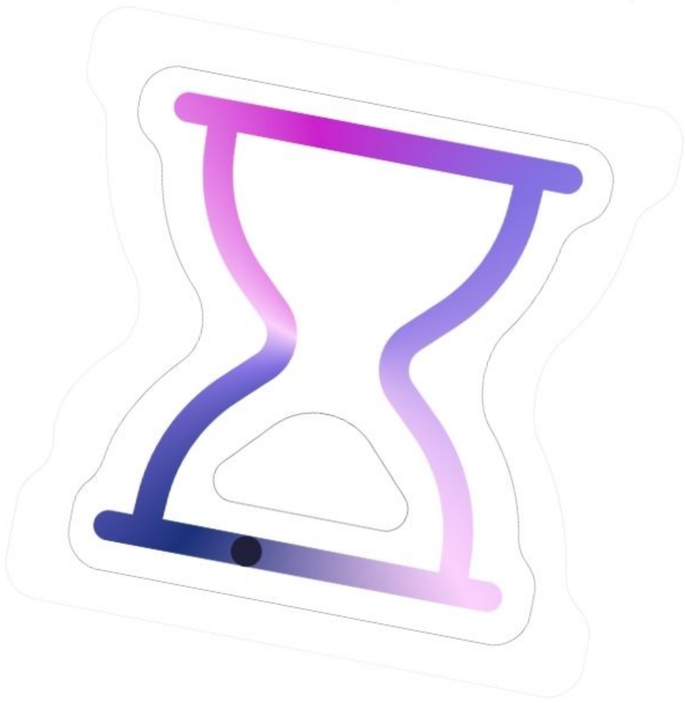
This applies equally to agentic AI and non-human identities.

In a ZSP architecture, compliance ceases to be a quarterly manual review. It becomes a continuous, mathematically provable byproduct of your runtime authorization engine.

OPERATIONALIZING ZSP: A MATURITY MODEL

THIS IS A JOURNEY, NOT A SWITCH FLIP

Most organizations cannot achieve 100% ZSP overnight.
The goal is to move the highest-risk assets to this model first.



PHASE 1

Assess and Reduce Access

- Inventory your standing privileges.
- Identify the critical assets like production databases and root accounts.
- **Goal**: Stop the bleeding. Prevent new standing admin accounts from being created.

PHASE 2

Implement Just-in-Time Access for Human Users

- Move human admins to on-demand access.
- Replace permanent admin roles with eligible roles that require checkout.
- **Goal:** Eliminate the risk of compromised employee credentials.

PHASE 3

Enable Policy-Driven Access for Machines

- Target service accounts and CI/CD pipelines.
- Replace long-lived API keys with ephemeral tokens minted at runtime.
- **Goal:** Secure the supply chain and automated workflows.

PHASE 4

Utilize Full ZSP for AI Agent Security

- Implement runtime authorization for AI agents.
- Ensure agents operate with 0% standing privilege, requesting access as needed during task execution.
- **Goal:** Enable autonomous innovation without existential risk.

COMMON MISCONCEPTIONS ABOUT ZSP

MYTH

"ZSP slows down engineers."

REALITY

ZSP accelerates engineers. By using pre-approved access profiles, developers get instant, self-service access without waiting for a helpdesk ticket. It removes the friction of manual approvals for routine tasks.

MYTH

"We already have JIT, so we are done."

REALITY

Many JIT implementations are just "manual provisioning with a timer." If you are still manually creating accounts and then deleting them later, you have operational drag. ZSP automates the minting and burning of access.

MYTH

"ZSP is too hard to implement for legacy apps."

REALITY

While ZSP is native to the cloud, modern ZSP platforms can extend this logic to legacy on-prem apps by managing local accounts dynamically. You don't need to refactor the app to secure the access.

EVALUATING ZERO STANDING PRIVILEGES IN PRACTICE

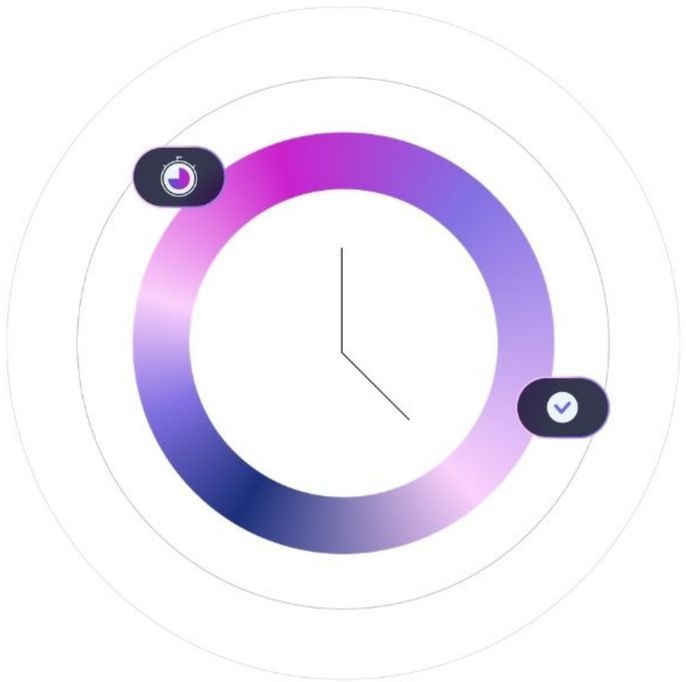
Not every tool claiming to offer zero standing privilege actually delivers it. Many solutions simply automate the creation and deletion of standing accounts, leaving operational drag in their wake.

When evaluating your access model or a vendor's claims, use this matrix to distinguish between legacy PAM, time-bound standing access, and true ZSP architecture.

ARCHITECTURAL DIMENSION	LEGACY VAULT-BASED PAM	TIME-BOUND STANDING ACCESS	RUNTIME ZSP ARCHITECTURE
When Access Exists	24/7 (Stored)	Before request & during session	Only during execution
Infrastructure Required	Heavy (Endpoint Agents, Proxies)	Moderate (Sync scripts, runners)	Minimal (API-driven)
Default State	Dormant risk	Standing access (hidden)	Zero access
Enforcement layer	The Vault	Identity Provider / Scripts	Natively at the resource
Agentic AI parity	None	Partial / Broken	First-class citizen
Audit signal	Credential checked out	Account activated	Intent verified & minted
Zero Trust Alignment	Implicit trust (always on)	Trust reduced, not removed.	No implicit trust. Verified per request

CONCLUSION

PRIVILEGE WITHOUT PERSISTENCE



The transition to Zero Standing Privilege is the inevitable next step for security architecture. Zero Trust promises an end to implicit trust, and standing privilege is the last place it survives. You can't claim Zero Trust while standing privileges exist, and ZSP is what closes that gap.

As we adopt cloud-native technologies and prepare for an AI-driven future, the old methods of managing secrets simply cannot scale. They are too slow for our machines and too complex for our humans.

ZSP offers a different path. It allows us to align security with the velocity of the business. By removing the "standing" risk, we remove the fear. We give our teams the freedom to build, deploy, and innovate, knowing that their access is secure, attributable, and ephemeral by design.

It is time to stop managing the keys, and start architecting a world where we no longer need them to be kept.