strongdm

From Vault to Runtime Authorization

Rethinking Privileged Access for Modern Infrastructure

2025

Contents

Executive Summary: The Universal Access Problem

For years, privileged access was a contained problem. A small group of administrators used a limited set of credentials to manage a finite number of servers and applications. Privileged Access Management (PAM) tools evolved to support that world: store secrets in a vault, control who can check them out, and record a session for later review.

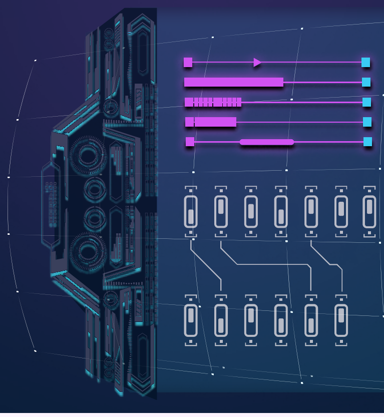
That world is gone. The access ecosystem has scaled and accelerated faster than the tools designed to control it.

Today's enterprise is defined by cloud platforms, SaaS, ephemeral infrastructure, and automation. The majority of access decisions are no longer made by people logging into machines, but by software—microservices, CI/CD pipelines, Kubernetes controllers, serverless functions, and third-party integrations. Each of these identities interacts continuously with sensitive systems and data.

This complexity has created a fundamental security gap in the middle of the access lifecycle: the moment a session opens and the user's context begins to change.

This paper explores why traditional PAM and static authorization are no longer sufficient, and introduces an updated model of runtime authorization: a unified control plane designed for the reality of modern, hybrid infrastructure.

The Changing Nature of Privileged Access



The current state of privileged access is characterized by unsustainable sprawl and fragmented visibility. Modern infrastructure no longer fits within the boundaries that traditional tools were designed to secure.

→ FROM HUMAN-CENTRIC TO MACHINE-DOMINATED

In the classic model of access, the threat surface was defined mostly by people. A privileged user, such as a system administrator or database engineer, would authenticate, retrieve a credential from a vault, and log into a target system. Tools were designed to manage that sequence: who could retrieve which credential, how long the session could last, and how activity would be recorded.

In a contemporary environment, that pattern still exists but has been dwarfed by machine-driven access: for every person in your organization, there are now multiple machine identities making authorization decisions without human intervention. Modern infrastructure has effectively inverted the ratio—what used to be 90% human decisions and 10% machine decisions is now overwhelmingly automated.

These aren't passive service accounts sitting dormant in an LDAP directory. They're Kubernetes pods spinning up by the thousands, CI/CD pipelines authenticating hundreds of times per deployment, microservices calling each other every millisecond, GitHub Actions workflows deploying across clouds, and serverless functions executing millions of times daily, each one requiring credentials and scoped permissions.

In this sense, privileged access is no longer a rare exception—it now shows up in the sheer volume and velocity of machine-driven authorization decisions, where any identity can become privileged at the moment of action. That reality demands continuous control over when and how privilege is exercised, rather than occasional, one-off management.

→ A HYBRID, FRAGMENTED ESTATE

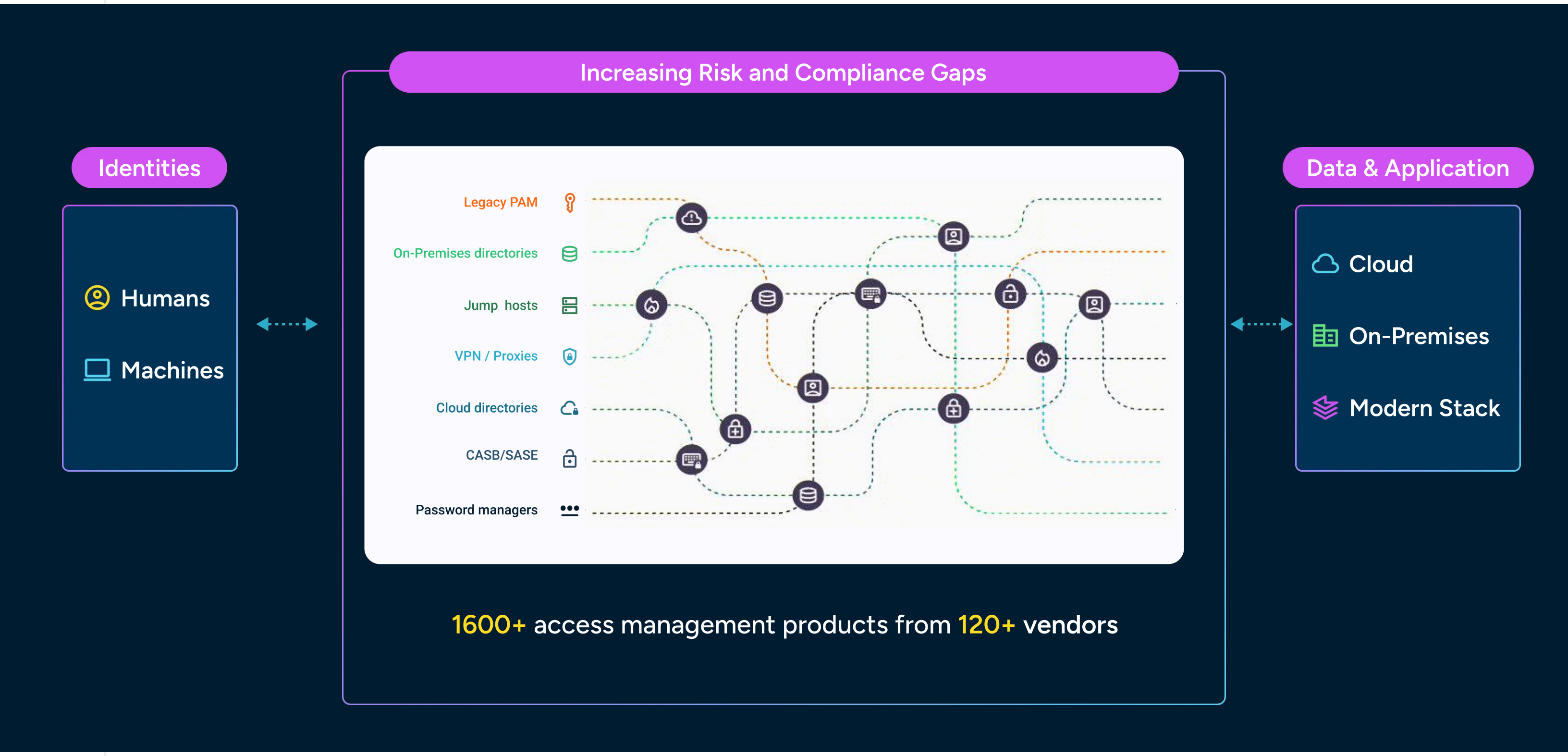
At the same time, the environment in which these identities operate has become dramatically more complex. A single organization might now span multiple cloud providers, several generations of database technology, Kubernetes clusters, legacy virtual machines, on-premises systems that cannot be retired, and a large collection of SaaS applications.

The average organization now operates across 1,295 different cloud services [1]. This isn't just about having accounts in AWS, Azure, and GCP. It's about the fractal complexity of modern infrastructure.

Each platform ships with its own access model and tools. Cloud providers offer their own IAM systems; databases have built-in roles and permissions; VPNs and jump hosts enforce network boundaries; password managers and vaults hold secrets; PAM tools govern some administrative paths; SaaS applications provide their own privilege structures.

The resulting picture is one of fragmentation. Policies are expressed differently in every system. Logs are spread across multiple consoles. Visibility into who did what, where, and why is partial at best. Security teams spend significant time stitching together views of access instead of governing it from a coherent control plane.

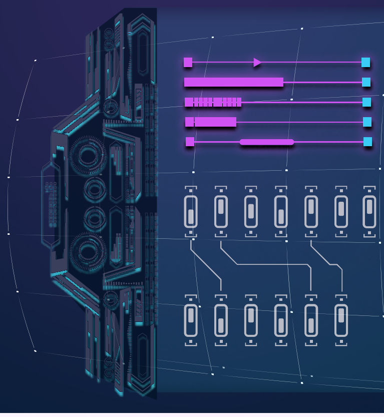
This is the setting in which attackers operate today. They do not need to break cryptography or bypass perimeter defenses if they can obtain valid credentials and move through gaps in the authorization fabric.



1600+ access management products from 120+ vendors

Figure-1: Current state of access management

An Authorization Gap in Today's Solutions



→ ONE-TIME DECISIONS IN A DYNAMIC WORLD

Most existing access control mechanisms still treat authorization as a discrete event that happens once, at the beginning of a session. A user or system presents a credential; the relevant tool checks whether that credential is valid and whether the requester is allowed to connect to the target system. If the answer is yes, access is granted and typically remains in place until the session ends.

This static model assumes that the conditions under which the decision was made remain stable. In reality, context evolves continuously. Devices can become compromised; network locations change; roles and projects are updated; new threat intelligence emerges; behavior during a session deviates from established patterns.

Despite all this, the original decision is rarely revisited. A session that began legitimately can become risky or outright malicious, yet the system continues to honor the initial authorization of access. That window —this gap between session start and the point at which detection or response eventually occurs—is the authorization gap: the period between session start and detection, where today's tools have no effective control. This is the essence of the "verify once, trust the session" problem.

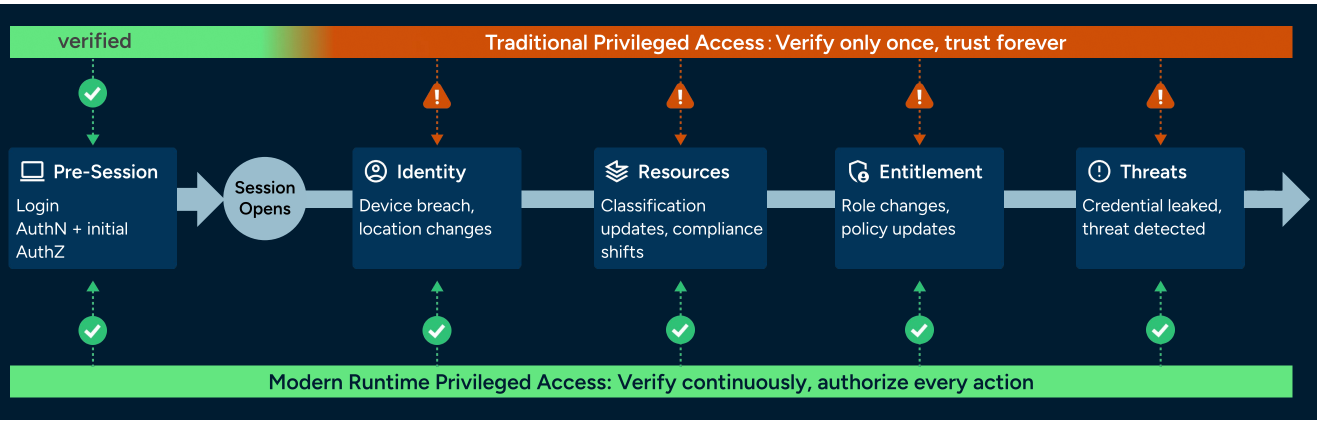

verified
Traditional Privileged Access : Verify only once, trust forever
Pre-Session
Login
AuthN + initial AuthZ
Session Opens
Identity
Device breach, location changes
Resources
Classification updates, compliance shifts
Entitlement
Role changes, policy updates
Threats
Credential leaked, threat detected
Modern Runtime Privileged Access: Verify continuously, authorize every action

Figure-2: Context Changes Faster Than Sessions

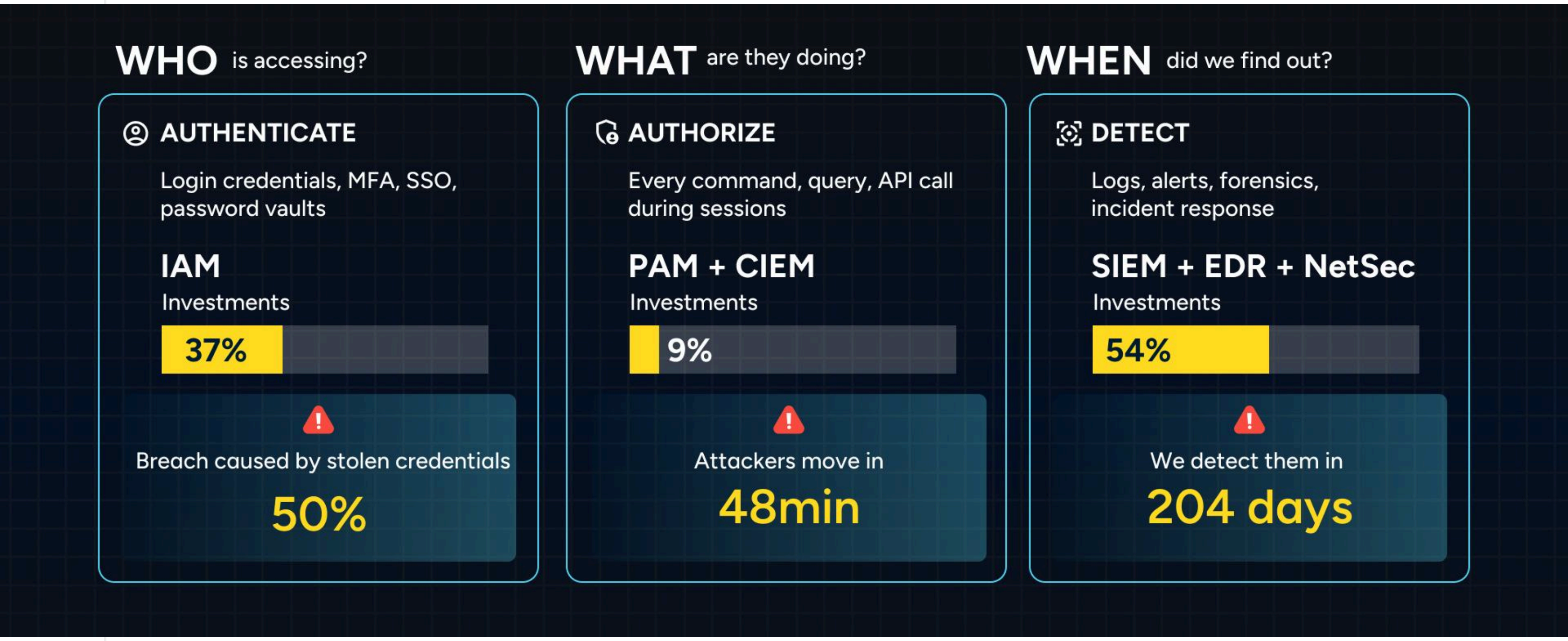
→ THE INVESTMENT IMBALANCE

Security investments over the past decade have understandably focused on two areas. The first is identity and authentication: directories, IAM platforms, SSO providers, and increasingly strong forms of multi-factor authentication. These reduce the likelihood that an attacker can impersonate a user or machine easily.

The second is detection and response. Organizations have invested in SIEM, EDR, network analytics, and behavioral monitoring to identify and understand suspicious activity after it has begun.

The layer in the middle—authorization—has not received comparable attention. Tools that decide, in real time, what a given identity may do on a given resource, under current conditions, are relatively sparse and often limited to specialized environments. As a result, when credentials are compromised, much of the protection provided by strong authentication can be bypassed, and detection systems are left to catch the resulting activity after the fact.

The consequence is an authorization gap: an interval between a successful login and eventual detection in which an attacker can operate with the full powers of a legitimate user or system. In a world of automated access and fast-moving workloads, that gap is both wide and dangerous.



WHO is accessing?

AUTHENTICATE

Login credentials, MFA, SSO, password vaults

IAM Investments

37%

Breach caused by stolen credentials

50%

WHAT are they doing?

AUTHORIZE

Every command, query, API call during sessions

PAM + CIEM Investments

9%

Attackers move in

48min

WHEN did we find out?

DETECT

Logs, alerts, forensics, incident response

SIEM + EDR + NetSec Investments

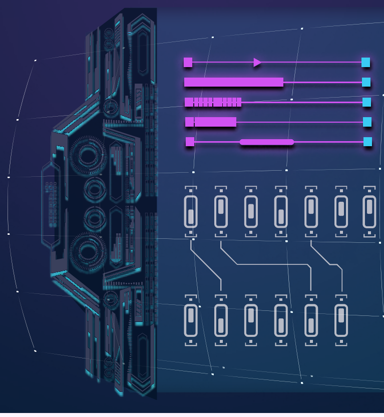
54%

We detect them in

204 days

Figure-3: We invest in doors and alarms but the breach unfolds in the middle [2]

The Case for Runtime Authorization



The old distinction between "privileged access" (special, requiring extra controls) and "regular access" (normal, lower risk) has collapsed.

In the earlier model, a privileged access decision was relatively simple. An administrator would check out a credential from the vault, log into a server, perform maintenance, and log out. Across a large enterprise, this might have amounted to tens of thousands of events per day.

Today, that number can be orders of magnitude higher—potentially in the billions.

Every API call between microservices is an authorization decision. Every database query from a containerized application. Every attempt by a CI/CD pipeline to deploy code. Every Terraform plan execution. Every serverless function invocation.

Each one requires an answer to the question:

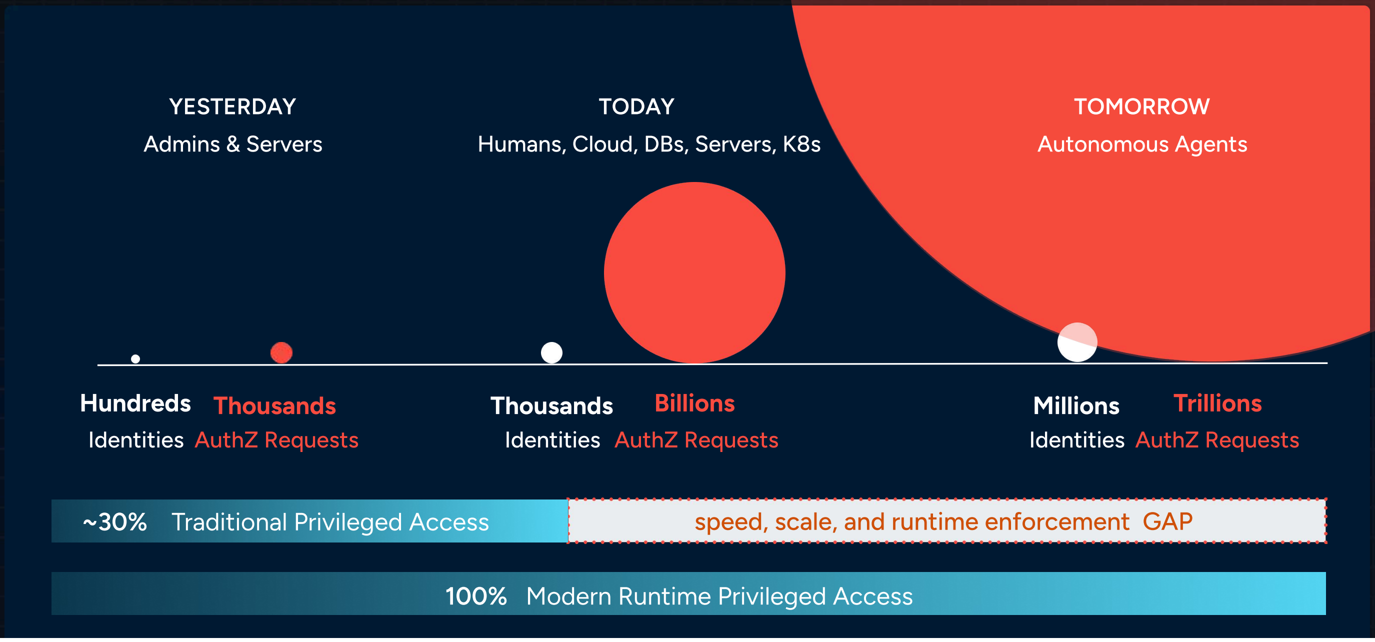
"Should this identity be allowed to do this action right now?"

The old model can't scale to this reality. It's not a matter of buying more licenses or adding more vault instances. The architectural pattern of "check out credentials at the beginning, use them for hours or days" simply cannot handle billions of ephemeral, context-dependent authorization requests. The gap between what legacy privileged access can enforce and what runtime privileged access demands is widening into an authorization gap that exposes organizations to breaches, outages, and compliance failures.

Looking ahead, the rise of autonomous agents will amplify this trend. AI-driven systems will initiate and coordinate access at scales and speeds that humans cannot supervise directly, further increasing the volume and sensitivity of authorization decisions.

THE PROBLEM

Identity explosion meets Cloud + AI



YESTERDAY
Admins & Servers

TODAY
Humans, Cloud, DBs, Servers, K8s

TOMORROW
Autonomous Agents

Hundreds Identities **Thousands** AuthZ Requests

Thousands Identities **Billions** AuthZ Requests

Millions Identities **Trillions** AuthZ Requests

~30% Traditional Privileged Access | speed, scale, and runtime enforcement GAP

100% Modern Runtime Privileged Access

Figure-4: Identity explosion meets Cloud & AI

To address this gap, authorization must be reimagined as a continuous process rather than a single checkpoint. Instead of merely confirming that a user or system is allowed to establish a session, the environment must repeatedly ask a more precise question:

> **Should this identity be allowed to perform this action, on this resource, at this moment, given everything we currently know?**

Answering that question requires more than a static role or group assignment. It depends on several dimensions of context:

- **Who is acting?** Is this a human or a machine identity? What is its role, history, and typical behavior?
- **From where and how?** Which device, network, and location are involved? Is the endpoint managed and healthy? Is the access path expected?

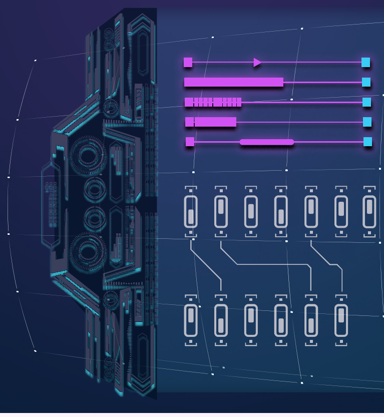
- **What is being touched?** How sensitive is the target system or data? Is this environment production, development, or something else?
- **What is the current level of risk?** Are there active incidents, elevated threat levels, or indicators that credentials may be compromised?
- **What is the temporal pattern?** Does the timing of the request align with normal usage? How has behavior evolved during the session?

Runtime authorization is the continuous, context-aware evaluation of privileged actions throughout the lifecycle of a session. It uses these inputs to adjust decisions on the fly. Access can be granted, limited, stepped up (for example by requiring additional verification), or revoked entirely based on how context evolves. In this model, trust is not a binary condition but a state that is continuously re-evaluated.

Traditional PAM architectures, built around vaults and static session policies, are not designed to operate at this level of granularity or speed, especially for machine identities operating at scale.

This is why a new model is required.

A New Model for Privileged Access



The old distinction between privileged and regular access has collapsed because the potential for privileged action is pervasive across modern infrastructure. The new model treats every access as a privileged access and every privileged interaction, whether initiated by a human or a machine, as subject to continuous authorization. Rather than treating privileged access as an exceptional case requiring special handling, this approach assumes any identity can, at some point, exercise privileged capabilities, necessitating that all access be governed by consistent, dynamic authorization principles.

Implementing this model requires an architectural element that sits between identities and resources, evaluating every privileged request in real time. This acts as a control plane for authorization decisions.

Much like a network firewall inspects packets flowing between network segments, this new architecture inspects privileged actions flowing between identities and systems. It receives context from identity providers, device security tools, threat intelligence platforms, and the resources themselves. It evaluates policy dynamically, makes authorization decisions in milliseconds, and records every decision and action for analysis and audit.

This represents a fundamental shift: from static permission assignment to dynamic authorization evaluation.

→ THE THREE FOUNDATIONAL CAPABILITIES

A functional runtime authorization architecture rests on three interdependent capabilities that work in concert:

1

Comprehensive Visibility

The system must provide a unified view of privileged activity across the entire environment. This means showing, in one place, who accessed which system, from where, using which identity, what actions they performed, and when.

This goes well beyond recording that a session existed. It requires capturing meaningful actions—specific commands executed, queries run, API calls made, configuration changes applied—and organizing them in a way that enables search, correlation, and investigation. It must transform fragmented logs scattered across dozens of systems into a coherent, queryable record of privileged activity.

2

Real-Time Control

Policies cannot simply be mapped to static roles and forgotten. The system must evaluate each request against the current context: the identity's posture, the device's health, environmental factors, resource sensitivity, and active threats.

Access should be granted just-in-time for specific tasks, scoped to particular operations, and automatically revoked when conditions change or time expires. Credentials themselves should be ephemeral, generated dynamically for each session and discarded immediately afterward, eliminating the risk of long-lived secrets being stolen or leaked. When behavior during a session deviates from expectations—unusual commands, anomalous access patterns, signs of compromise—the system should be able to restrict or terminate that session immediately.

In short, authorization decisions should remain valid at the moment of action, not just the moment of login.

3

Centralized Governance

Access policies must be defined centrally and applied consistently across diverse infrastructure: cloud services, databases, Kubernetes clusters, legacy systems, and modern platforms alike.

The architecture should make it practical to implement least privilege by default, reduce or eliminate standing privileges entirely, and enforce separation of duties for sensitive operations. It should provide clear, reliable records that demonstrate control to auditors and regulators without requiring manual reconstruction of events. It turns policy from fragmented point-solution configurations into a unified, enforceable framework.

//

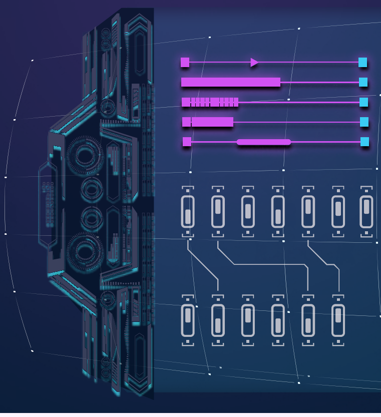
Together, these three capabilities transform identity from a static element in a directory into a dynamic instrument of security.

When all three capabilities converge, the result is an authorization model that:

- Adapts to change rather than assuming static conditions
- Scales to billions of decisions rather than hundreds
- Works across all environments rather than siloed systems
- Protects during the session rather than only at login or after detection

This is the architectural foundation required to secure modern infrastructure where context changes faster than sessions, machine identities outnumber humans, and authorization requests grow exponentially.

Path Toward Runtime Authorization



Privileged access is no longer a narrow problem limited to a small set of administrators and static servers. It is a pervasive feature of modern, automated infrastructure—exercised continuously by both people and machines. Tools designed for a credential-centric, point-in-time world cannot keep pace with this reality. When authorization is treated as a one-time decision at session start, it creates a wide window in which attackers can freely operate using legitimate credentials while detection systems race to catch up.

Runtime authorization offers a more resilient approach. By evaluating context continuously, governing privileged actions as they occur, and unifying visibility across platforms, it elevates identity into a true security control plane. This shift closes the gap between authentication and detection, reducing both lateral movement opportunities and dwell time.

Adopting runtime authorization does not require abandoning existing IAM, endpoint, or monitoring investments. In most organizations, it complements and strengthens these systems by adding a missing enforcement layer. Practically, adoption tends to follow an incremental, measurable path:

1

Start with High-Value Systems

Identify a limited set of critical assets—often production databases or administrative interfaces—and ask a simple question: Can we clearly see who is doing what here, in real time?

2

Introduce Runtime Controls

Once visibility is established, apply runtime authorization policies in front of those systems. Enforce just-in-time access and context-aware approvals. Early deployments typically uncover over-privileged identities and highlight opportunities to streamline access patterns.

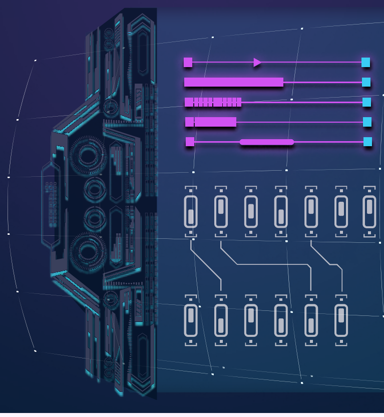
3

Expand and Mature

Extend coverage across additional environments—development, staging, cloud control planes, and internal tools. Policies evolve based on real usage insights and incident learnings, while standing privileges diminish as confidence in the model grows.

Throughout this progression, strong integration with existing identity, endpoint, and monitoring tools ensures that runtime authorization enhances the broader security posture rather than competing with it. In an environment defined by constant change, this model is no longer optional—it is a foundational requirement for securing modern systems.

References



1

Netskope Cloud Report, August 2019

https://www.netskope.com/resources/reports-guides/netskope-cloud-report-august-2019

2

50% of breaches start here: Verizon 2022 Data Breach Investigations Report

https://www.verizon.com/business/resources/T67b/reports/2022-dbir-public-sector-snapshot.pdf

3

48 min: CrowdStrike's 2025 Global Threat Report

https://go.crowdstrike.com/2025-global-threat-report.html?utm_campaign=brand&utm_conte[...]ENtX8FwO-3nYGBIo6dk_4o9WP5vohvl3bWpEIbNbX50NzcaAgrfEALw_wcB

4

204 days: IBM's Cost of a Data Breach Report, 2025

https://www.ibm.com/reports/data-breach

strongdm

STRONGDM.COM