britive

8 Questions to Pressure-Test Your PAM Vendor

How to tell if a PAM vendor’s architecture is fit for cloud, ephemeral workloads, and agentic identity, or if it’s just retrofitted to look like it is.

For more than a decade, privileged access management meant one thing: a vault. Store the credential, rotate it on a schedule, broker the session, log the activity. That model worked when privileged access lived behind a small number of long-lived servers and a stable population of admins.

Modern environments look nothing like that. Cloud and SaaS access is API-driven and federated. Workloads spin up and tear down in seconds. Non-human identities — service accounts, CI/CD pipelines, and now AI agents and MCP integrations — now far outnumber the humans they support. The credential is no longer the only thing worth protecting; in many cases, it shouldn't exist between sessions at all.

That's why the PAM conversation has shifted from "where do we store the credential?" to "should this credential exist when it's not being used?" The architectural answer is runtime Zero Standing Privileges (ZSP): no identity — human or non-human — holds privileged access at rest. Access is brokered just-in-time against policy, scoped to the task, and revoked at expiry.

THE QUESTIONS BELOW ARE DESIGNED TO EXPOSE HOW A VENDOR'S ARCHITECTURE ANSWERS THAT SHIFT — NOT JUST WHETHER THEIR FEATURE LIST MENTIONS IT.

Whether you're evaluating your first PAM platform or weighing renewal of one you've had for years, the 8 questions below are designed to pressure-test a vendor's architecture — not their pitch. Each includes context for why it matters and what to listen for in a credible answer.

QUESTION 1

Does the platform eliminate standing privileges at the source, or just store them more securely?

A recurring theme in modern cybersecurity is that PAM decisions are often fragmented across DevOps, InfoSec, and IT, creating massive organizational inertia. Vaultless JIT breaks this gridlock by delivering a foundational capability that provides a clear win-win-win for every stakeholder.

Why it matters

Vault-based PAM was designed to reduce the risk of standing privileges by centralizing and rotating credentials. But the credential still exists. It's still mapped to an identity. It still sits in storage between uses, waiting to be misused, replayed, or stolen by anyone who can read the vault metadata. The majority of breaches reported in recent years have involved compromised credentials. A credential that exists is a credential that can be compromised as the starting point of an attack.

True risk reduction means eliminating the standing credential, not protecting it better. Zero Standing Privileges (ZSP) — a state where no identity holds privileged access between sessions — collapses the entire attack surface that credential rotation, MFA, and session recording are designed to mitigate.

What good looks like

A vendor whose architecture is built around ZSP will describe access as something that's minted on demand and destroyed upon expiration, rather than something checked out and returned. They should be able to articulate what happens to the credential the moment a session ends, and the answer should be: it doesn't exist anymore.

Be wary of vendors who frame ZSP as a feature layered on top of a vault rather than as the underlying access model. The former narrows the window of exposure. The latter removes it.

QUESTION 2

What does "just-in-time" actually mean in their architecture?

Why it matters

"Just-in-time access" has become one of the most overloaded terms in security. Some vendors mean truly ephemeral credentials minted at request time. Others mean limited-duration checkout windows from a vault that holds a static credential. The two are architecturally different, and only one of them actually eliminates standing privilege.

The distinction matters because every downstream benefit — reduced lateral movement, smaller blast radius, simpler audit, less compliance overhead — depends on whether the credential exists between sessions.

What good looks like

Ask the vendor to walk through, end-to-end, what happens from access request to expiry. A credible answer involves a policy evaluation at request time, a credential or role assumption minted for the duration of the task, and an automatic revocation when the timer expires or the task completes. The user or workload never holds the underlying credential.

If the answer involves a credential that pre-exists in a vault, gets retrieved into the user's session, and is then returned or rotated, that's still vault-mediated access. It’s useful in managing and limiting access, but it doesn’t achieve true ZSP.

Also ask about time-to-live. Defaults access times that are measured in minutes (15–60) are aligned with modern threat models. Access that’s measured in hours or days are not.

QUESTION 3

How broadly does the platform cover modern infrastructure? Does it cover cloud, SaaS, Kubernetes, databases, and on-prem?

Why it matters

Modern enterprises run on an environment the vault model wasn't designed for: AWS, Azure, and GCP organizations; Kubernetes clusters and container registries; SaaS platforms like Salesforce, Snowflake, Okta, and ServiceNow; data warehouses, DBaaS, and on-prem systems that still hold critical workloads. Each has its own access pattern — federation, IAM roles, service principals, workload identity, native credentials — and a PAM platform that only handles one or two of them leaves the rest behind in a separate tool, with a separate policy and a separate audit trail.

The risk isn't just leaving coverage gaps behind. It's the operational and compliance cost of governing privileged access across three or four point solutions, each with its own console, policy language, and reporting model.

What good looks like

A modern PAM platform should manage privileged access consistently across cloud infrastructure, Kubernetes, SaaS applications, databases, and on-prem systems under a single policy engine and a single audit plane. Ask specifically about:

- Kubernetes coverage, often the weakest area in vault-centric platforms
- Infrastructure-level access (IAM roles, service principals, workload identities) where the highest-blast-radius privilege concentrates
- Whether the same policy applies consistently to a cloud admin role, a SaaS admin role, and a database admin role

Coverage at the application level alone is not enough. The platform should reach the infrastructure layer.

QUESTION 4

Does the same platform secure non-human identities, including AI agents and MCP-based workflows?

Why it matters

Non-human identities (NHIs) — service accounts, CI/CD pipelines, API keys, integration users — already outnumber human identities in most enterprises, often by ten to one or more. The emergence of agentic AI and protocols like MCP (Model Context Protocol) only continues to compound the difference. One human user can spawn dozens of short-lived agent identities, each calling tools and APIs on the user's behalf.

Vault-centric PAM was never built for this. Vaulting a credential per agent is architecturally untenable. Granting an agent a session-long credential gives it the same blast radius as the underlying role for actions the agent decides to take autonomously. The control model has to evaluate each tool call, not each session.

What good looks like

A modern PAM platform should govern human and non-human identities through the same policy engine, the same audit trail, and the same access model.

For traditional NHIs (service accounts, pipelines), look for support for workload identity federation and brokered short-lived credentials to eliminate the risk posed by static stored secrets entirely where possible.

For agentic identities, look for per-call policy evaluation (not just per-session), task-scoped access with hard expiry, and the ability to bind an agent's identity and permissions to the user it's acting on behalf of.

If the vendor's NHI answer is "we vault the API key," that's the legacy model, and it doesn't scale to with where the identity population is heading.

QUESTION 5

How does the platform handle credentials that genuinely can't be ephemeralized yet?

Why it matters

Every enterprise has them: applications that can't be refactored quickly, vendor-locked integrations that won't change, on-prem systems that don't support federation or short-lived tokens. Pretending these don't exist is dishonest. Locking customers into a pure-JIT model that can't handle them is operationally impractical. So is the inverse of keeping everything vaulted indefinitely just because some things can't move.

The right answer is a coexistence layer: the same platform that delivers JIT access can also vault, rotate, and govern the credentials that haven't been ephemeralized yet under one policy and audit model with a clear, tracked path to ZSP for each one as the underlying constraints change.

What good looks like

Ask the vendor how they handle the long tail of accounts that aren't ZSP-ready today. A credible answer acknowledges the reality and describes:

- The migration patterns available (workload identity federation, brokered short-lived credentials)
- The residual handling for the remainder (vaulted with rotation, tagged as technical debt, owned, revisited as constraints change)
- Whether the vaulted footprint is treated as a contracting metric or as the steady state

What you want to avoid: a vendor whose only answer is "everything goes in the vault" (the legacy model) or whose only answer is "everything is JIT-only" (which leaves a real operational gap).

QUESTION 6

What does migrating off our existing PAM platform look like? Especially for service accounts and pipelines?

Why it matters

For most enterprises evaluating a PAM change, the architectural debate is the easy part. The harder question is operational: how do we get from the platform we have today to the model we want without breaking what we already have in production?

Service accounts, CI/CD pipelines, break-glass workflows, and dual-control approvals all depend on the current platform. Migrating human admin access is the visible work; migrating everything else a vault might touch is what determines whether the program succeeds.

A vendor without a clear migration story will leave you running two platforms in parallel indefinitely, paying for both, and never reaching the security or operational outcome that motivated the change.

What good looks like

The vendor should be able to describe a phased migration, not just a flag-day cutover, that handles both the human admin workstream and the service account workstream in parallel, with clear exit criteria for retiring the legacy platform. Ask specifically:

- How are service accounts moved off the legacy vault, and where do they land?
- What has to change in the applications consuming those credentials, and how minimal can those changes be in the first phase?
- Are CI/CD pipeline tokens, break-glass accounts, and rotation cadence preserved across the cutover?
- How is the legacy platform actually decommissioned, and what's the customer's path to a single PAM platform of record?

A vendor with experience displacing vault-centric platforms will have specific guidance for each major source platform — including how to handle proprietary APIs, agents, and rotation engines — and a prescriptive sequence for getting workloads off the legacy vault before retiring it. If the vendor's migration plan is "we'll figure it out together," the cost and timeline are unbounded.

QUESTION 7

How quickly can we deploy and how much infrastructure do we have to run ourselves?

Why it matters

Implementation complexity is one of the most consistent reasons PAM programs miss their targets. Vault-centric platforms typically require an array of infrastructure — vault servers, session brokers, password change agents, distributed engines, proxies — that can take months to deploy, demands coordination across networking, IT, and security teams, and becomes an ongoing operational burden. Every additional component is something to patch, monitor, scale, and pay for.

For enterprises that have standardized on SaaS for almost everything else, that infrastructure model is increasingly out of step with how the rest of the stack is run.

What good looks like

A modern PAM platform should be cloud-native and SaaS-delivered, with minimal required on-prem servers, no agents on user devices, and no proxies in the network path. Time-to-first-value should be something that can be measured in weeks for an initial scope, not quarters. Ask specifically:

- What infrastructure do we have to install for the use cases that need to be met, and what do we have to maintain?
- Are there agents on managed endpoints or on target systems? Are they required for the platform's functionality? If so, what needs to be done to maintain these?
- Does the platform offer open APIs and SDKs so we can build integrations in-house, or are we dependent on vendor professional services for every customization?

Lightweight API-first architectures shift the operational burden from your team to the vendor and shorten the gap between contract signature and risk reduction.

QUESTION 8

Once standing credentials are gone, what does day-to-day operations and audit look like?

Why it matters

The vault model imposes hidden operational costs that compound over time. Quarterly attestation requires reviewing every standing privilege and explaining why it's still needed. Audit trails span request workflows, vault check-outs, session recordings, and rotation logs — across multiple consoles. Access reviews depend on self-attestation rather than actual usage. Break-glass workflows are tracked manually. Every new system adds another integration, another set of credentials, and another rotation policy.

Removing standing credentials simplifies all of this, but only if the platform was designed around that simplification, not retrofitted to it.

What good looks like

In a true ZSP model, there are no long-lived credentials to review during attestation, no rotation drift to remediate, and no separate session-broker logs to reconcile. Every privileged action produces a complete audit record with the same shape: identity, target, justification, scope, duration, approver, source, outcome.
Ask the vendor:

- What does a quarterly access review look like in your model? Can it run against actual usage data, not self-attestation?
- Is the audit trail unified across human, service account, and agentic access?
- Does the policy engine evaluate modern conditions (device posture, ticket validation, MFA step-up, time-of-day, source location) at request time, rather than once per session?

The operational story isn't just "fewer credentials to manage." It's a smaller, simpler, and more defensible privileged access program.

Modern PAM Requires a Modern Architecture

Vault-centric PAM solved a real problem in a static world. But the environments enterprises actually run today — federated clouds, ephemeral workloads, non-human identities at scale, autonomous agents calling APIs on a user's behalf — require an access model the vault-based model was never designed to deliver. Layering JIT, MFA, or session recording on top of a vault doesn't change the underlying fact: the standing credential still exists.

Modernizing PAM isn't about adding features, but replacing the architecture. Enterprises need to shift from credential storage to runtime brokering, from session-scoped access to per-task and per-call evaluation, from human-only governance to a single policy model that covers humans, workloads, and agents.

THE RIGHT PLATFORM WILL GIVE YOU DIRECT, SPECIFIC ANSWERS TO ALL EIGHT QUESTIONS — NOT FEATURE MAPPINGS AGAINST A VAULT.